Time Series Econometrics for Food and Energy Prices

James Vercammen

2025-05-12

Contents

Chapter 1

Preface

This graduate level textbook is for those interested in learning time series analysis through the lens of food and energy prices. Alternatively, this textbook is for those interested in learning about food and energy prices through the lens of time series analysis. This book is not about forecasting since there are many excellent alternatives. Instead, this book is about introductory applied time series econometrics. The emphasis on econometrics means that most of the content is centered on estimating simple univariate or multivariate regression models.

This textbook was written because there does not appear to exist an introductory textbook on applied time series econometrics which both highlights the relevant theory and provides data and code for estimating the various time series econometric models. The 17 data sets which are used in the various empirical applications generally consist of the prices of key food and energy commodities such as wheat, vegetable oils, potatoes, biodiesel, diesel and crude oil. The main emphasis of this book is the statistical relationship between the price of one or more food commodities and the price of one or more energy commodities.

Although the emphasis is on food and energy prices this book will be of interest to those with other policy-focused interests in statistical analysis in the food, natural resources and the environment sectors. For example, Wakamatsu [2014] uses a cointegration framework to examine the impact of MSC certification on a Japanese certified fishery and Eriksson and Lundmark [2020] uses cointegration to analyze the Nordic roundwood markets. Chambers et al. [2008] uses vector autoregression to examine the link between economic growth and threatened and endangered species listings and Minlah et al. [2021] uses Granger causality for assessing the nature of the environmental Kuznets curve for deforestation in Ghana. More details about food and resource applications of time series econometrics are provided in Chapter 1.

This textbook is best suited for those with intermediate training in economics and statistics. For those with entry level skills in using R, this textbook will

help to advance those skills to an intermediate level. The empirical applications use a *tidyverse* approach to coding, which for the most part is easy to follow. The applications also use a relatively small number of packages and functions. The various data sets are converted to *tsibble* time series objects and a core set of functions which include *ur.df()*, *ca.jo()*, *ARIMA()*, *VARselect()*, *VAR()* are used for most of the time series analysis. The coding of the empirical applications is not the central feature. The emphasis is on the theoretical and statistical relationships between the economic and non-economic time series variables which enter the various empirical applications.

1.1 Context

At the time of this writing most countries are experiencing high rates of inflation, particularly in food and energy markets. The price surge in these markets are being attributed to high post-pandemic global demand for goods and services, on-going supply chain disruptions and the war-induced loss of food and energy commodities from Ukraine and Russia. Despite emerging famines in countries such as South Sudan and Yemen due to internal conflict, drought and unusually high food prices, the global markets for ethanol and other biofuels remain strong. The use of corn, soybeans and other food crops for conversion into energy during this current food crisis has been strongly criticized [Levi and Molnar, 2022] and [Halland et al., 2022]. The current high cost of natural gas is also causing the price of fertilizer to surge, which is another link between high food and energy prices (Hebebrand and Laborde, 2022).

The goal of this textbook is to provide a time series framework for statistically analyzing food and energy prices. Time series analysis is often used for forecasting, especially in recent years with the emergence of powerful machine learning forecasting models. This textbook is not about forecasting commodity prices since the methods described in the many excellent forecasting books, articles and blog sites can be applied to commodity prices. Rather, this textbook uses the standard time series tools from the macroeconomic literature to examine both individual commodity prices and the co-movement of food and energy prices. Empirical models of stationarity, regressions with ARIMA errors, vector autoregression (VAR), cointegration, vector error correction (VEC) and generalized autoregressive conditional heteroskedasticity (GARCH) are central in the analysis.

Policy makers are often quick to assume a causal link between food and energy prices. The most common argument is that high energy prices increase the demand for biofuels which in turn drives up the price of food due to a reduced supply of food for human consumption. Despite the strong policy interest in this causal link, in this textbook the statistical analysis of food and energy prices is for the most part non-causal. There are time series methods which focus on causal linkages but these methods are too advanced for this textbook.

Nevertheless, measures of the short and long run associations between food and energy prices are still of considerable interest because they identify food security vulnerability in a dynamic context. Measures of impulse response are of particular interest because they show the dynamic response of food prices to shocks in energy prices.

Chapter 2

Overview of Food and Energy Prices

This introductory chapter is organized as follows. Section 2.1 provides more context and background to the issue of food and energy prices. A review of the relevant literature is provided in Section 2.3. An outline of the remaining chapters in this textbook is provided in Section 2.4 and concluding comments are provided in Section 2.5.

2.1 Background

In 2006-2008 agricultural commodity prices surged along with the price of energy, metals and other industrial commodities. After a short reprieve during the great recession, the prices of agricultural commodities surged again, most noticeably during the mid 2010 - mid 2012 period. The most recent surge in commodity prices (especially wheat, vegetable oils, crude oil and natural gas) began in late 2020, which is when countries began to emerge from the first phase of the COVID pandemic. This particular price surge strengthened considerably in early 2022 with the onset of the Russian-Ukraine war.

Despite warnings by economists that correlation does not imply causation, non-economist policy makers are often quick to assume that the strong co-movement between agricultural commodity prices and energy prices is the result of biofuel policies and fertilizer linkages. Biofuel polices divert crops such as corn and soybeans into ethanol and biodiesel production, and higher prices for nitrogen fertilizer (which is made from natural gas) are passed on to food consumers. While it is true that these linkages exist what is less certain is the extent that they are responsible for driving the observed price comovements. Another example where correlation was interpreted as causation (and later proven to be

incorrect) is the 2006-2008 surge in the price of agricultural commodities and the simultaneous surge in institutional long positions in agricultural commodity futures. This so-called financialization of food markets was believe to be an important reason for the strengthening of the price co-movement for agricultural and energy commodities.

Time series econometrics is a powerful set of tools which can be used to carefully assess the short run and long run statistical properties of a commodity's price and the statistical pricing relationship across multiple commodities. As such, time series econometrics is well suited to evaluate the extent that the comovement between food and energy prices has strengthened in recent years. Careful analysis allows us to distinguish between common macroeconomics factors such as a surge in global GDP or a reduction in the real rate of interest which drive up the prices of all commodities from the specific structural relationships which link food and energy markets. Examples of relevant time series models include regression with ARIMA errors, vector autoregression (VAR), cointegration and vector error correction (VEC). Vector error correction models (VECM) are particularly valuable because they allow short run pricing deviations from long run structural links to be identified. Another class of models allow measurements of the extent that price volatility in one market (e.g., energy) is responsible for concurrent or lagging price volatility in another market (e.g., food). These volatility spillover relationships make use of the well known generalized autoregressive conditional heteroskedasticity (GARCH) model.

The original set of time series tools which were developed in the 1970 - 1990s were relatively simple to understand and implement. Over time these tools have become much more sophisticated and more difficult to understand and implement. For example, it is now common to test for cointegration with panel data rather than a pair of univariate data series. As well, vector autoregression (VAR) models and vector error correction models (VECM) are now routinely estimated while accounting for structural breaks and implementing thresholds and regime switching. Software such as Stata, E-Views and R handle these extensions without too much difficulty but if the user does not have a good understanding of the underlying procedures the results which are generated have questionable value.

A general goal of this textbook is to demonstrate how the various times series packages and functions in R can be used to conduct standard time series analysis of energy and food prices. A particular goal is to avoid "black boxes" by ensuring that all procedures are transparent, typically by implementing the procedure manually before turning to the package. Time series analysis is a large and complex discipline which can initially be rather intimidating. Consequently, a second particular goal of this textbook is to keep the analysis as simple as possible while still generating results which have real-world policy relevance.

Original time series analysis in R was conducted using the *ts*, *xts* and *zoo* packages, none of which were compatible with R's popular *tidyverse* collection of packages. The *tsibble* package added time series capabilities to the *tibble* pack-

age, and by doing so allowed users to conduct time series analysis with the familiar *tidyverse* grammar and programming style. The *tsibble* package is largely designed to facilitate forecasting [Hyndman and Athanasopoulos, 2021] and so it does not have the full set of time series data analysis functions. In this book all time series data are converted to *tsibble* objects and in most cases the associated *tsibble* functions are adequate.

2.2 Time Series Econometrics vs Regular Econometrics

What are the distinguishing features of time series econometrics? The most obvious distinction is that time series econometrics must have data in time series format. In particular, time series data must be a sequence with a time order that has regular intervals. In cases where daily data is available for the weekend only, normally an assumption is made that the data which is generated on a Monday is linked to the data which was generated on the previous Friday. In other words, nothing happens on the weekend which affects the Monday data. This assumption is obviously not valid but few alternatives exist to deal with this regularly occurring gap in the data. This data gap problem is not an issue in this textbook because with one exception the data has been collected on a monthly basis or the daily data has been aggregated to a weekly or monthly level.

Time series econometrics combines time series analysis and regular econometrics. Time series analysis often involves a single variable (e.g., extracting seasonal adjustments) whereas regular econometrics will always involve at least two variables. Time series analysis is mainly used for forecasting whereas time series econometrics is used for both forecasting and the estimation of policy relevant parameters (e.g., impulse response, speed of adjustment, volatility spillover). As previously noted, this textbook mostly focuses on the estimation of policy relevant parameters.

Regular econometric models are mainly used to estimate causal relationships. For this reason, they typically have a set of exogenous control variables. In contrast, all of the variables in a time series econometric model are typically endogenous (e.g., test for cointegration) and there is no attempt to make causal inference. When forecasting with a regular econometric model the emphasis is on structural equations which explain how the endogenous variables are simultaneously determined. In contrast, when forecasting with a time series econometric model the lagged values of the endogenous variables within the model are often the only variables which serve as predictors.

Two of the most important concepts in time series econometrics are stationarity and autocorrelation. Variables which are not stationary and are not cointegrated must enter the time series econometric model as a first difference, and it doing so

valuable information is typically lost. In a time series model autocorrelation is sizeable because current values are typically highly correlated with past values. To eliminate this autocorrelation the dependent variable is normally modeled as an autoregressive stochastic process with multiple lags and/or the error term is modeled as a moving average with multiple lags. In contrast, in a regular econometric model time trends or time fixed effects are typically used to control for time dependent omitted variables. Similarly, in regular econometrics with adequate control variables autocorrelation is typically not a problem or can be dealt with by including one lag of the dependent variable on the right side of the equation.

Time series econometric models are particularly good at distinguishing between long run equilibrium relationships and short run dynamics. For example, in a vector autoregression (VAR) estimates of the impulse response show how the impact on variable y from a shock to variable x plays out over time. The estimate of the λ parameter in a vector error correction model (VECM) is a measure of the short run speed of adjustment of that variable towards its long run equilibrium path. In a regular econometric model these types of dynamic relationships are seldom of interest. Similarly, a regular econometric model is not well suited to model volatility and volatility spillovers, both of which are important areas of application in time series econometrics.

This section concludes by listing the number of entries in a Google Scholar search with the exact phrase "time series econometrics" and one additional search term appearing any where in the document (see below). The search was done on October 13, 2022 and it includes all Google Scholar listings from 2018 onward.

- Energy (1740)
- Commodities (1400)
- Agriculture (1360)
- Food (1030)
- Conservation (711)
- Forestry (473)
- Wildlife (218)
- Fisheries (187)

These listing numbers are still small in comparison to the volume of studies which use regular econometric models. Nevertheless the numbers do show the importance of time series econometrics in applied disciplines.

2.3 Review of the Literature

In this section key concepts which will be used in this textbook are reviewed and the corresponding literature is discussed.

2.3.1 Stationarity Overview

To motivate the concept of stationary it is useful to do a side by side comparison of the plots of the monthly prices of wheat and crude oil. This data, which comes from the Federal Reserve of Economic Data (FRED) website, was cleaned in Chapter 3 and saved as an RDS file. Let's read this data in now and then generate the plots. The procedure begins by loading the packages which are used in this chapter.

```
pacman::p_load(tidyverse,  here, tsibble, feasts, gridExtra)

wht_oil <- readRDS(here("data/ch1","wht_oil.RDS"))
```

```
wht_plot <- wht_oil %>% autoplot(P_wht) +
  labs(y = "Price of Wheat ($/bu)", x="Date")
oil_plot <- wht_oil %>% autoplot(P_oil) +
  labs(y = "Price of Crude Oil ($/barrel)", x="Date")

grid.arrange(wht_plot, oil_plot, ncol = 2)
```

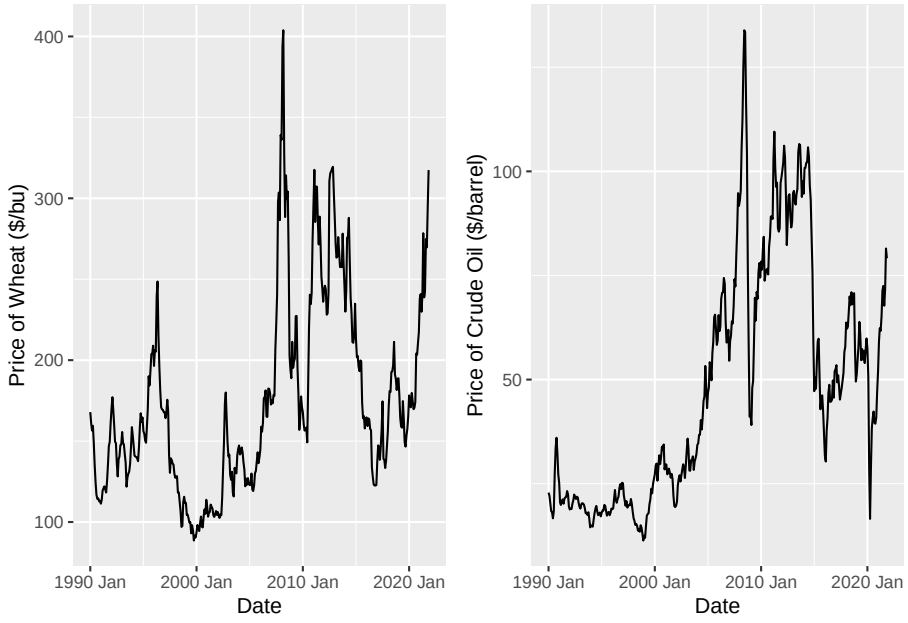

Figure 2.1: Monthly global price of Wheat and Crude Oil: 1990 - 2021

Figure 2.1 shows the monthly average price of wheat and crude oil in global markets for the years 1990 to 2021. Suppose we hypothesized that because of

biofuel policies, the price of crude oil impacts the price of wheat because wheat acreage is diverted to corn and soybean acres when oil prices surge (corn and soybeans produce ethanol and biodiesel, respectively). To test this hypothesis we regress the price of wheat on the price of crude oil as follows:

$$P_t^W = \alpha + \beta P_t^C + e_t$$

With this particular regression, the R^2 value is 0.593 and the estimated slope coefficient (equal to 1.613) is highly statistically significant. Should we conclude that the critics of biofuels are correct in their assertion that the energy prices are having a major impact on the price of food?

To answer this question let's estimate the previous equation in first difference format:

$$P_t^W - P_{t-1}^W = \beta(P_t^C - P_{t-1}^C) + e_t - e_{t-1}$$

The estimate of β should be the same as the estimate of the β in the original equation if the original error term, e_t, and the difference in the error term, $e_t - e_{t-1}$, satisfy the assumptions for standard linear regression (e.g., are white noise). What we see is that the R^2 has dropped from 0.593 down to 0.0045), the estimated slope coefficient has decreased from 1.613 down to 0.252 and the slope estimate is no longer statistically significant. Based on this method of estimation we should conclude that the price of oil does not impact the price of wheat in a significant way.

We will see in a later chapter that the second estimation method is acceptable and the first method is unacceptable. This is because the data in levels, as displayed in Figure 2.1, are non-stationary. Non-stationary data has unit roots and thus are characterized by stochastic trends. If the stochastic trends in the pair of prices happen to trend in the same direction then statistically significant econometric estimates will emerge even though there is no structural relationship between the pair of prices. For this reason, the first procedure with the pair of prices in levels is referred to as a spurious (i.e., "useless" regression). Estimating the model with the first differences of the variables eliminates the stochastic trends and so the regression results can be trusted.

The field of empirical macroeconomics was turned on its head in the 1970s when it was discovered that many of the earlier results concerning macroeconomic relationships were spurious due to the inclusion of data series in econometric models which were non stationary [Granger and Newbold, 1974, Plosser and Schwert, 1978, Davidson et al., 1978, Nelson and Plosser, 1982b, Phillips, 1986, Watson, 1986]. Despite the significant problem with spurious regression it is still common for modern-day researchers to include non-stationary data in their econometric models and in doing so contaminate their results with spurious regression results.

Are agricultural and energy commodity prices typically stationary or non-stationary? Prices can be stationary due to deterministic trends and seasonality but it most cases the reason for the non-stationarity is stochastic trends, similar to what was observed in Figure 2.1. In theory agricultural commodity prices should be stationary because the long run price of these commodities should be connected to the crop's relatively stable cost of production. In practice we tend to see a mixture of both stationary and non-stationary commodity prices.

In a comprehensive study Landajo and Presno [2022] rigorously examined the stationarity properties of renewable commodities such as grains, livestock, textiles and tropical crops over the period 1900–2018. They concluded that about half of the commodities tested have non stationary prices, with grain prices typically stationary and livestock prices typically non-stationary. For the soft commodities such as tropical crops, about half of the price series are stationary and the other half are non stationary. These outcomes highlight the importance of always testing for stationary as the starting point for any time series econometric analysis.

It is important to note that if commodity prices are analyzed from the perspective of an investor, then what matters is the return from holding the commodity in a portfolio. The first difference in the log of a commodity's price is a good estimate of the percent return from holding the commodity. For this reason the log difference of a commodity's price is commonly used in investment-focused research. The log difference series is almost always stationary and thus can be used directly in econometric analysis.

In the next two sections the literature on time series econometric modeling in the food and energy sectors is reviewed. In these two sections little is said about stationary. This should not be interpreted as meaning that stationary testing is ignored in these studies. Indeed, with few exceptions when analyzing prices (versus the log difference in prices) the modeling procedure begins with stationary testing. Nevertheless, readers may be surprised to learn that in many cases non-stationary data is being used in time series econometric analysis. This is because there is an important exception to the rule about non-stationary data. The exception, which is fully explained in Chapter 8, concerns cointegrated prices.

2.3.2 Vector Autoregression (VAR)

In agricultural and energy markets, prices will be interdependent, and current prices will be influenced by lagged values of the full set of variables. The vector autoregression (VAR) model with stationary data has been designed to accommodate these dynamic linkages. A structural VAR with two stationary variables, each with one lag, can be written as follows:

$$\begin{aligned} x_{1t} &= \phi_{11} x_{1,t-1} + \phi_{12} x_{2,t-1} + b_{11}\epsilon_{1t} + b_{12}\epsilon_{2t} \\ x_{2t} &= \phi_{21} x_{1,t-1} + \phi_{22} x_{2,t-1} + b_{21}\epsilon_{1t} + b_{22}\epsilon_{2t} \end{aligned} \tag{2.1}$$

We can write this same equation in matrix format as

$$\begin{bmatrix} x_{1t} \\ x_{2t} \end{bmatrix} = \begin{bmatrix} \phi_{11} & \phi_{12} \\ \phi_{21} & \phi_{22} \end{bmatrix} \begin{bmatrix} x_{1,t-1} \\ x_{2,t-1} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \begin{bmatrix} \epsilon_t \\ \epsilon_{2t} \end{bmatrix} \tag{2.2}$$

This model is *structural* because the b_{ij} parameters have structural policy interpretations. For example, b_{11} shows how a shock in market 1 impacts the price in that market in the current period. Similarly, b_{12} shows how a shock in market 2 impacts the price in market 1 in the current period. If market 1 is agricultural and market 2 is energy, then we are particularly interested in the sign and size of b_{12}.

The structural VAR analysis is complicated because the above system is underidentified and thus the four b_{ij} coefficients cannot be estimated directly. What can be estimated with regular least squares is the reduced form VAR:

$$X_t = \Phi X_{t-1} + U_t \tag{2.3}$$

where:

$$U_t = \begin{bmatrix} u_{1t} \\ u_{2t} \end{bmatrix} \qquad \text{and} \qquad \begin{aligned} u_{1t} &= b_{11}\epsilon_{1t} + b_{12}\epsilon_{2t} \\ u_{2t} &= b_{21}\epsilon_{1t} + b_{22}\epsilon_{2t} \end{aligned}$$

The estimated variance/covariance of the reduced form VAR contains information about how a shock in the energy market impacts the price in the agricultural market in the current period, but this information can only be recovered if identifying restrictions are imposed on the model. The simplest identifying restriction is that a shock in the agricultural market has no impact on the price in the energy market in the current period. Other less restrictive assumptions which allow for identification of the VAR are more complex.

The VAR model was pioneered by Sims [1980] but important contributions were made by Sargent and Sims [1977] and Litterman and Weiss [1985]. Sims [1980] showed that in addition to recovering estimates of the structural b_{ij} coefficients, impulse response analysis can be used to trace how a shock in one market dissipates over time in all impacted markets. Variance decomposition is another procedure which is designed to assign relative importance to the shocks in the various markets when explaining a particular price impact.

The Granger causality test, which was developed by Granger [1969], is also an important part of a standard VAR analysis. The Granger procedure is not testing whether one price "causes" the other price in the way that causation

is traditionally defined. Rather, the Granger procedure asks whether one time series is useful when forecasting future values of a second time series. In other words, the Granger procedure is concerned with "predictive causality".

Applied VAR analysis for examining agricultural and energy markets is relatively uncommon due to the restriction that variables in the VAR model must be stationary, and estimating a model with the data in first differences looses valuable information. In a highly-cited study, Kilian [2009] estimates a structural VAR with the focus on the price of oil and standard macroeconomic variables. The real price of oil is used to ensure stationary and variables such as oil production are expressed as a percentage change. The structural VAR approach allows the impact of shocks in oil supply and demand to be separately analyzed, and the bi-directional impacts between the price of oil and the general economy can be assessed.

Baumeister et al. [2014] estimated a VAR when analyzing food and energy markets but they ignored the requirement that all variables must be stationary. Their conclusion that there is little evidence that oil price shocks have impacted food prices should therefore viewed with caution. Du and McPhail [2012] estimate a structural VAR for examining the linkages between ethanol, gasoline and corn prices. They ensure data stationary by working with the log difference (i.e., percent change) in prices rather than prices in levels. The problem with this approach is that examining how the percent change in a price is impacted by shock in another market is quite different than examining how the price itself is impacted.

A nice application of a structural VAR model for the analysis of agricultural and energy markets is [Wang et al., 2014]. The structural VAR approach, estimated with prices in first differences, allows Wang et al. [2014] to distinguish how agricultural commodity prices respond to shocks in oil supply, precautionary oil demand or aggregate demand. They conclude that prior to the 2006-2008 food crisis, oil market shocks were a relatively minor determinant of agricultural commodity prices. However, this linkage strengthened considerably after the 2006-2008 food price crisis.

In a highly cited paper, Toda and Yamamoto [1995] shows that if a researcher is only interested in testing linear restrictions on the reduced form VAR (e.g., for testing Granger causality) then it is not necessary to ensure that all variables in the VAR are stationary (a polynomial time trend is used to eliminate the stochastic trends). This approach was used by Nazlioglu and Soytas [2011] and Nazlioglu [2011] when examining the interaction between agricultural commodity and energy prices. Another strand of research focuses on VAR estimation using panel data (typically multiple countries over multiple time periods). Taghizadeh-Hesary et al. [2019] use the results from an estimated panel VAR to conclude that the price of oil has a significant impact on the price of agricultural commodities.

2.3.3 Cointegration Overview

There is an important exception regarding the use of non-stationary data in a time series econometric model. Suppose a pair of prices are non-stationary and of the same order, which means that both can be made stationary with one level of differencing. In this case if both prices are cointegrated, then it is acceptable to use these variables in standard time series econometric analysis. Formally, cointegration means that even though the individual data series are non-stationary, a linear combination of the prices is stationary. The presence of cointegration implies that there exists a cointegrating vector which describes the long run relationship between the prices. We might expect the price of corn and the price of ethanol to be cointegrated because in the long run the prices of these two commodities should reflect the corn-ethanol input-output relationship.

Cointegration is important for economists because it reflects the long run economic relationship between variables. If the price of an agricultural commodity and the price of crude oil are found to be cointegrated, then the estimated coefficients in the cointegrating vector tell us the sign and the size of the long run economic relationship between this pair of commodities. If there are more then two time two series variables then there may be more than one cointegration relationship which is implied by the cointegrating vector. For example, we may expect two cointegrating relationships when analyzing the prices of corn, ethanol and crude oil because corn and ethanol are likely to have a long run economic relationship and the same for ethanol and crude oil.

If a set of time series variables are cointegrated then the short and long run relationships amongst these variables can be estimated using an error correction model (ECM) with two variables or a vector error correction model (VECM) with more than two variables. A VECM is essentially a VAR with an additional error correction term (ECT). When one or more of the prices are shocked within the system of equations, the ECT shows how quickly prices are drawn toward the equilibrium values which are implied by the cointegrating vector. The ECT is particularly important when analyzing agricultural and energy commodities because it informs us about the duration of the adjustment in food prices when the price of crude oil is shocked.

Engle and Granger [1987] did the pioneering work on cointegration and the estimation of a ECM. Their test for cointegration is designed for pairs of time series variables. First, the pair of variables are tested to ensure that they are both non-stationary and the pair is integrated of the same order. Then, one of the prices is regressed on the second price, possibly with a linear time trend included. The formal test for cointegration involves testing the residuals from this regression for stationarity. If the pair of variables are cointegrated then according to Engle and Granger [1987] the lag of the residuals from the previous regression can be used to construct the ECT, which is a key feature of the ECM estimation.

Johansen [1991] extended the cointegration test procedure to scenarios involving

more than two time series variables. His procedure is relatively complex because it involves estimating the rank of the matrix of estimated coefficients from the system of time series variables. The rank information is then used to identify the number of cointegrating relationships which exist amongst the system of time series variables. Unlike the ECM, which uses a two-step procedure for estimation, the VECM is estimated using a one-step process. The interpretation of the ECT is the same in the ECM and the VECM.

There are a large number of papers which test for cointegration and estimate a VECM in the context of agricultural and energy markets. For example, Yu et al. [2006] and Hameed and Arshad [2009] use the Johansen procedure to show that the four major vegetable oils and crude oil are cointegrated. Harri et al. [2009] show that the price of oil is cointegrated with corn, cotton and soybeans but not with wheat. Saghaian et al. [2018] test for cointegration and estimate an VECM for the case of corn, ethanol and crude oil. They find at least one cointegration relationship, and conclude that the prices are subject to overshooting during their short-run adjustments toward equilibrium. Natanelov et al. [2011] find cointegrationg relationships between crude oil and both major and non-major agricultural commodities. Estimates of the ECT coefficients are in the expected range. Zhang et al. [2010] includes sugar in the list of agricultural commodities and rather surprisingly concludes that sugar is the only agricultural commodity with a cointegrating relationship with energy commodities. Using prices from agricultural and energy markets from 1998 to 2009, Frank and Garcia (2010) determine there was a structural break in prices in September of 2006. In the 1998 - 2006 data, there are no cointegrating relationships and so a VAR model is estimated. In the 2006 - 2009 data there are cointegrating relationships and so a VECM model is estimated.

Ciaian and d'Artis Kancs [2011] construct a theoretical mode of price transmission within the agricultural, bio-energy and energy markets. Their results from cointegration analysis and the estimation of a VECM allows them to conclude that the biofuel linkage is a comparatively important price transmission pathway. Baek and Koo [2014] treat the exchange rate, energy price and commodity price as exogenous when estimating a VECM for the consumer price index (CPI) for specific food categories (e.g., baked goods). They conclude that energy and commodity prices have significant impacts on U.S. food prices. In a similar line of work, Lambert and Miljkovic [2010] use the farm product price index and manufacturing wages as exogenous variables and indexes of food prices, fuel prices and consumer income as endogenous variables within a cointegration and VECM framework. Dillon and Barrett [2016] describes three channels of transmission between energy prices and food prices. Specifically, higher oil prices raise the price of key farm inputs (e.g., fuel and fertilizer), the demand for maize used in biofuel and transport costs. Dillon and Barrett [2016] show that the transport cost linkage is particularly important in east Africa.

Nazlioglu and Soytas [2012], Rezitis [2015] and Olayungbo [2021] use comparatively powerful panel models of cointegration and VECM to examine the linkage

between agricultural commodity and energy prices. Peri and Baldi [2010] and Chen et al. [2019] allow for asymmetrix responses in their cointegration and VECM analysis. Myers et al. [2014] use a VECM model as part of their trend decomposition procedure. There are many other papers which examine the link between agricultural commodity and energy prices. See Serra and Zilberman [2013] and Filip et al. [2019] for a detail list and a categorization of the large number of papers which have contributed to this rich literature.

2.4 Outline of Remaining Chapters

This textbook consists of nine chapters beyond this introductory chapter. Chapter 2 provides an overview of the time series packages and programming methods in R. The reason for choosing to conduct most of the analysis with data formatted to *tsibble* is explained. Chapter 3 describes the specific procedures for cleaning the data which is used in Chapters 5 though 10. After being cleaned, the data for a particular chapter is stored in an *.RDS* file and later imported into the empirical application. Chapter 4 provides an overview of the stochastic processes which underlie the various time series applications. Of particular importance is the concept of stationarity, autocorrelation and seasonality of a time series.

The empirical applications are in Chapters 5 through 10. The focus of Chapter 5 is time series stationarity with featured data including the U.S. price of crude oil, the global price of wheat and the Canadian price of eggs. A regression model with ARIMA errors is used in Chapter 6 to analyze the association between the U.S. farm price for potatoes and the exchange rate between the U.S. dollar and the British pound. Chapter 7 uses data on wood chips and heating fuel to estimate a vector autoregression (VAR) model. This chapter strays somewhat from the focus on food and energy price but the central concept of substitution across the wood and energy markets is highly relevant. The concept of cointegration is introduced in Chapter 8. The data used for this chapter is the monthly price of crude oil, diesel fuel, biodiesel and soybean oil. Closely related to cointegration are vector error correction models (VECM). In Chapter 9 a VECM is estimated and used to examine the pricing relationship between diesel fuel and U.S. retail vegetable prices.

2.5 Conclusions

This chapter set the stage for formal time series econometric analysis. The methods used have been perfected in the macroeconomic literature and now enjoy widespread applications in many food, natural resource and environmental applications. R has a full suite of packages and functions which are designed to make testing and model estimation relatively easy. On other hand the functions

are often "black boxes" which do not always deliver the results which a reviewer expects. For this reason in the chapters to follow the solution is typically worked out with manual methods before turning to R's convenient time series packages and functions.

The following example illustrates why this issue is important. Consider the following regression model with one lag and one exogenous variable: $y_t = \alpha + \beta y_{t-1} + \gamma X_t + \epsilon_t$. One can easily estimate the parameters of this model using regular least squares. One can also use the *ARIMA()* function in the *vars* package to estimate this model. A reader may be surprised to learn that the estimated value of γ is different in these two cases. The reason is that the *ARIMA* function first converts the model to a regression equation with a lagged error before estimation. Specifically, $y_t = \alpha + \gamma X_t + \eta_t$ where $\eta_t = \epsilon_t + \theta \epsilon_{t-1}$. This change explains why the estimate of γ is different when the *ARIMA()* function is used rather than straight regression. Unfortunately, this difference is not obvious in the help menu for the *ARIMA()* function.

Chapter 3

Time Series Programming

The purpose of this chapter is to describe a set of commonly used time series objects, packages and functions in R. The original set of time series objects are *ts*, *zoo* and *xts*. Newer *tidyverse* compatible objects include *tsibble* [Wang et al., 2020], *timetk* [Dancho and Vaughan, 2022] and *tibbletime*. These newer objects are essentially data frames with combined *tibble* and time series properties (recall that a *tibble* is a R data frame with enhanced properties). This textbook makes extensive use of the *tsibble* object. It was chosen because of its strong programming features (e.g., *tidyverse* compatibility and panel data capabilities) and because it forms the basis of a popular textbook on forecasting with R [Hyndman and Athanasopoulos, 2021].

Section 3.1 provides a brief introduction to time series programming in R. The challenge of working with dates in R is the topic of Section 3.2. The bulk of this chapter resides in Section 3.3 where the *ts*, *zoo*, *xts* and *tsibble* time series objects are examined using heating oil futures prices as a common data set. Concluding comments are provided in Section 3.4.

3.1 Introduction

When working with time series applications in R it is important to distinguish between a time series class of object (e.g., *ts*, *zoo*, *xts* and *tsibble*), a time series package (e.g., *zoo*, *xts*, *tsibble*, *vars*, *forecast*) and a time series function (e.g., *as.ts()*, *xts()*, *lag()*, *VAR()*, *forecast()*). For example, *xts* is a class of time series object, the name of the package which utilizes *xts* objects and the name of the function which is used to create *xts* objects. In other cases the function name is different from the package name and the function is able to accommodate different classes of time series objects. For example, the *VAR()* function within the *vars* package can utilize *zoo*, *xts* or *tsibble* objects.

Someone who is new to programming time series in R is likely to find the large number of alternative time series objects, packages and functions rather confusing. Adding to this potential for confusion, online blog posts which feature time series programming methods are often out of date. In this textbook the potential for confusion is reduced because most of the analysis is conducted with *tsibble* objects, and the programming style is consistent with that used by [Hyndman and Athanasopoulos, 2021]. A *tsibble* object can be used with the various *dplyr* functions to create tidy code and keep the number of named objects to a minimum. For example, lags, differences and percent change calculations can be added to the original *tsibble* object and a variety of testing on this single data set can be accomplished using only a few lines of code.

Chapter 3 shows the steps for cleaning the various data sets which are used in the empirical applications in Chapters 5 through 10. It is during this data cleaning stage where the *tsibble* objects are created. Creating a *tsibble* typically involves first coercing the date column of imported data into R date format and then specifying that column as the index for the *tsibble* object. The cleaned data which is in *tsibble* format is saved to individual *.RDS* files. One or more of the individual *.RDS* files are imported into R at the beginning of each empirical application. During the empirical applications the *tsibble* objects typically have a passive role but occasionally functions from *tsibble* package are used as a convenient alternative to other R functions. These include *filter_index()*, *group_by_key()* and *update_tsibble()*.

Despite the large number of packages and functions in the R universe, relatively few packages and functions are used in this textbook. Namely, the *urca* package [Pfaff, 2008a] is used for testing for unit roots and cointegration in time series data. The main functions used from the *urca* package include the *ur.df()* function for testing for a unit root, and the *ca.jo()* function for simultaneously testing for cointegration and estimating a vector error correction model (VECM). The *vars* package [Pfaff, 2008a] and [Pfaff, 2008b] is used for working with vector autoregressive (VAR) models of time series data. The main functions used from the *vars* package include *VARselect()* for selecting the optimal number of lags to use when modeling a time series variable, *VAR()* for estimating a VAR model, *causality()* for testing for Granger causality and *irf()* for generating the impulse response outcomes. The *fable* package [O'Hara-Wild et al., 2021] is used for various time series forecasting procedures. The main function used from the *fable* package is *ARIMA()* for estimating an ARIMA model. In addition to these time series package, the well-known *tidyverse* suite of packages [Wickham et al., 2019] such as *dplyr* and *lubridate* is used extensively for the empirical applications.

It is useful to point out the *tsibble* object and the various functions in the *tsibble* package were designed with forecasting in mind [Wang et al., 2020]. As noted in Chapter 1, this textbook largely steers away from forecasting models and instead focuses on estimating economic relationships. The combination of not having a forecasting focus and using a *tsibble* object rather than a *ts*, *zoo* or *xts*

object makes this textbook unique. The *tsibble* object may at some point be replaced by a newer time series object with even more desirable features but it is unlikely that such a change will occur in the near future.

3.2 Date Formats for Time Series Applications

Imported time series data from a website or a csv file is usually in the form of a *tibble* data frame with an explicit date column. The values in the date column will typically have a numeric (double) format or a string format. An important step in the data cleaning process is to convert these numeric/string columns into a R date format. A R date is the number of days which have elapsed since January 1, 1970, formatted to appear as a date. Thus, if the string "22/11/2021" is in the date column of the imported data, R will implicitly convert this to 18953 since this is the number of days between January 1, 1970 and November 22, 2021. R will then use the year-month-day default format to display the date as "2021-11-22".

Let's pull in the date column from four commonly used data sources for time series analysis: *investors.com*, *Chicago Mercantile Exchange (CME)*, *Federal Reserve of Economic Data (FRED)* and *USDA-Feed Grain Database*. We begin by loading the packages required for reading in this data and for the other R applications below.

```
pacman::p_load(tidyverse, here, lubridate, xts, tsibble, timetk, urca, forecast)
```

The *read_csv()* function is used to read in the csv file which contains the date data. There are two ways to do this. First, we can manually specify the column types when the data is being read in (e.g., "character", "double", "date"). Alternatively we can choose not specify the column type and let R automatically assign a column type. Let's begin with the first approach, noting that the first and third columns contain standard dates, and the remaining columns are either characters or numeric.

```
dateX <- read_csv(here("data/ch2", "dates_sample.csv"),
                  col_types=cols(col_date("%m/%d/%Y"),
                                 col_character(),
                                 col_date("%d/%m/%Y"),
                                 col_double(),
                                 col_character()))
head(dateX)
```

```
## # A tibble: 4 x 5
##   date1      date2  date3      date4yr date4mth
##   <date>     <chr>  <date>       <dbl> <chr>
```

```
## 1 2022-09-29 22-Dec 1999-01-01    1989 Jan
## 2 2022-09-28 23-Mar 1999-01-02    1989 Feb
## 3 2022-09-28 23-May 1999-01-03    1989 Mar
## 4 2022-09-27 23-Jul 1999-01-04    1989 Apr
```

Note that when importing the first and third columns as date variables it was necessary to specify the format of the date. The date1 variable has format month-day-year and thus requires *"%m/%d/%Y"* whereas the date3 variable has format day-month-year and thus requires *"%d/%m/%Y"*. In both cases a "/" is used to separate the values which define the date. If instead the first date 1 entry was "9-29-22" it would be necessary to use *"%m-%d-%Y"* instead of *"%m/%d/%Y"*. The format for the second date2 column is specified as a character because it has an incomplete date format. The second last year column is imported as a double (representing the year) and the last month column is imported as a string (representing the month).

If we do not specify the column types when importing the csv file we would obtain the following:

```
dateY <- read_csv(here("data/ch2", "dates_sample.csv"))
```

```
## Rows: 4 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (4): date1, date2, date3, date4mth
## dbl (1): date4yr
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(dateY)
```

```
## # A tibble: 4 x 5
##   date1     date2  date3    date4yr date4mth
##   <chr>     <chr>  <chr>      <dbl> <chr>
## 1 9/29/2022 22-Dec 1/1/1999    1989 Jan
## 2 9/28/2022 23-Mar 2/1/1999    1989 Feb
## 3 9/28/2022 23-May 3/1/1999    1989 Mar
## 4 9/27/2022 23-Jul 4/1/1999    1989 Apr
```

In this case we see that all columns except the year column of the date4 variable were imported as a string. It is not clear why R did not understand that the *date1* and *date3* variables have numeric values. In any event, we can use either the *as.Date()* function from base R or the *mdy()*/*dmy()* function from

the *lubridate* package to convert the string date into a R date variable. Lets use the base R method to convert *date1* to a R date variable and the *lubridate* package to convert *date3* to a R date variable.

```
dateY <- dateY %>% mutate(
  date1 = as.Date(date1,format = "%m/%d/%Y"),
  date3 = dmy(date3)
  )
head(dateY)
```

```
## # A tibble: 4 x 5
##   date1      date2  date3      date4yr date4mth
##   <date>     <chr>  <date>       <dbl> <chr>
## 1 2022-09-29 22-Dec 1999-01-01    1989 Jan
## 2 2022-09-28 23-Mar 1999-01-02    1989 Feb
## 3 2022-09-28 23-May 1999-01-03    1989 Mar
## 4 2022-09-27 23-Jul 1999-01-04    1989 Apr
```

The *lubridate* package has a number of powerful date conversion functions. For example, we can use the *ym()* function to convert the strings in the *date2* column to a R date. Similarly, we can use the *match* function to convert the month abbreviation string (e.g., "Jan") in the *date4mth* column to a numeric representation of the month, and then use lubridate's *make_date()* function to create a R date by combining the *date4mth* and *date4yr* columns.

```
dateY <- dateY %>% mutate(
  date2 = ym(date2),
  date4mth = match(date4mth,month.abb),
  date4 = make_date(year=date4yr,month=date4mth)
  )
head(dateY)
```

```
## # A tibble: 4 x 6
##   date1      date2      date3      date4yr date4mth date4
##   <date>     <date>     <date>       <dbl>    <int> <date>
## 1 2022-09-29 2022-12-01 1999-01-01    1989        1 1989-01-01
## 2 2022-09-28 2023-03-01 1999-01-02    1989        2 1989-02-01
## 3 2022-09-28 2023-05-01 1999-01-03    1989        3 1989-03-01
## 4 2022-09-27 2023-07-01 1999-01-04    1989        4 1989-04-01
```

The *tsibble* package has a *yearweek()* function and *yearmonth()* function which can be used to change the display format of a R date. Converted objects will have class *week* and *mth* respectively. Let's reformat *date4* as a *mth* variable and display this as a new column.

```
dateY <- dateY %>% mutate(
  date4_month = yearmonth(date4)
  )
head(dateY)
```

```
## # A tibble: 4 x 7
##   date1      date2      date3      date4yr date4mth date4      date4_month
##   <date>     <date>     <date>       <dbl>    <int> <date>           <mth>
## 1 2022-09-29 2022-12-01 1999-01-01    1989        1 1989-01-01    1989 Jan
## 2 2022-09-28 2023-03-01 1999-01-02    1989        2 1989-02-01    1989 Feb
## 3 2022-09-28 2023-05-01 1999-01-03    1989        3 1989-03-01    1989 Mar
## 4 2022-09-27 2023-07-01 1999-01-04    1989        4 1989-04-01    1989 Apr
```

3.3 Four Commonly Used Time Series Objects

In this section we briefly describe the historic development and general properties of four commonly-used time series objects: *ts*, *zoo*, *xts* and *tsibble*. Regarding historic development, in the early days of R the *ts* object (inherent in base R) was a single column of data with an implicit regularly-spaced time index (e.g., monthly, quarterly, yearly). The *ts* package can be used for plotting the data and measures of autocorrelations, decomposing time series into trends, cycles and residuals, and implementing a variety of forecasting procedures. The *zoo* package [Zeileis and Grothendieck, 2005] created more powerful *zoo* time series objects. Most importantly, *zoo* time series objects accommodate time series with an irregular frequency (e.g., financial data which is reported only on non-holiday weekdays) and have enhanced vector/matrix capabilities. The *xts* package [Ryan and Ulrich, 2020] extends *zoo* by providing more functions and enhanced data conversion.

An important shortcoming of the *zoo* and *xts* objects is that they are not compatible with the overall *tidyverse* approach to programming in R (e.g., using the *%>%* and *mutate()* functions to create new columns). In contrast, the *tsibble* time series objects have been designed to work with most *tidyverse* functions such as *%>%*, *mutate()* and *select()*. As well, the *tsibble* package has functions for dealing with higher-order panel data. For example, if monthly data is collected at the U.S. state level for each of n industries, the *tsibble* index can be linked to the *year-month* column and two *tsibble* keys can be linked to the state and industry column.

The remainder of this chapter is used to showcase some of the properties of the *ts*, *zoo*, *xts* and *tsibble* objects. Heating oil futures prices will be used as the common data set in all four cases.

3.3.1 The *ts* Time Series Objects

A *ts* time series object is a column of data with an implicit regularly-spaced time stamp. The empirical applications in this textbook mainly use monthly data from the food and energy sectors. To be consistent with this focus, let's create a *ts* object using monthly heating oil futures prices. The data, which runs from November 2017 to October 2022, was downloaded from the Investing.com website (https://ca.investing.com/commodities/heating-oil-historical-data) and saved to a csv file.

We begin by reading in the data and coercing the Date column from R characters to a R Date. The data is downloaded from Investing.Com with the most recent data on top and so dplyr's *arrange()* function can be used to arrange the data from oldest to newest.

```
M <- read_csv(here("data/ch2","heating_monthly.csv"), show_col_types = FALSE) %>%
  dplyr::select(Date,Price,Open,Low) %>%
  mutate(Date = mdy(Date)) %>%
  arrange(Date)
head(M)
```

```
## # A tibble: 6 x 4
##   Date       Price  Open   Low
##   <date>     <dbl> <dbl> <dbl>
## 1 2017-11-01  1.89  1.89  1.84
## 2 2017-12-01  2.08  1.90  1.86
## 3 2018-01-01  2.07  2.07  2.04
## 4 2018-02-01  1.91  2.07  1.81
## 5 2018-03-01  2.03  1.91  1.84
## 6 2018-04-01  2.17  2.02  1.95
```

In the previous code chunk the select function was used to select the closing/settle price (Price) as well as the opening (Open) price and low (Low) price for that day. The *select()* function is preceded by the package name, *dplyr*, because the *select()* function in *dplyr* conflicts with the *select()* function in other packages.

Lets convert the Price column in the imported M data frame into a *ts* object called Mts. This requires specifying the starting month (November 2017) and the data frequency (use 12 for monthly data).

```
Mts <- ts(M$Price, start=c(2017, 11), frequency=12)
Mts
```

```
##         Jan    Feb    Mar    Apr    May    Jun    Jul    Aug    Sep    Oct
## 2017
```

```
## 2018 2.0692 1.9136 2.0284 2.1674 2.1914 2.2093 2.1319 2.2413 2.3518 2.2618
## 2019 1.8788 2.0235 1.9734 2.0812 1.8418 1.9446 1.9550 1.8282 1.9056 1.8780
## 2020 1.6245 1.4906 1.0121 0.7319 0.9647 1.1781 1.2171 1.1961 1.1454 1.0813
## 2021 1.6004 1.8565 1.7713 1.9211 2.0445 2.1287 2.1994 2.1310 2.3417 2.4964
## 2022 2.7592 3.0134 3.6912 4.7817 4.0909 3.8982 3.5215 3.6674 3.2216 3.6803
##         Nov    Dec
## 2017 1.8927 2.0755
## 2018 1.8455 1.6808
## 2019 1.8789 2.0283
## 2020 1.3559 1.4763
## 2021 2.0638 2.3301
## 2022
```

The previous output shows that our newly-created Mts object has an implicit year and month time stamp. When creating this object, the imported dates in the Date column were not used. This implicit time stamp feature is unique to *ts* objects. The remaining three objects which are examined below will each have an explicit date column which serves as a time index.

It is common to use lags and differences in time series analysis. Because of package conflict with the *lag()* function it is important to use *stats::lag*. There is no conflict with the *diff()* function and so the preceding *stats* is not required. A peculiar feature of the *ts* package is that a traditional lag is set with the negative value for the lag parameter instead of the usual positive value. Thus, to lag the Price variable one month use $k = -1$ rather than $k = 1$.

Let's use the *ts.union()* function to join the lag of Price and the Price difference to the unadjusted Price data.

```
L.Mts <- stats::lag(Mts, k = -1)
MtsL <- ts.union(Mts, L.Mts)
D.Mts <- diff(Mts)
MtsAll <- ts.union(MtsL,D.Mts)
head(MtsAll)
```

```
##          MtsL.Mts MtsL.L.Mts   D.Mts
## Nov 2017   1.8927         NA      NA
## Dec 2017   2.0755     1.8927  0.1828
## Jan 2018   2.0692     2.0755 -0.0063
## Feb 2018   1.9136     2.0692 -0.1556
## Mar 2018   2.0284     1.9136  0.1148
## Apr 2018   2.1674     2.0284  0.1390
```

The price of heating oil has a strong seasonal component. Let's observe this seasonality by using the *stl()* function to strip seasonality out of the data

```
Mts_decomp <- stl(Mts, s.window="period")
plot(Mts_decomp)
```

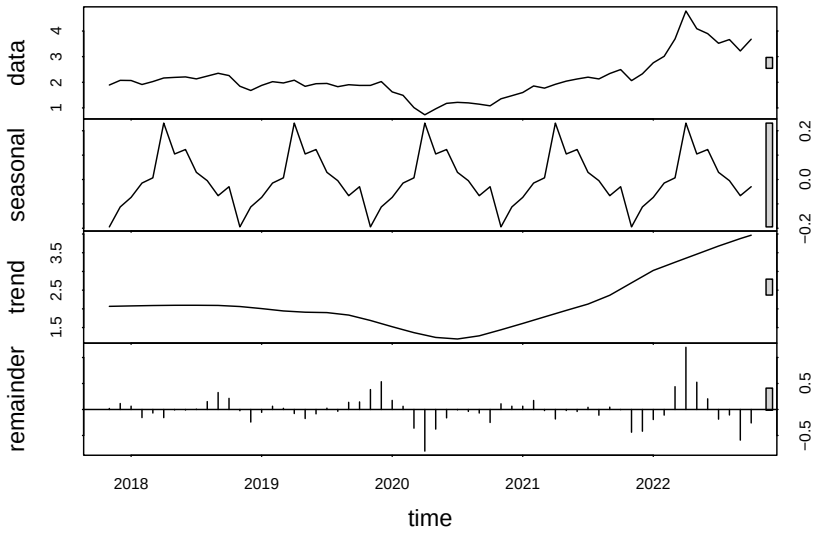

Figure 3.1: Decomposition of Heating Oil Price

The second plot in Figure 3.1 shows that indeed there is strong seasonality in the price of heating oil. The bottom plot shows the autocorrelation of the residuals which remains after a trend line and the seasonal component is removed. In later chapters we will spend a great deal of time talking about autocorrelation of the residuals.

There are several forecasting functions which are designed to work with *ts* objects. For example, the *HoltWinters()* and the *ets()* functions can be used for exponential forecasting. Similarly, the *arima()* function can be used for forecasting with an autoregressive integrated moving average (ARIMA) model.

3.3.2 The *zoo* Time Series Object

Commodities do not trade on weekends and major public holidays. Consequently, *ts* time series objects cannot be used to model daily commodity prices. The *zoo* package addresses this shortcoming by using an explicit time index rather than an implicit time stamp. The properties of the *zoo* object will be illustrated using daily heating oil futures prices. These prices, which run from

November 5, 2017 to October 5, 2022, also come from Investor.Com (see previous web address) and are read in from a csv file.

```
D <- read_csv(here("data/ch2","heating_daily.csv"), show_col_types = FALSE) %>%
  dplyr::select(Date,Price,Open,Low) %>%
  mutate(Date = mdy(Date)) %>%
  arrange(Date)
head(D)
```

```
## # A tibble: 6 x 4
##   Date       Price  Open   Low
##   <date>     <dbl> <dbl> <dbl>
## 1 2020-10-05  1.13  1.08  1.08
## 2 2020-10-06  1.19  1.14  1.13
## 3 2020-10-07  1.16  1.17  1.14
## 4 2020-10-08  1.19  1.16  1.16
## 5 2020-10-09  1.19  1.19  1.18
## 6 2020-10-12  1.16  1.19  1.15
```

An *xts* object is created by explicitly assigning the Date column to serve as a time index. It is important that the Date column be coerced from character to date format before making this assignment.

```
Z <- read.zoo(D,index.column = 1)
head(Z)
```

```
##             Price   Open    Low
## 2020-10-05 1.1333 1.0845 1.0844
## 2020-10-06 1.1886 1.1372 1.1317
## 2020-10-07 1.1608 1.1714 1.1443
## 2020-10-08 1.1923 1.1638 1.1581
## 2020-10-09 1.1933 1.1895 1.1815
## 2020-10-12 1.1571 1.1898 1.1484
```

The previous output shows that the explicit date column from the imported data has been replaced by a *zoo* index which forms the basis for all data manipulations. The weekend gap between June 10 and June 13 will be accounted for when creating summary statistics such as monthly average prices and using various other time series functions. The *str()* function confirms that Z is indeed a *zoo* object.

The following functions calculate the monthly average price, the price on the last trading day of the month and a rolling average of the price on the last five trading days.

```
agg1 <- aggregate(Z$Price,as.yearmon,mean)
head(agg1)
```

```
## Oct 2020 Nov 2020 Dec 2020 Jan 2021 Feb 2021 Mar 2021
## 1.155080 1.253738 1.447750 1.579314 1.787333 1.856048
```

```
agg2 <- aggregate(Z$Price, as.yearmon, tail, 1)
head(agg2)
```

```
## Oct 2020 Nov 2020 Dec 2020 Jan 2021 Feb 2021 Mar 2021
##   1.0813   1.3559   1.4763   1.6004   1.8565   1.7713
```

```
roll <- rollapply(Z$Price, 5, mean)
head(roll)
```

```
## 2020-10-07 2020-10-08 2020-10-09 2020-10-12 2020-10-13 2020-10-14
##    1.17366    1.17842    1.17450    1.18084    1.18012    1.17728
```

Similar to the *ts* time series we can merge lagged values and the first difference of the price variable with the original Price column. Once again the *stats::lag()* function with $k = -1$ is used to obtain a standard lag. The *na.pad = TRUE* setting ensures that "NA" is inserted in rows which no longer exist due to lagging and taking the first difference of a variable.

```
L.Z <- stats::lag(Z$Price, k = -1, na.pad = TRUE)

Z.L <- merge(Z$Price,L.Z, all=TRUE)

D.Z <- diff(Z$Price)
ZAll <- merge(Z.L,D.Z, all=TRUE)
head(ZAll)
```

```
##            Z$Price    L.Z     D.Z
## 2020-10-05  1.1333     NA      NA
## 2020-10-06  1.1886 1.1333  0.0553
## 2020-10-07  1.1608 1.1886 -0.0278
## 2020-10-08  1.1923 1.1608  0.0315
## 2020-10-09  1.1933 1.1923  0.0010
## 2020-10-12  1.1571 1.1933 -0.0362
```

3.3.3 The *xts* Time Series Object

The *xts* time series object is largely an enhanced version of the *zoo* object. If the data is read from a csv file the resulting data frame can be converted to a *xst* object. The conversion procedure is similar to the procedure we used to create a *zoo* object except now use *order.by = dates* rather than using *index.column* to identify the column containing the time index. If the data is already in *zoo* format then the *as.xts()* can be used to coerce *zoo* to *xts*. Let's use this second method now to create a *xts* object consisting of the monthly heating oil prices.

```
X <- as.xts(Z)
head(X)
```

```
##             Price   Open    Low
## 2020-10-05 1.1333 1.0845 1.0844
## 2020-10-06 1.1886 1.1372 1.1317
## 2020-10-07 1.1608 1.1714 1.1443
## 2020-10-08 1.1923 1.1638 1.1581
## 2020-10-09 1.1933 1.1895 1.1815
## 2020-10-12 1.1571 1.1898 1.1484
```

```
str(X)
```

```
## An 'xts' object on 2020-10-05/2022-10-06 containing:
##   Data: num [1:531, 1:3] 1.13 1.19 1.16 1.19 1.19 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:3] "Price" "Open" "Low"
##   Indexed by objects of class: [Date] TZ: UTC
##   xts Attributes:
##  NULL
```

Notice that the printout of the *xts* data looks identical to the *zoo* data. The *str()* function verifies that indeed X is an *xts* object.

Lets use the *indexwday()* function to extract the data for Monday through Thursday only.

```
wkday <- X[.indexwday(X) %in% 1:4]
head(wkday)
```

```
##             Price   Open    Low
## 2020-10-05 1.1333 1.0845 1.0844
## 2020-10-06 1.1886 1.1372 1.1317
```

```
## 2020-10-07 1.1608 1.1714 1.1443
## 2020-10-08 1.1923 1.1638 1.1581
## 2020-10-12 1.1571 1.1898 1.1484
## 2020-10-13 1.1690 1.1588 1.1564
```

Next we can merge together the lag of Price and the first difference of price. An important difference is in this current case *lag()* uses a positive value for k in order to create a standard lag.

```
L.X <- stats::lag(X$Price, k = +1, na.pad = TRUE)
X.L <- merge(X$Price,L.X, all=TRUE)

D.X <- diff(X$Price)

XAll <- merge(X.L,D.X, all=TRUE)
head(XAll)
```

```
##             Price Price.1 Price.2
## 2020-10-05 1.1333      NA      NA
## 2020-10-06 1.1886  1.1333  0.0553
## 2020-10-07 1.1608  1.1886 -0.0278
## 2020-10-08 1.1923  1.1608  0.0315
## 2020-10-09 1.1933  1.1923  0.0010
## 2020-10-12 1.1571  1.1933 -0.0362
```

When working with commodity prices it is common to calculate returns from holding the commodity. Returns, which is calculated as the log difference in prices, is approximately equal to the percent change in the daily price and then adjusted to account for the time period. Let's use the *apply.weekly()* function from the *xts* package to calculate the weekly volatility for the heating oil futures price.

```
Xreturn <- apply.weekly(diff(log(X$Price)),mean)
head(Xreturn)
```

```
##                   Price
## 2020-10-09           NA
## 2020-10-16 -0.002394229
## 2020-10-23 -0.004771939
## 2020-10-30 -0.012545544
## 2020-11-06  0.011028469
## 2020-11-13  0.010501816
```

3.3.4 The *tsibble* Time Series Object

The *tsibble* package is the new kid on the block. As the name suggests, a *tsibble* time series object combines the enhance properties of a data frame which is embedded in a *tibble* object with a wide array of time series properties. Let's begin with a simple *tsibble* which is constructed from the monthly heating oil prices. A *tsibble* can be created directly from imported data or by coercing a *zoo* or a *xts* object. Let's use the first method to create a *tsibble*.

The first step is to import the data and coerce character dates into R dates. The procedure is similar to what we did during the creation of the *zoo* time series object.

```
S <- read_csv(here("data/ch2","heating_monthly.csv"), show_col_types = FALSE) %>%
  dplyr::select(Date,Price,Open,Low) %>%
  mutate(Date = mdy(Date)) %>%
  arrange(Date)
head(S)
```

```
## # A tibble: 6 x 4
##   Date       Price  Open   Low
##   <date>     <dbl> <dbl> <dbl>
## 1 2017-11-01  1.89  1.89  1.84
## 2 2017-12-01  2.08  1.90  1.86
## 3 2018-01-01  2.07  2.07  2.04
## 4 2018-02-01  1.91  2.07  1.81
## 5 2018-03-01  2.03  1.91  1.84
## 6 2018-04-01  2.17  2.02  1.95
```

The Date column is somewhat cumbersome to read because it displays the first of the month day for each entry. The *yearmonth()* function from *tsibble* can be used to create a more visually appealing format for the date (let's call the new column *period* and de-select the Date column). The newly created month column can now serve as the index during the creation of the the *tsibble* object.

```
# assign "Date" as index
Sib <- S %>% mutate(period=yearmonth(Date)) %>%
  tsibble(index=period)
   head(Sib)
```

```
## # A tsibble: 6 x 5 [1M]
##   Date       Price  Open   Low   period
##   <date>     <dbl> <dbl> <dbl>    <mth>
## 1 2017-11-01  1.89  1.89  1.84 2017 Nov
## 2 2017-12-01  2.08  1.90  1.86 2017 Dec
```

```
## 3 2018-01-01  2.07  2.07  2.04 2018 Jan
## 4 2018-02-01  1.91  2.07  1.81 2018 Feb
## 5 2018-03-01  2.03  1.91  1.84 2018 Mar
## 6 2018-04-01  2.17  2.02  1.95 2018 Apr
```

The *filter_index()* function can be used to filter *tsibble* data. For example, including data from June 2020 to December 2020 due to Covid-related market volatility can be accomplished as follows:

```
Sib2 <- Sib %>% filter_index(~"2020 May","2020 Jan"~.)
```

As noted, an important benefit of working with *tsibble* time series objects is that they lend themselves to tidy programming. In the example below new columns for the log of Price and the log of Open are created and then the former is regressed on the latter. Feeding the entire *tsibble* object into the function eliminates the use of *$* and the creation of multiple univariate time series. This tidy approach to time series programming is used throughout the empirical applications.

```
fit <- Sib %>% mutate(
  lnP = log(Price),
  lnO = log(Open)
  ) %>%
  lm(lnP~lnO, .)
summary(fit)
```

```
##
## Call:
## lm(formula = lnP ~ lnO, data = .)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.37159 -0.05879  0.01379  0.06907  0.32199
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.004578   0.035523  -0.129    0.898
## lnO          1.026725   0.046999  21.846   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1212 on 58 degrees of freedom
## Multiple R-squared:  0.8916, Adjusted R-squared:  0.8898
## F-statistic: 477.2 on 1 and 58 DF,  p-value: < 2.2e-16
```

A second important feature of *tsibble* objects is that they are well suited to model multi-level panel data. To show how this works lets begin by adding a new month and year column to the data frame containing the daily futures prices for heating oil.

```
D <- D %>% mutate(Month=month(Date),
                  Year=year(Date))
head(D)
```

```
## # A tibble: 6 x 6
##   Date       Price  Open   Low Month  Year
##   <date>     <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 2020-10-05  1.13  1.08  1.08    10  2020
## 2 2020-10-06  1.19  1.14  1.13    10  2020
## 3 2020-10-07  1.16  1.17  1.14    10  2020
## 4 2020-10-08  1.19  1.16  1.16    10  2020
## 5 2020-10-09  1.19  1.19  1.18    10  2020
## 6 2020-10-12  1.16  1.19  1.15    10  2020
```

We now have the potential to summarize daily heating oil prices by month and by year. In other words we now have two dimensional panel data. To work with this panel data we need to assign both an index and a key to this *tsibble* object. We could assign the index to year and the key to month or vice versa - the results will be the same. The first approach is more intuitive and so we will go with that. However, for the *summarise* functions to work properly it is best to initially specify index as Date and later change it to Year. The new *tsibble* object is created as follows:

```
SibDay <- D %>% as_tsibble(index=Date,key=Month)
head(SibDay)
```

```
## # A tsibble: 6 x 6 [1D]
## # Key:       Month [1]
##   Date       Price  Open   Low Month  Year
##   <date>     <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 2021-01-04  1.46  1.48  1.45     1  2021
## 2 2021-01-05  1.52  1.46  1.45     1  2021
## 3 2021-01-06  1.53  1.52  1.50     1  2021
## 4 2021-01-07  1.54  1.53  1.52     1  2021
## 5 2021-01-08  1.58  1.54  1.54     1  2021
## 6 2021-01-11  1.57  1.59  1.55     1  2021
```

We are interested in finding the minimum and maximum price for each month of each year in our three year sample. This can be accomplished by first grouping

the data by key using the *group_by_key* function and then grouping the data by the index using the *index_by* function. Following this the *summarise* functions can be used to identify the maximum and minimum prices for each month of each year.

```
P_sum <- SibDay %>%
  group_by_key() %>%
  index_by(key=Year) %>%
  summarise(
    Mthmax = max(Price),
    Mthmin = min(Price)
  )
P_sum
```

```
## # A tsibble: 25 x 4 [1Y]
## # Key:       Month [12]
##    Month   key Mthmax Mthmin
##    <dbl> <dbl>  <dbl>  <dbl>
##  1     1  2021   1.62   1.46
##  2     1  2022   2.79   2.36
##  3     2  2021   1.91   1.65
##  4     2  2022   3.01   2.74
##  5     3  2021   1.97   1.75
##  6     3  2022   4.44   3.03
##  7     4  2021   1.96   1.77
##  8     4  2022   5.14   3.27
##  9     5  2021   2.07   1.95
## 10     5  2022   4.20   3.67
## # i 15 more rows
```

If instead the goal was to calculate the mean price for each year then the *group_by_key* function should be eliminated

```
P_year <- SibDay %>%
  index_by(key=Year) %>%
  summarise(
    YrMean = mean(Price)
  )
P_year
```

```
## # A tsibble: 3 x 2 [1Y]
##     key YrMean
##   <dbl>  <dbl>
## 1  2020   1.29
## 2  2021   2.07
## 3  2022   3.55
```

3.4 Conclusions

An interesting characteristic of time series analysis in R is that many of the time series functions throughout this textbook work with non-time series data. For example, the well-known Augmented Dickey Fuller (ADF) test works with a vector of data as well as a time series object such as an *xts* or **tsibble*. Time series data is perhaps most valuable when data wrangling is required such as extracting daily averages, combining weekly data sets etc. For standard monthly data sets, which is mostly what we will use in this textbook, there is no inherent advantage in using dedicated time series objects.

Having said this, time series packages have tools for identifying and imputing values for missing data. This is important because many time series functions assume regular data frequency (e.g., Wednesday of each week). The *tsibble* package is particularly effective at managing data missingness. For some time series functions such as *ARIMA* in the *Fable* package, the function will not return a result if there are gaps in the data. Converting *tibbles*, matrices and vectors into time series objects is not difficult and so it is a good idea to always use time series objects such as *xts* and *tsibble* even if this is not an explicit requirement of the function.

Chapter 4

Data Cleaning

This chapter shows the R programming steps which are used to clean the data for the various empirical applications in this textbook. For each chapter the raw data is read into R either as an API (full details are provided below) or from an Excel or csv file. The cleaned data for each chapter is saved in one or more *.RDS* files. These *.RDS* files are read back into R at the beginning of each empirical application.

Section 3.1 provides a brief introduction, which includes a link to a blog post about data cleaning best practices and the links to the various websites which contain the raw data to be cleaned. In Section 4.2 the general data import procedures are described and the various R packages which are used for the data cleaning are loaded. The data cleaning procedures for each of the empirical applications in Chapters 1 and 4 through 10 are described in Section 4.3. Section 4.4 contains the optional code for saving the cleaned data sets to individual *.RDS* files.

4.1 Introduction

Time series raw data is seldom ready to be analyzed. In some cases simple steps such as filtering for the correct range of dates and identifying the date column for creating the time series variable is all that is is needed. In other cases, considerable data wrangling is required. This includes omitting rows of a csv or Excel file when importing, converting column names into strings which are compatible with R (e.g., no spaces) and converting data which arrives in character format into date format. The term data cleaning refers to the process of importing the data and preparing it for analysis. In this textbook the cleaned data is saved to a RDS file rather than csv file because RDS files are much smaller and thus quicker to read. As well, storing raw data in either an csv or Excel file

and storing cleaned data in an RDS file is an effective way of keeping the two types of data readily distinguishable.

Coding practices vary across individuals but it is nevertheless important to adhere to coding best practices when cleaning data. A good summary of the general coding practices which are relevant for R can be found at https://style.tidyverse.org/syntax.html.

The specific data cleaning procedures are shown for:

- Chapter 1: Global price of wheat and crude oil.
- Chapter 4: Bananas, CPI, Crude oil prices (daily) and potato prices
- Chapter 5: CPI, Industrial production, wheat and crude oil prices, egg prices
- Chapter 6: Gasoline, electricity and simulated breaks.
- Chapter 7: Fruit, gasoline, corn, simulated ARIMA, energy consumption and hog prices
- Chapter 8: Wood chips and heating oil prices
- Chapter 9: Vegetable oil prices and biofuel prices
- Chapter 10: Canada and U.S. crude oil prices, and Veggie CPI and diesel fuel prices
- Chapter 10: To be determined.

Unless otherwise noted, all data are U.S. measures. The on-line links to these data sets are as follows:

Global Wheat Price

https://fred.stlouisfed.org/series/PWHEAMTUSDM

Monthly price of West Texas Intermediate (WTI) Crude Oil

https://fred.stlouisfed.org/series/MCOILWTICO

Daily Crude Oil Price via a web API

- https://eodhistoricaldata.com

Consumer Price Index

- https://fred.stlouisfed.org/series/CPIAUCSL

Farm Price of Potatoes

- https://fred.stlouisfed.org/series/WPU01130603

Industrial Production

- https://fred.stlouisfed.org/series/IPUTIL

Wheat

- https://fred.stlouisfed.org/series/PWHEAMTUSDM

Eggs, Alberta, Jumbo

- https://agriculture.canada.ca/en/market-information-system/rp/index-eng.cfm?action=pR&r=16&pdctc=&wbdisable=true

Temperature Anomaly

- https://www.ncei.noaa.gov/access/monitoring/national-temperature-index/time-series/anom-tavg/1/12

Energy Consumption

- https://www.eia.gov/totalenergy/data/monthly/

Potatoe Prices

- https://fred.stlouisfed.org/series/WPU01130603

U.S. Dollar - British Pound Exchange Rate

- https://fred.stlouisfed.org/series/EXUSUK

Wood Chips and Heating Oil

- https://fred.stlouisfed.org/series/DCOILWTICO
- https://fred.stlouisfed.org/series/PCU3211133211135
- https://fred.stlouisfed.org/series/WPU083
- https://fred.stlouisfed.org/series/WPU05730201

Vegetable Oils

- https://www.worldbank.org/en/research/commodity-markets

Biofuels

- https://www.card.iastate.edu/research/biorenewables/tools/hist_bio_gm.aspx
- http://www.eia.gov/oog/info/wohdp/diesel.asp
- https://www.eia.gov/dnav/pet/hist/LeafHandler.ashx?n=PET&s=RWTC&f=W

Canada and U.S. Crude Oil Prices and Exchange Rate

- https://economicdashboard.alberta.ca/oilprice
- https://fred.stlouisfed.org/series/TWEXBGSMTH

Lumber

- http://blogs.ubc.ca/jvercammen/files/2022/04/Lumber-Assortment-USA.csv
- https://unece.org/sites/default/files/2021-09/PriceOutputTable.xls
- https://fred.stlouisfed.org/series/EXSDUS

4.2 Data Import and Packages

There are three common ways to import data into R: (1) packaged API method; (2) API using URL address; and (3) read data from a locally-stored Excel or csv file. We will use all three methods in this chapter.

The widely used *tidyverse* package includes *readxl* and *readr*. Note that *readxl* is not part of the core *tidyverse* package and so it must be loaded using "library(readxl)" even if the *tidyverse* package has been loaded. In contrast, *readr* is part of the *tidyverse* core and so it is not required to use "library(readr)" if the *tidyverse* package is loaded.

For some of the applications we will use the *read_excel()* function from the *readxl* package to read in data from an Excel file. We will use the *read_csv()* function from the *readr* package when reading from a csv file. If the date is in text format with a tab separation or some other data format then the *read_tsv()* or *read_delim()* functions should be used.

When the data is read in from Excel or csv there is no need to include in the read command a statement which specifies how the data is organized (e.g., sep = ',').

In most cases the top row of the data contains the names of the columns (i.e., the header). If your data does not have a header then you need to use "header = FALSE" in your Excel or csv read statement.

In the older versions of R, it was important to use "stringsAsFactors=FALSE" in the Excel or csv read statement. With current versions of R this restriction is

now the default setting and so it is no longer necessary to include this statement when importing data.

If you believe there are white spaces in some of the data (e.g., "re d" instead of "red") then the data can be read in from Excel or a csv file with "strip.white = TRUE" included in the read statement.

When reading in data the default is for R to assign to "1, 2, 3…" as row names. If you do not wish row names then use "row.names = FALSE" in the read statement.

The four data cleaning modules below have a similar format.

- Import data with an API or read data from Excel or csv into a tibble object.
- Convert date columns from character format to R date format if required.
- Select columns to retain.
- Convert the tibble object to a time series tsibble object.
- Merge data sets if required.

The data cleaning concludes by saving the final tsibble object as a .RDS file. These .RDS files will be imported and used in the empirical applications in the chapters to follow.

The cleaning of the four data sets requires a number of packages to be loaded. Packages are typically loaded with the *library()* function. Rather than specifying a separate library function for each package to be loaded, the *p_load()* function from R's *pacman* package can be used instead. Specifically,

pacman::p_load(tidyverse, readxl, lubridate, here, janitor, xts, tsibble, forecast, fredr)

Loading the *tidyverse* package automatically loads the core set of related packages, which include: *ggplot2*, *dplyr*, *tidyr*, *readr*, *purrr*, *tibble*, *stringr* and *forcats*. It was previously noted that although *readxl* is part of the *tidyverse* family it is not automatically loaded when *tidyverse* is loaded. The situation is the same for the *lubridate* package. The *here* package is used as part of the R project configuration. The *janitor* package allows the *clean_names()* function to be used. The remaining packages, *xts*, *urca*, *tsibble* and *forecast* are used as part of the time series analysis.

The code for loading the packages is as follows:

```
pacman::p_load(tidyverse, readxl, lubridate, here, janitor, tsibble, forecast, fredr)
```

4.3 Applications

In this section the data cleaning procedures are described for the empirical application chapters of this textbook (Chapters 4 through 10). It begins with a brief discussion about using APIs to access data.

4.3.1 Accessing Data with an API

There are many websites which offer subscription-based data APIs for the retrieval of various types of data including commodity prices. Two well known sites include Nasdaq (formerly Quandl) (https://commodities-api.com/,https://data.nasdaq.com/) and EOD Historical Data (https://eodhistoricaldata.com). There are also a number of public agencies which offer free data APIs such as the U.S. Bureau of Labour Statistics (https://www.bls.gov/developers/) and the *FRED*, which stands for the *Federal Reserve Economic Data* (https://fred.stlouisfed.org/docs/api/fred/). In the analysis below data APIs from *eodhistoricaldata.com* and the *FRED* are used to import data directly into R.

4.3.2 Chapter 1: Monthly prices of Crude Oil and Wheat

The monthly price of West Texas Intermediate (WTI) oil can be imported directly with an API from the Federal Reserve of Economic Data (FRED) website. A free API code can be obtained by selecting *FRED Tools – FRED API* in the menu bar. The FRED API is easy to use because R has a *fredr* package which accommodates an intuitive entry of the search and read parameters. The user-specific API code is identified in R as follows:

```
fredr_set_key("a63c5424469efa8009045b7d02da6a0f")
```

There are several steps for reading in and cleaning the FRED data:

- Filter the data for a start and end date.
- Select and possibly rename the desired columns.
- Reformat the date column to *yearmonth* format (this step is used only to improve the appearance of the data and the graphs).
- Coerce the object into a *tsibble* object.

```
fred_oil <- fredr(
  series_id = "MCOILWTICO",
  observation_start = as.Date("1990-01-01"),
  observation_end = as.Date("2021-11-01")) %>%
  dplyr::select(date,value) %>%
```

```
  rename(P_oil = value) %>%
  mutate(month = yearmonth(date)) %>%
  as_tsibble(index=month)
```

The global price of wheat is also imported from FRED and is cleaned using the same procedures as crude oil.

```
fred_wht <- fredr(
  series_id = "PWHEAMTUSDM",
  observation_start = as.Date("1990-01-01"),
  observation_end = as.Date("2021-11-01")) %>%
  dplyr::select(date,value) %>%
  rename(P_wht = value) %>%
  mutate(month = yearmonth(date)) %>%
  as_tsibble(index=month)
```

The crude oil and wheat data sets can now be joined.

```
wht_oil <- fred_wht %>% inner_join(fred_oil,by="month")
head(wht_oil)
```

```
## # A tsibble: 6 x 5 [1M]
##   date.x     P_wht    month date.y     P_oil
##   <date>     <dbl>    <mth> <date>     <dbl>
## 1 1990-01-01  168. 1990 Jan 1990-01-01  22.9
## 2 1990-02-01  161. 1990 Feb 1990-02-01  22.1
## 3 1990-03-01  157. 1990 Mar 1990-03-01  20.4
## 4 1990-04-01  159. 1990 Apr 1990-04-01  18.4
## 5 1990-05-01  149. 1990 May 1990-05-01  18.2
## 6 1990-06-01  136. 1990 Jun 1990-06-01  16.7
```

4.3.3 Chapter 4 (Part A): Daily Crude Oil Futures Prices

Daily closing prices for commodities can be accessed using the API offered by *eodhistoricaldata.com*. A free API is available for end-of-day data with the following restrictions: a maximum of 20 API calls per day, and a maximum of one year's worth of daily data. A trial API allows users to retrieve multiple years of rolling crude oil futures prices. Let's use this trial API to download several years of daily crude oil futures prices.

Using the instructions from https://eodhistoricaldata.com, the trial API token and the symbol for crude oil futures are as follows:

```
# trial API for crude oil: http://eodhistoricaldata.com
api_token <- "OeAFFmMliFG5orCUuwAKQ8l4WWFQ67YX"
symbol <- "CL.COMM"
```

The following code pulls in daily opening, high, low and closing prices for WTI crude oil for the period January 1, 2017 to December 31, 2021.

```
ticker.link <- paste("http://eodhistoricaldata.com/api/eod/",
          symbol,"?from=2017-01-01&to=2021-12-31&period=d",
          "&api_token=",
          api_token)
ticker.link
```

```
## [1] "http://eodhistoricaldata.com/api/eod/ CL.COMM ?from=2017-01-01&to=2021-12-31&p
```

```
data <- read_csv(ticker.link, show_col_types = FALSE)
tail(data)
```

```
## # A tibble: 6 x 7
##   Date        Open  High   Low Close Adjusted_close Volume
##   <date>     <dbl> <dbl> <dbl> <dbl>          <dbl>  <dbl>
## 1 2021-12-23  72.6  73.6  71.8  73.4           73.4  95322
## 2 2021-12-27  73    75.7  72.2  75.2           75.2  76665
## 3 2021-12-28  75.6  76.5  75.1  75.6           75.6  66326
## 4 2021-12-29  75.7  77.0  75.0  76.2           76.2  78605
## 5 2021-12-30  76.2  77.1  75.4  76.6           76.6  61024
## 6 2021-12-31  76.1  76.7  74.6  74.9           74.9  63140
```

This API call looks rather complicated but it follows a standard set of API rules. The paste function concatenates three bits of code: the API URL, http://eodhistoricaldata.com/api/eod/, the parameters which indicate that daily data (d) is being requested, beginning on January 1, 2017 and ending on December 31, 2021, and lastly the API token. This string of concatenated characters is then placed into R's *read.csv* function and the data is retrieved.

It is useful to have a csv backup of the daily price data in case the previous API fails. If the backup data is to be used the data has format month-day-year rather than year-month day. This means the code below must be changed accordingly. Let's import this backup data now.

```
data_bk <- read.csv(here("data/ch3","CL.COMM.csv"))
```

To prepare the crude oil prices for formal analysis, it is first necessary to convert the date column from character format to R date format and then convert the data frame into a *tsibble* object.

```
crude_daily <- data_bk %>%
  mutate(Date = lubridate::ymd(as.Date(Date)))
head(crude_daily)
```

```
##         Date  Open  High   Low Close Adjusted_close Volume
## 1 0001-03-20 54.20 55.24 52.11 52.33          52.33 727793
## 2 0001-04-20 52.49 53.43 52.15 53.26          53.26 512641
## 3 0001-05-20 53.39 54.12 52.79 53.76          53.76 517362
## 4 0001-06-20 53.73 54.32 53.32 53.99          53.99 528333
## 5 0001-09-20 53.75 53.83 51.76 51.96          51.96 564893
## 6 0001-10-20 51.83 52.37 50.71 50.82          50.82 632573
```

4.3.4 Chapter 4 (Part B): CPI, Banana and Potato Prices

The monthly U.S. CPI and an index of the farm price of potatoes can be accessed using the API of the Federal Reserve of Economic Data (FRED).

Let's import the U.S. CPI from January 1990 to December 2021.

```
cpi_mnth <- fredr(
     series_id = "CPIAUCSL",
     observation_start = as.Date("1990-01-01"),
     observation_end = as.Date("2021-12-01")
   )
```

The FRED CPI data can be cleaned and converted into a tsibble object as follows:

```
cpi_mnth <- cpi_mnth %>%
  mutate(month = yearmonth(date)) %>%
  as_tsibble(index=month) %>%
  dplyr::select(month, value) %>%
  rename(CPI = value)
head(cpi_mnth)
```

```
## # A tsibble: 6 x 2 [1M]
##      month   CPI
##      <mth> <dbl>
## 1 1990 Jan  128.
## 2 1990 Feb  128
## 3 1990 Mar  129.
## 4 1990 Apr  129.
## 5 1990 May  129.
## 6 1990 Jun  130.
```

Now let's use the FRED API to read in the potato price index from January of 2000 to December of 2021.

```
ppi_potat <- fredr(
  series_id = "WPU01130603",
  observation_start = as.Date("2000-01-01"),
  observation_end = as.Date("2021-12-01"))
```

This potato PPI data can be cleaned in the same way that the CPI data was cleaned.

```
ppi_potat <- ppi_potat %>%
  mutate(month = yearmonth(date)) %>%
  as_tsibble(index=month) %>%
  dplyr::select(month, value) %>%
  rename(PPI = value)
head(ppi_potat)
```

```
## # A tsibble: 6 x 2 [1M]
##      month   PPI
##      <mth> <dbl>
## 1 2000 Jan  104.
## 2 2000 Feb  103.
## 3 2000 Mar  104.
## 4 2000 Apr  102.
## 5 2000 May  104
## 6 2000 Jun  103
```

The monthly global price of bananas is read in from an Excel file and converted to a *tsibble* object as follows.

```
bananas <- read_excel(here("data/ch3", "bananas.xlsx"),sheet = "bananas") %>%
  mutate(month = yearmonth(Month)) %>%
  dplyr::select(month,Price) %>%
  as_tsibble(index=month)
head(bananas)
```

```
## # A tsibble: 6 x 2 [1M]
##      month Price
##      <mth> <dbl>
## 1 2002 Sep  0.49
## 2 2002 Oct  0.54
## 3 2002 Nov  0.4
## 4 2002 Dec  0.63
## 5 2003 Jan  0.56
## 6 2003 Feb  0.46
```

4.3.5 Chapter 5: Industrial Production and Eggs

The empirical applications in Chapter 5 require four data sets. The first is a measure of monthly U.S. industrial production (electric and gas utilities), the second is the monthly U.S. CPI, the third is a data set which combines the monthly prices of wheat and crude oil and the fourth is the monthly producer price of eggs in the Province of Alberta. The combined wheat- crude oil data and the U.S. CPI were cleaned in the previous chapters and so they are excluded from this current round of cleaning. The industrial production data comes from the Federal Reserve of Economic Data (FRED) and the egg data comes from Agriculture and Agri-Food Canada.

We begin by reading in industrial production from FRED, filtering the dates from January of 1990 to December of 2021, reformatting the date column and coercing the data frame to a *tsibble* object.

```
industrial <- fredr(
  series_id = "IPUTIL",
  observation_start = as.Date("1990-01-01"),
  observation_end = as.Date("2021-12-01")) %>%
  dplyr::select(date,value)
head(industrial)
```

```
## # A tibble: 6 x 2
##   date       value
##   <date>     <dbl>
## 1 1990-01-01  71.4
## 2 1990-02-01  70.5
## 3 1990-03-01  71.5
## 4 1990-04-01  72.0
## 5 1990-05-01  72.4
## 6 1990-06-01  73.1
```

```
industrial <- industrial %>%
  mutate(month = yearmonth(date)) %>%
  as_tsibble(index=month) %>%
  dplyr::select(month, value) %>%
  rename(industrial = value)
head(industrial)
```

```
## # A tsibble: 6 x 2 [1M]
##      month industrial
##      <mth>      <dbl>
## 1 1990 Jan       71.4
## 2 1990 Feb       70.5
```

```
## 3 1990 Mar       71.5
## 4 1990 Apr       72.0
## 5 1990 May       72.4
## 6 1990 Jun       73.1
```

Reading and cleaning the producer price of jumbo eggs in the province of Alberta is more complex. The data for all provinces can be downloaded from the Statistics Canada website one year at a time. The data for each year is stored in a separate worksheet. The data runs from January of 1985 to December of 2021, which is 37 years. Let's use a loop procedure to read in each workbook and store it as a set of data frames with names *y1*, *y2*, etc.

```
for (i in 1:37) {

year <- eval(parse(text = "i"))

obj <- read_excel(here("data/ch3", "Canadian Monthly Egg Prices.xlsx"),
                  sheet = year, skip = 9,n_max = 12, na = "..",col_types = c("text", "numer
  clean_names() %>%
  rename(month = dollars_per_dozen,
         price = alberta) %>%
  dplyr::select("month","price")

assign(paste("y", year, sep=""),obj)
}
```

In this loop the variable *year*, which takes on values of *y1=1*, *y2=2*, etc as *i* moves through the loop, is used to identify the particular worksheet to read in. The data for a particular year is stored in a vector called *obj*. The last line of code assigns the data which is stored in the *obj* vector to a corresponding *yi* vector (e.g., *y1* for the 1985 worksheet, *y2* for the 1986 worksheet, etc.).

The next step is to vertically stack the 37 years of data into long format. There are eloquent ways to do this but to keep things simple a manual *rbind()* function is used. After the stacking a date vector is created to serve as labels for the respective egg prices.

```
price <- rbind(y1,y2,y3,y4,y5,y6,y7,y8,y9,y10,y11,y12,y13,y14,y15,y16,y17,y18,y19,y20,y

date <- seq(as.Date("1985/1/1"), by = "month", length.out = 444)
```

The last step is to merge the price and date vectors and then create a *tsibble* object out of the resulting data frame.

```
eggs <- tibble(price,date) %>%
  mutate(month = yearmonth(date),
         price = as.numeric(price)) %>%
  as_tsibble(index=month) %>%
  dplyr::select(price,month)
head(eggs)
```

```
## # A tsibble: 6 x 2 [1M]
##   price    month
##   <dbl>    <mth>
## 1  1.05 1985 Jan
## 2  1.04 1985 Feb
## 3  1.04 1985 Mar
## 4  1.03 1985 Apr
## 5  1.03 1985 May
## 6  1.02 1985 Jun
```

4.3.6 Chapter 6: Energy Consumption and Potato Prices

The analysis in this chapter uses four U.S. data sets:

- Monthly temperature anomaly (i.e., deviation from long term average)
- Monthly energy consumption
- Farm price of potatoes
- U.S. dollar and Great Britain pound exchange rate.

The first pair of data sets are merged to create a combined temperature and energy consumption data set. Similarly, the last pair of data sets are merged to create a combined potato price and exchange rate data set.

Temperature Anomaly Data

A temperature anomaly is the average temperature for the current month minus the long term average for that month. The temperature anomaly data resides in 12 worksheets of an Excel workbook, with a worksheet corresponding to each month. For example, the "Jan" worksheet contains the temperature anomaly data for each year from 1895 to 2022. Let's begin by importing the 12 data sets.

```
jan <- read_excel(here("data/ch3", "temperature.xlsx"),
                  sheet = "Jan", skip = 1, na = "..") %>%
  mutate(Month="01")

feb <- read_excel(here("data/ch3", "temperature.xlsx"),
                  sheet = "Feb", skip = 1, na = "..") %>%
```

```
  mutate(Month="02")

mar <- read_excel(here("data/ch3", "temperature.xlsx"),
                  sheet = "Mar", skip = 1, na = "..") %>%
  mutate(Month="03")

apr <- read_excel(here("data/ch3", "temperature.xlsx"),
                  sheet = "Apr", skip = 1, na = "..") %>%
  mutate(Month="04")

may <- read_excel(here("data/ch3", "temperature.xlsx"),
                  sheet = "May", skip = 1, na = "..") %>%
  mutate(Month="05")

jun <- read_excel(here("data/ch3", "temperature.xlsx"),
                  sheet = "Jun", skip = 1, na = "..") %>%
  mutate(Month="06")

jul <- read_excel(here("data/ch3", "temperature.xlsx"),
                  sheet = "Jul", skip = 1, na = "..") %>%
  mutate(Month="07")

aug <- read_excel(here("data/ch3", "temperature.xlsx"),
                  sheet = "Aug", skip = 1, na = "..") %>%
  mutate(Month="08")

sep <- read_excel(here("data/ch3", "temperature.xlsx"),
                  sheet = "Sep", skip = 1, na = "..") %>%
  mutate(Month="09")

oct <- read_excel(here("data/ch3", "temperature.xlsx"),
                  sheet = "Oct", skip = 1, na = "..") %>%
  mutate(Month="10")

nov <- read_excel(here("data/ch3", "temperature.xlsx"),
                  sheet = "Nov", skip = 1, na = "..") %>%
  mutate(Month="11")

dec <- read_excel(here("data/ch3", "temperature.xlsx"),
                  sheet = "Dec", skip = 1, na = "..") %>%
  mutate(Month="12")
```

The 12 data sets can now be combined using *year* as the common variable. The *arrange()* function is used to convert the wide data into the desired long data format (i.e., a format with the monthly temperature anomaly data stacked

rather than placed side-by-side).

```
temp <- rbind(jan,feb,mar,apr,may,jun,jul,aug,sep,oct,nov,dec) %>%
  rename(Year=Date) %>%
  mutate(Year=as.numeric(Year),
         Month=as.numeric(Month),
         Date=make_date(Year,Month),
         month=yearmonth(Date)
         )
temp2 <-arrange(temp, Year, Month) %>%
  filter(Year>=1986) %>%
  slice(-n())
head(temp2)
```

```
## # A tibble: 6 x 6
##    Year ClimDiv USCRN Month Date          month
##   <dbl>   <dbl> <dbl> <dbl> <date>        <mth>
## 1  1986    2.33 -100.     1 1986-01-01 1986 Jan
## 2  1986    0.2  -100.     2 1986-02-01 1986 Feb
## 3  1986    2.61 -100.     3 1986-03-01 1986 Mar
## 4  1986    1.01 -100.     4 1986-04-01 1986 Apr
## 5  1986    0.03 -100.     5 1986-05-01 1986 May
## 6  1986    0.91 -100.     6 1986-06-01 1986 Jun
```

Merge with Energy Consumption Data

Energy consumption data, which runs from January 1973 to April 2022, consists of monthly U.S. residential energy use. Let's read in this data from a csv file and then convert the imported data into a *tsibble* object.

```
ener_dat <- read.csv(here("data/ch3","energy.csv")) %>%
  mutate(month = yearmonth(date)) %>%
  as_tsibble(index = month) %>%
  dplyr::select(month,energy)
head(ener_dat)
```

```
## # A tsibble: 6 x 2 [1M]
##      month energy
##      <mth>  <dbl>
## 1 1973 Jan  1339.
## 2 1973 Feb  1175.
## 3 1973 Mar   983.
## 4 1973 Apr   715.
## 5 1973 May   536.
## 6 1973 Jun   367.
```

The energy consumption data and the temperature anomaly data can now be merged.

```
energy_temp <- ener_dat %>%
  inner_join(temp2,by="month") %>%
  dplyr::select(month,energy,ClimDiv) %>%
  rename(temp=ClimDiv) %>%
  slice(-(1:2))
head(energy_temp)
```

```
## # A tsibble: 6 x 3 [1M]
##      month energy  temp
##      <mth>  <dbl> <dbl>
## 1 1986 Mar   835.  2.61
## 2 1986 Apr   565.  1.01
## 3 1986 May   424.  0.03
## 4 1986 Jun   316.  0.91
## 5 1986 Jul   294. -0.9
## 6 1986 Aug   295. -1.35
```

The combined energy consumption and temperature anomaly data set is completed by adding a trend variable and monthly seasonal dummies.

```
n <- nrow(energy_temp)
energy_temp <- energy_temp %>% mutate(
  trend = 1:n,
  jan = as.integer(month(month) == 1),
  feb = as.integer(month(month) == 2),
  mar = as.integer(month(month) == 3),
  apr = as.integer(month(month) == 4),
  may = as.integer(month(month) == 5),
  jun = as.integer(month(month) == 6),
  jul = as.integer(month(month) == 7),
  aug = as.integer(month(month) == 8),
  sep = as.integer(month(month) == 9),
  oct = as.integer(month(month) == 10),
  nov = as.integer(month(month) == 11),
  dec = as.integer(month(month) == 12))
head(energy_temp)
```

```
## # A tsibble: 6 x 16 [1M]
##      month energy  temp trend   jan   feb   mar   apr   may   jun   jul   aug
##      <mth>  <dbl> <dbl> <int> <int> <int> <int> <int> <int> <int> <int> <int>
## 1 1986 Mar   835.  2.61     1     0     0     1     0     0     0     0     0
```

```
## 2 1986 Apr   565.  1.01     2     0     0     0     1     0     0     0     0
## 3 1986 May   424.  0.03     3     0     0     0     0     1     0     0     0
## 4 1986 Jun   316.  0.91     4     0     0     0     0     0     1     0     0
## 5 1986 Jul   294. -0.9      5     0     0     0     0     0     0     1     0
## 6 1986 Aug   295. -1.35     6     0     0     0     0     0     0     0     1
## # i 4 more variables: sep <int>, oct <int>, nov <int>, dec <int>
```

Potato and Exchange Rate Data

The data set for the second analysis in Chapter 6 consists of the producer price index (ppi) of U.S. potatoes and the U.S. - Britain exchange rate. Both of these data series can be accessed using the API of the Federal Reserve of Economic Data (FRED) - see above for details.

```
fredr_set_key("a63c5424469efa8009045b7d02da6a0f")

potato <- fredr(
  series_id = "WPU01130603",
  observation_start = as.Date("1992-01-01"),
  observation_end = as.Date("2022-08-01")) %>%
  rename(ppi = value) %>%
  mutate(month = yearmonth(date)) %>%
  dplyr::select(month,ppi)
head(potato)
```

```
## # A tibble: 6 x 2
##      month   ppi
##      <mth> <dbl>
## 1 1992 Jan  98.3
## 2 1992 Feb  93.9
## 3 1992 Mar  97
## 4 1992 Apr 123.
## 5 1992 May 116.
## 6 1992 Jun 132.
```

```
ex_uk <-  fredr(
  series_id = "EXUSUK",
  observation_start = as.Date("1992-01-01"),
  observation_end = as.Date("2022-08-01")) %>%
  rename(xchng = value) %>%
  mutate(month = yearmonth(date)) %>%
  dplyr::select(month,xchng)
head(ex_uk)
```

```
## # A tibble: 6 x 2
```

```
##      month xchng
##      <mth> <dbl>
## 1 1992 Jan  1.81
## 2 1992 Feb  1.78
## 3 1992 Mar  1.72
## 4 1992 Apr  1.76
## 5 1992 May  1.81
## 6 1992 Jun  1.86
```

The pair of data sets can now be merged into a combined potato price PPI and exchange rate data frame and then coerced into a *tsibble* object.

```
ppi <- potato %>% inner_join(ex_uk, by = "month") %>%
   as_tsibble(index=month)
head(ppi)
```

```
## # A tsibble: 6 x 3 [1M]
##      month   ppi xchng
##      <mth> <dbl> <dbl>
## 1 1992 Jan  98.3  1.81
## 2 1992 Feb  93.9  1.78
## 3 1992 Mar  97    1.72
## 4 1992 Apr 123.   1.76
## 5 1992 May 116.   1.81
## 6 1992 Jun 132.   1.86
```

This combined data set also requires a trend variable and monthly seasonal dummies.

```
ppi <- ppi %>% mutate(
  trend = 1:nrow(ex_uk),
  jan = as.integer(month(month) == 1),
  feb = as.integer(month(month) == 2),
  mar = as.integer(month(month) == 3),
  apr = as.integer(month(month) == 4),
  may = as.integer(month(month) == 5),
  jun = as.integer(month(month) == 6),
  jul = as.integer(month(month) == 7),
  aug = as.integer(month(month) == 8),
  sep = as.integer(month(month) == 9),
  oct = as.integer(month(month) == 10),
  nov = as.integer(month(month) == 11))
head(ppi)
```

```
## # A tsibble: 6 x 15 [1M]
##      month   ppi xchng trend   jan   feb   mar   apr   may   jun   jul   aug
##      <mth> <dbl> <dbl> <int> <int> <int> <int> <int> <int> <int> <int> <int>
## 1 1992 Jan  98.3  1.81     1     1     0     0     0     0     0     0     0
## 2 1992 Feb  93.9  1.78     2     0     1     0     0     0     0     0     0
## 3 1992 Mar  97    1.72     3     0     0     1     0     0     0     0     0
## 4 1992 Apr 123.   1.76     4     0     0     0     1     0     0     0     0
## 5 1992 May 116.   1.81     5     0     0     0     0     1     0     0     0
## 6 1992 Jun 132.   1.86     6     0     0     0     0     0     1     0     0
## # i 3 more variables: sep <int>, oct <int>, nov <int>
```

4.3.7 Chapter 7: Woodchips and Heating Oil

The wood-energy data set consists of the prices of wood chips, plywood, heating oil and crude oil. Similar to the import procedure used for potato prices in Chapter 6, the four prices in this data set are imported from the FRED website via an api (the data for crude oil was previously imported but it will be repeated here because the start data is earlier). Cleaning this data involves filtering the data to begin in January of 1986 and ending in April of 2022. As well, a descriptive name of each time series is used to replace "value" as the series name.

```
# import crude oil prices from FRED
crude_day <- fredr(
  series_id = "DCOILWTICO",
  observation_start = as.Date("1986-01-01"),
  observation_end = as.Date("2022-04-01")) %>%
  dplyr::select(date,value) %>%
  rename(crude_day = value)

# import chip prices from FRED
chips <- fredr(
  series_id = "PCU3211133211135",
  observation_start = as.Date("1986-01-01"),
  observation_end = as.Date("2022-04-01")) %>%
  dplyr::select(date,value) %>%
  rename(chips_lev = value,
         month = date)

# import plywood prices from FRED
plywood <- fredr(
  series_id = "WPU083",
  observation_start = as.Date("1986-01-01"),
  observation_end = as.Date("2022-04-01")) %>%
  dplyr::select(date,value) %>%
```

```
  rename(plywood_lev = value,
         month = date)

# import heating oil prices from FRED
heat <- fredr(
  series_id = "WPU05730201",
  observation_start = as.Date("1986-01-01"),
  observation_end = as.Date("2022-04-01")) %>%
  dplyr::select(date,value) %>%
  rename(heat_lev = value,
         month = date)
```

An additional cleaning step is required because the fourth data series (crude oil prices) arrives in daily format rather than monthly format. The daily data is aggregated into monthly average prices for crude oil as follows.

```
crude_day <- crude_day[complete.cases(crude_day),]
crude <- crude_day %>%
  mutate(month = yearmonth(date)) %>%
  group_by(month) %>%
  summarise(crude_lev = mean(crude_day))
```

The four imported prices for wood chips, plywood, heating oil and crude oil can now be merged and the resulting data frame can then be coerced into a *tsibble* time series object.

```
wood <- chips %>% inner_join(plywood, by = "month") %>%
  inner_join(heat, by = "month") %>%
  inner_join(crude, by = "month") %>%
  as_tsibble(index=month)
```

This wood - energy data set will be used in Chapter 7 to estimate a vector autoregression (VAR) model with price premium variables. The wood chip price premium is the log of the difference between the price of wood chips and the price of plywood (recall that the log of the price difference is approximately equal to the percent difference). The heating oil price premium is the log of the difference between the price of heating oil and the price of crude oil. Let's add these two price premium variables to our data set and then deselect the original price variables.

```
wood <- wood %>%
  mutate(premE=log(heat_lev-crude_lev),
         premW=log(plywood_lev-chips_lev)) %>%
  dplyr::select(month,premE,premW)
```

The price premium data requires a month variable for the purpose of creating seasonality variables. The month variable is added as follows.

```
wood <- wood %>%
  mutate(period = as.numeric(format(month, "%m")))
```

The final step is to merge the price premium data sets with the temperature anomaly data which was created in Chapter 6.

```
wood <- wood %>%
  mutate(temp=temp2$ClimDiv) %>%
  mutate(L.temp=lag(temp),
         L2.temp=lag(temp,2))
head(wood)
```

```
## # A tsibble: 6 x 7 [1D]
##   month      premE premW period  temp L.temp L2.temp
##   <date>     <dbl> <dbl>  <dbl> <dbl>  <dbl>   <dbl>
## 1 1986-01-01  4.10  2.58      1  2.33  NA      NA
## 2 1986-02-01  3.83  2.54      2  0.2    2.33   NA
## 3 1986-03-01  3.72  2.69      3  2.61   0.2     2.33
## 4 1986-04-01  3.63  2.97      4  1.01   2.61    0.2
## 5 1986-05-01  3.52  2.87      5  0.03   1.01    2.61
## 6 1986-06-01  3.48  2.77      6  0.91   0.03    1.01
```

4.3.8 Chapter 8: Vegetable Oils and Biofuels

This data set consists of the monthly prices of soybeans, rapeseed, palm oil. These are prices are extracted from a large World Bank data set. The monthly data runs from January 1960 to the current month. Importing the World Bank data into R is complicated for several reasons: (1) the column names reside in row 7; (2) the column names are in all capital letters; (3) there is no name for the date column; and (4) missing data is identified with ".." rather than "na".

The *clean_names()* function will convert capital letters to small letters in the column names, and will also assign "x1" as the name for the date column. Assigning *skip = 6* and *na = ".."* ensures that the first six rows are skipped when reading the data, and all occurrences of ".." are converted to "na". Let's import the data now.

Using these procedures the data can be imported as follows.

```
cmo <- read_excel(here("data/ch3", "CMO-Historical-Data-Monthly.xlsx"),
                  sheet = "Monthly Prices", skip = 6, na = "..") %>%
  clean_names()
```

```
## New names:
## * `` -> `...1`
```

The date format in the imported data is a little unusual (e.g., "1960M01"). Nevertheless, the *ym()* function from R's lubridate package is able to manage this conversion (note that the "x1" date column is renamed "month"). A filter function is used to select the rows which correspond to January of 2003 to December of 2021.

```
cmo <- cmo %>%
  rename(month = x1) %>%
  dplyr::select(month, palm_oil, soybean_oil, rapeseed_oil) %>%
  mutate(month = ym(month)) %>%
  filter(month >= "2003-01-01" & month <= "2020-12-01")
  head(cmo)
```

```
## # A tibble: 6 x 4
##   month      palm_oil soybean_oil rapeseed_oil
##   <date>        <dbl>       <dbl>        <dbl>
## 1 2003-01-01     486.        541.         624.
## 2 2003-02-01     477.        522.         586.
## 3 2003-03-01     454.        523.         552.
## 4 2003-04-01     443.        541.         557.
## 5 2003-05-01     454.        549.         607.
## 6 2003-06-01     466.        546.         612.
```

The final step is to reformat the month column to make it more readable. The reformatted month column can now serve as the index when coercing the imported data frame into a *tsibble* time series object.

```
cmo <- cmo %>% mutate(month = yearmonth(month)) %>%
  as_tsibble(index = month) %>%
  dplyr::select(month, everything())
  head(cmo)
```

```
## # A tsibble: 6 x 4 [1M]
##      month palm_oil soybean_oil rapeseed_oil
##      <mth>    <dbl>       <dbl>        <dbl>
## 1 2003 Jan     486.        541.         624.
## 2 2003 Feb     477.        522.         586.
## 3 2003 Mar     454.        523.         552.
## 4 2003 Apr     443.        541.         557.
## 5 2003 May     454.        549.         607.
## 6 2003 Jun     466.        546.         612.
```

The next data set to be created requires combining the weekly price of soybean oil and biodiesel, regular diesel and crude oil. The first pair of prices will be imported from an csv file and the second pair from individual Excel files. An inner join is used to merge these three data sets. The final data set runs from week 2 of 1994 to week 2 of 2022.

When reading in the soybean oil and biodiesel data, the first two rows are skipped and the first two columns, which contain the desired prices, are selected. The names of the two columns are then changed. Finally, the date which is imported in character format is converted to an R date format using the *mdy()* function. The procedure for reading in the price of diesel from one of the Excel files and the price of crude oil from the second Excel file is similar.

```
iowa_data <- read_csv(here("data/ch3", "hist_bio_gm.csv"), skip = 2) %>%
  dplyr::select(1:3) %>%
  dplyr::rename(date = 1, biodiesel = 2, soyoil = 3) %>%
  mutate(date = mdy(date)) # convert character to date format
```

```
## Rows: 775 Columns: 7
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): Date
## dbl (6): Biodiesel Price ($/gallon), Soybean Oil Price (cents/pound), Methan...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(iowa_data)
```

```
## # A tibble: 6 x 3
##   date       biodiesel soyoil
##   <date>         <dbl>  <dbl>
## 1 2007-04-13      3.1    29.9
## 2 2007-04-20      3.1    29.3
## 3 2007-04-27      3.08   30.2
## 4 2007-05-04      3.14   31.1
## 5 2007-05-11      3.14   31.1
## 6 2007-05-18      3.18   32.9
```

```
diesel_data <- readxl::read_excel(here("data/ch3","Diesel Fuel Prices-EIA.xls"), sheet = "Data 1"
  dplyr::select(1,2) %>%
  dplyr::rename(date = 1, diesel = 2) %>%
  mutate(date = ymd(date))
head(diesel_data)
```

```
## # A tibble: 6 x 2
##   date       diesel
##   <date>      <dbl>
## 1 1994-03-21   1.11
## 2 1994-03-28   1.11
## 3 1994-04-04   1.11
## 4 1994-04-11   1.11
## 5 1994-04-18   1.10
## 6 1994-04-25   1.11
```

```
crude_data <- readxl::read_excel(here("data/ch3","Crude Oil Prices-EIA.xls"), sheet = "
  dplyr::select(1,2) %>%
  dplyr::rename(date = 1, crude = 2) %>%
  mutate(date = ymd(date))
head(crude_data)
```

```
## # A tibble: 6 x 2
##   date       crude
##   <date>     <dbl>
## 1 1986-01-03  25.8
## 2 1986-01-10  26.0
## 3 1986-01-17  24.6
## 4 1986-01-24  20.3
## 5 1986-01-31  19.7
## 6 1986-02-07  16.7
```

The next step is to recognize that the dates for the soybean and biodiesel data do not perfectly match with the dates for the diesel and crude oil data (e.g., prices within the first set are measured on a Friday, and prices for the second set are measured on a Monday). It is useful to use the *yearweek()* function from the *tsibble* package to assign a common year-week ID for both sets of data. After creating the new *week* variable, the original *date* variable is deselected so that it is no longer part of the tibble.

```
iowa_data <- iowa_data %>%
  mutate(week = yearweek(date)) %>%
  as_tibble(index=week) %>%
  dplyr::select(-date)

diesel_data <- diesel_data %>%
  mutate(week = yearweek(date)) %>%
  as_tibble(index=week) %>%
  dplyr::select(-date)

crude_data <- crude_data %>%
```

```
  mutate(week = yearweek(date)) %>%
  as_tibble(index=week) %>%
  dplyr::select(-date)
```

With the dates aligned across the three data sets, it is possible to merge the two data sets. The joins are done in pairs and so in the first join the diesel and crude oil prices are merged, and in the second join the soybean oil and biodiesel prices are merged with the diesel and crude oil prices. Note that the *inner_join()* function from the *dplyr* package includes all rows which are present in both sets of data. After the join the *date* column is moved to the right of the last column. The merged data is then converted into a tibble and the *distinct()* function from the *dplyr* package is used to eliminate duplicate rows. After moving the *week* column to the left of the first column, the tibble is converted into a time series *tsibble*.

```
diesel_crude <- inner_join(diesel_data, crude_data,by="week")
data <- inner_join(iowa_data,diesel_crude,by="week") %>%
  relocate(week) %>%
  as_tibble(index=week) %>%
  distinct() %>%
  as_tsibble(index = week)
head(data)
```

```
## # A tsibble: 6 x 5 [1W]
##       week biodiesel soyoil diesel crude
##     <week>     <dbl>  <dbl>  <dbl> <dbl>
## 1 2007 W15      3.1    29.9   2.84  62.6
## 2 2007 W16      3.1    29.3   2.88  63.1
## 3 2007 W17      3.08   30.2   2.85  65.3
## 4 2007 W18      3.14   31.1   2.81  63.8
## 5 2007 W19      3.14   31.1   2.79  61.9
## 6 2007 W20      3.18   32.9   2.77  63.6
```

When working with weekly time series data it is common for the data for some weeks to be missing. With the data residing in tsibble object, the *check_gaps()* function can be used to identify the gaps. Note that when working with *tsibble* objects the *forecast* package must be loaded for the *check_gaps()* function to work:

```
data_gaps <- data %>%
  count_gaps(.full = TRUE)
```

```
## Warning: Returning more (or less) than 1 row per `summarise()` group was deprecated in
```

```
## dplyr 1.1.0.
## i Please use `reframe()` instead.
## i When switching from `summarise()` to `reframe()`, remember that `reframe()`
##   always returns an ungrouped data frame and adjust accordingly.
## i The deprecated feature was likely used in the tsibble package.
##   Please report the issue at <https://github.com/tidyverts/tsibble/issues>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
data_gaps
```

```
## # A tibble: 3 x 3
##      .from      .to    .n
##     <week>   <week> <int>
## 1 2013 W40 2013 W42     3
## 2 2016 W12 2016 W12     1
## 3 2017 W21 2017 W21     1
```

We can see that data is missing for three consecutive weeks in 2013, one week in 2016 and one week in 2017. Data can be imputed for these missing weeks using the *fill_gaps()* function:

```
biofuel <- data %>%
  fill_gaps(.full = TRUE) %>%
  mutate(
    biodiesel = na.interp(biodiesel),
    soyoil = na.interp(soyoil),
    diesel = na.interp(diesel),
    crude = na.interp(crude))
 head(biofuel)
```

```
## # A tsibble: 6 x 5 [1W]
##       week biodiesel soyoil diesel crude
##     <week>     <dbl>  <dbl>  <dbl> <dbl>
## 1 2007 W15      3.1    29.9   2.84  62.6
## 2 2007 W16      3.1    29.3   2.88  63.1
## 3 2007 W17      3.08   30.2   2.85  65.3
## 4 2007 W18      3.14   31.1   2.81  63.8
## 5 2007 W19      3.14   31.1   2.79  61.9
## 6 2007 W20      3.18   32.9   2.77  63.6
```

4.3.9 Chapter 9: Regional Crude Oil and Vegetable Prices

There are two data sets for this chapter. The first consists of the monthly prices of West Texas Intermediate (wti) and Western Canada Select (wcs) crude oil plus the U.S. dollar index. The second data set consists of the monthly U.S. price of diesel fuel and the monthly U.S. consumer price index (CPI) for vegetables.

Crude Oil

The pair of oil prices are read in from an Excel file. The usual cleaning procedure is performed, which consists of converting the date to a month format and then using this converted date as the index for a *tsibble* time series object. The data is filtered to begin in January of 2005, which is the first month the wcs data is available.

```
col_names <- read_excel(here("data/ch3", "wti_wcs_price.xlsx"),sheet = "amCharts", n_max = 0) |>

oil <- read_excel(here("data/ch3", "wti_wcs_price.xlsx"),sheet = "amCharts", skip = 262, col_name
```

```
## New names:
## * `` -> `...1`
## * `` -> `...2`
## * `` -> `...3`
## * `` -> `...4`
## * `` -> `...5`
```

```
names(oil) <- col_names

oil <- oil %>%
  mutate(month = yearmonth(date)) %>%
  as_tsibble(index=month) %>%
  dplyr::select(month,wti,wcs)

# oil <- read_excel(here("data/ch3", "wti_wcs_price.xlsx"),
#       sheet = "amCharts", col_types = c("date", "skip", "numeric", "numeric","numeric")) %>%
#   mutate(month = yearmonth(date)) %>%
#   as_tsibble(index=month) %>%
#   dplyr::select(month,wti,wcs) %>%
#   filter_index("2005 Jan" ~.)
# head(oil)
```

To complete the first data set the U.S. dollar index is imported through use of a FRED api:

```
# import crude oil prices from FRED
dollar <- fredr(
  series_id = "TWEXBGSMTH",
  observation_start = as.Date("2006-01-01"),
  observation_end = as.Date("2022-08-01")) %>%
  rename(dollar = value) %>%
  mutate(month = yearmonth(date)) %>%
  dplyr::select(month,dollar)
head(dollar)
```

```
## # A tibble: 6 x 2
##      month dollar
##      <mth>  <dbl>
## 1 2006 Jan  100
## 2 2006 Feb  100.
## 3 2006 Mar  100.
## 4 2006 Apr   99.7
## 5 2006 May   97.5
## 6 2006 Jun   98.7
```

The dollar index and the oil price data can now be merged.

```
oil <- oil %>% inner_join(dollar,by="month")
head(oil)
```

```
## # A tsibble: 6 x 4 [1M]
##      month wti   wcs   dollar
##      <mth> <chr> <chr>  <dbl>
## 1 2006 Jan 65.54 38.80  100
## 2 2006 Feb 61.93 28.68  100.
## 3 2006 Mar 62.97 36.77  100.
## 4 2006 Apr 70.16 52.02   99.7
## 5 2006 May 70.96 57.29   97.5
## 6 2006 Jun 70.97 51.63   98.7
```

Vegetable CPI

The second data set is constructed by importing the U.S. price of diesel fuel and the vegetable consumer price index (cpi) from the FRED website:

```
veggie_df <- fredr(
  series_id = "CUSR0000SAF113",
  observation_start = as.Date("1996-04-01"),
  observation_end = as.Date("2022-04-01")) %>%
```

```
  rename(veggies = value) %>%
  mutate(month = yearmonth(date)) %>%
  dplyr::select(month,veggies)

head(veggie_df)
```

```
## # A tibble: 6 x 2
##      month veggies
##      <mth>   <dbl>
## 1 1996 Apr    186.
## 2 1996 May    183.
## 3 1996 Jun    186.
## 4 1996 Jul    185.
## 5 1996 Aug    184.
## 6 1996 Sep    184.
```

```
diesel_wk <- fredr(
  series_id = "GASDESW",
  observation_start = as.Date("1996-04-01"),
  observation_end = as.Date("2022-05-01")) %>%
  dplyr::select(date,value) %>%
  rename(diesel_wk = value) %>%
  as_tsibble(index=date)
```

The diesel fuel price is weekly and so it must be aggregated and logged to create a monthly logged data series.

```
diesel_mth <-diesel_wk %>%
  index_by(month = ~ yearmonth(.)) %>%
  summarise(
    diesel_raw = mean(diesel_wk)) %>%
  mutate(diesel = log(diesel_raw))
head(diesel_mth)
```

```
## # A tsibble: 6 x 3 [1M]
##      month diesel_raw diesel
##      <mth>      <dbl>  <dbl>
## 1 1996 Apr       1.27  0.241
## 2 1996 May       1.28  0.244
## 3 1996 Jun       1.21  0.188
## 4 1996 Jul       1.18  0.163
## 5 1996 Aug       1.20  0.183
## 6 1996 Sep       1.26  0.235
```

The monthly diesel fuel and veggie cpi data series can now be merged.

```
diesel_cpi <- diesel_mth %>%
 inner_join(veggie_df,by="month") %>%
  dplyr::select(month,veggies,diesel)
  head(diesel_cpi)
```

```
## # A tsibble: 6 x 3 [1M]
##      month veggies diesel
##      <mth>   <dbl>  <dbl>
## 1 1996 Apr    186.  0.241
## 2 1996 May    183.  0.244
## 3 1996 Jun    186.  0.188
## 4 1996 Jul    185.  0.163
## 5 1996 Aug    184.  0.183
## 6 1996 Sep    184.  0.235
```

Add to this data frame the monthly price of wti (U.S. crude oil).

```
diesel_cpi <- diesel_cpi %>%
 inner_join(oil,by="month") %>%
  dplyr::select(month,veggies,diesel,wti)
  head(diesel_cpi)
```

```
## # A tsibble: 6 x 4 [1M]
##      month veggies diesel wti
##      <mth>   <dbl>  <dbl> <chr>
## 1 2006 Jan    251.  0.903 65.54
## 2 2006 Feb    251.  0.906 61.93
## 3 2006 Mar    250.  0.939 62.97
## 4 2006 Apr    248.  1.00  70.16
## 5 2006 May    247.  1.06  70.96
## 6 2006 Jun    251   1.06  70.97
```

4.4 Saved RDS files

This section consists of the code for saving the data set for each chapter into a *.RDS* file. At the beginning of the empirical application in each chapter, the corresponding *.RDS* file is imported in order to make the clean data available. The lines of code are currently commented out. The code should be made live and the save initiated if a change is made to one or more of the data sets.

```
# saveRDS(wht_oil, here("data/ch3", "wht_oil.RDS"))
# saveRDS(crude_daily, here("data/ch3", "crude_daily.RDS"))
# saveRDS(cpi_mnth, here("data/ch3", "cpi_mnth.RDS"))
# saveRDS(bananas, here("data/ch3", "bananas.RDS"))
# saveRDS(ppi_potat, here("data/ch3", "ppi_potat.RDS"))
# saveRDS(industrial, here("data/ch3", "industrial.RDS"))
# saveRDS(eggs, here("data/ch3", "eggs.RDS"))
# saveRDS(cmo, here("data/ch3", "vegoils.RDS"))
# saveRDS(energy_temp, here("data/ch3", "energy_temp.RDS"))
# saveRDS(ppi, here("data/ch3", "potatoes.RDS"))
# saveRDS(wood, here("data/ch3", "wood.RDS"))
# saveRDS(biofuel, here("data/ch3", "biofuel.RDS"))
# saveRDS(oil, here("data/ch3", "oil.RDS"))
# saveRDS(diesel_cpi, here("data/ch3", "diesel_cpi.RDS"))
```

4.4.1 Extra: Lumber Data

This last data set on lumber prices is currently not being used in any of the empirical analysis. It consists of the monthly prices of four types of U.S. lumber, one type of Swedish lumber and the U.S. - Sweden exchange rate. The data runs from January of 1997 to December of 2013. The U.S. prices are in a csv file, the Swedish price is in an Excel file and the exchange rate is in a csv file.

The dates in the U.S. data are into two separate columns (i.e., month with an abbreviated label in one column and year in a second column). The *match()* function matches the month abbreviation to a standard full month label. The *paste()* function is then used to paste the month and year labels together to create a new column which contains a month-year combo label. The *ym()* function can be applied to this new column to create the R date. The original month and year columns can then be deselected.

```
lumbUSA <- read_csv(here("data/ch3", "Lumber Assortment USA.csv"))
```

```
## Rows: 204 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): month
## dbl (4): year, df_2x4, west_ply, strd_brd
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
lumbUSA <- lumbUSA %>%
  mutate(date = ym(paste(year, match(month,month.abb), sep="-"))) %>%
  select(!c(year,month))
```

Cleaning the Swedish lumber price data is relatively complex because the Excel file is organized to be used with an Excel search macro. It is best to assign no column names and to assume that all incoming data is in text format. If this latter option is not used then R will generate warnings due to a failure of the auto-detect function.

After the Swedish data is read in, the month and year columns, and the column which contains the price of the selected Swedish wood product are selected. As well, the data is sliced so that only those rows which correspond to the U.S. data are retained in the tibble. The cleaning of the Swedish table concludes by converting the prices from text to numeric and converting the month and year columns into a single R date. Also, the original month and year columns are deselected, similar to what was done for the U.S. lumber data.

```
lumbSWD <- read.csv(here("data/ch3","PriceOutputTable.csv")) %>%
  mutate(date = ym(paste(year, match(month,month.abb), sep="-")))
# lumbSWD <- suppressMessages(readxl::read_excel(here("data/ch3","PriceOutputTable.xls
#   dplyr::select(3,4,16) %>%
#   rename(year = 1, month = 2, sweden = 3) %>%
#   slice(462:749) %>%
#   mutate(sweden=as.numeric(sweden)) %>%
#   mutate(date = ym(paste(year, match(month,month.abb), sep="-"))) %>%
#   dplyr::select(!c(year,month))
# head(lumbSWD)
```

Reading in the exchange rate data is straight forward because there are only two columns and the date column has a standard format:

```
exchange <- read_csv(here("data/ch3", "EXSDUS.csv")) %>%
  mutate(date = mdy(DATE)) %>%
  select(!DATE)
```

```
## Rows: 614 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): DATE
## dbl (1): EXSDUS
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

We can now merge the cleaned U.S. and Swedish data, and then merge the resulting tibble with the exchange rate data. The last step is to convert the resulting tibble to a time series tsibble object.

```
lumber_both <- merge(lumbUSA,lumbSWD,by="date")
lumber <- merge(lumber_both,exchange,by="date") %>%
as_tsibble(index = date)
head(lumber)
```

```
## # A tsibble: 6 x 8 [1D]
##   date       df_2x4 west_ply strd_brd  year month Sweden EXSDUS
##   <date>      <dbl>    <dbl>    <dbl> <int> <chr>  <dbl>  <dbl>
## 1 1997-01-01    356      294      242  1997 Jan     1630   7.07
## 2 1997-02-01    371      315      249  1997 Feb     1774   7.41
## 3 1997-03-01    364      319      239  1997 Mar     1753   7.65
## 4 1997-04-01    384      303      231  1997 Apr     1835   7.69
## 5 1997-05-01    383      314      230  1997 May     1855   7.69
## 6 1997-06-01    378      343      235  1997 Jun     1789   7.75
```

4.5 Conclusions

In this chapter a variety of methods were used to import and clean data. The easiest cleaning is for the data imported by API from the FRED website. This is because the data is standardized and so the same cleaning steps can be used for each data set. Importing data from Excel files and csv files are similar regarding cleaning steps. Excel has the advantage that a "source" worksheet can be used to describe where and when the data was accessed. It is important to identify when the data was accessed from a website so that different versions of the data can be effectively managed.

When accessing data from a website it is tempting to do some of the initial cleaning steps in Excel. For example, it may be tempting to delete a blank row between the header and the first row of data, or perhaps merge two data sets before reading the data into R. Resist this temptation because it moves the analysis further from reproducibility. The goal is to make it as easy as possible for another researcher to be able to reproduce your results. It is for this reason why it is preferable to import data using API rather than downloading the data to a csv or Excel file and then reading the data into R.

Chapter 5

Stochastic Processes

In the field of applied economics and business, time series analysis broadly consists of two general areas: (1) forecasting and decomposition, which has strong overlap with data science; and (2) time series econometrics, which is used for estimating dynamic relationships between key economic and non-economic variables. Included in this latter category are vector autotregression (VAR) models for simulating policy-driven impulse response, cointegration models for measuring long run economic relationships, vector error correction models (VECM) for untangling the short and long run price dynamics and GARCH models for measuring volatility spillover impacts. Models such as the VAR are used extensively for both forecasting and time series econometrics.

Forecasting and time series analysis requires assumptions to be made about the underlying stochastic processes. For example, is the variable in question stationary over time or does it follow a non-stationary random walk, typically with a stochastic trend? How many lags of the variable in question should be included as explanatory variables in a regression model? Should a time series regression model be estimated with an autoregressive integrated moving average (ARIMA) error term in order to simultaneously ensure that the included variables are stationary and free of autocorrelation? To what extent should the data be aggregated as way to reduce autocorrelation? Finally, how should seasonality in the data be modeled?

This chapter provides an introductory treatment of the stochastic processes which underlie time series analysis. The emphasis is on three key components of stochastic processes: (1) possible non-stationarity of the time series due to stochastic trends; (2) possible non-stationarity of the time series due to seasonality; and (3) autocorrelation of the time series. Gaining a solid understanding of the stochastic processes which are emphasized in this chapter will make it much easier to understand the empirical applications in the chapters to follow. Fortunately R has a number of highly useful functions which are well suited for analyzing the underlying stochastic processes of time series variables.

Four data sets are used in this chapter to highlight our examination of stochastic processes: the global price of bananas, the U.S. consumer price index (CPI), the price of West Texas Intermediate (WTI) crude oil and a U.S. potato producer price index (PPI). These data sets were previously cleaned and stored in *RDS* files – see Chapter 3. Section 5.1 consists of a brief introduction to stochastic processes. The focus of Section 5.2 is stochastic trends as a cause of non-stationarity. Section 5.3 examines autocorrelation and Section 5.4 addresses seasonality. Summary comments are provided in Section 5.5.

5.1 Introduction

Modeling the stochastic processes which underlie time series analysis varies widely in terms of breadth and depth. Examples of complex topics which are not included in this textbook are state space analysis, spectral analysis and filtering, and various non-linear time series applications. A textbook which is most relevant for this current analysis of food and energy prices is Mills and Markellos [2008]. This graduate level textbook focuses on the econometric modelling of financial time series. The most relevant chapters in Mills and Markellos [2008] are:

- Chapter 2: Univariate linear stochastic models: basic concepts
- Chapter 3: Univariate linear stochastic models: testing for unit roots and alternative trend specifications
- Chapter 5: Univariate non-linear stochastic models: martingales, random walks and modelling volatility.

It is useful to remind readers why understanding the stochastic properties of time series data is important, even if the data is being used in a non-time series application. Suppose the goal is to use annual time series data on fertilizer use and crop yields in order to estimate the yield response to fertilizer. Suppose further that both crop yields and fertilizer use are trending up over time. It is possible that the trend in these two variables is entirely causal (i.e., higher fertilizer use is driving higher crop yields) but it is also possible that the trend is partially spurious. The relationship would be partially spurious if crop yields are increasing due to improved genetics and fertilizer use is increasing due to a decreasing real price of fertilizer over time. In general, analyzing the underlying stochastic properties of time series variables is required to determine if co-movements are spurious.

One of the most basic concepts in the assessment of stochastic processes is stationarity. Stationarity is examined in detail in Chapter 5. For now it is sufficient to view a stochastic process y_t as stationary if it has a mean, y^*, and variance, σ^2 which do not change over time. Suppose $y_t = 5 + 0.7y_{t-1} + \epsilon_t$ where ϵ_t is a white noise random error term. To calculate the mean and thus show that

this is a stationary series, assume $\epsilon_t = 0$ for all future time periods. The long run mean value of y_t satisfies $y_t = y_{t-1} = y^*$. Using the previous equation we have $y^* = 5 + 0.5y^*$. Solve this equation to obtain the constant mean $y^* = 10$,

In contrast, $y_t = 5 + y_{t-1} + \epsilon_t$ is non-stationary because it does not have a constant mean and variance. Indeed, using the procedure from the previous paragraph it should be clear that $y_t = 5 + y_{t-1} + \epsilon_t$ cannot be solved for y^*. This non-stationary stochastic process is known as a random walk with drift. For reasons which will be explained later, a random walk is equivalent to a stochastic process with a unit root. A theme which will be repeated many times in subsequent chapters is that non stationary time series variables should not be used in a regression (there is one important exception which is the topic of Chapter 8). Non stationary data should not be used because variables with unit roots are subject to stochastic trends, and these stochastic trends can result in spurious regression outcomes.

A second key concept in our study of stochastic processes is trend stationary versus difference stationary. Returning to our previous example, suppose $y_t = 5 + 0.7y_{t-1} + 0.25T + \epsilon_t$ where T is a time trend variable. This current specification is not stationary because the mean value is increasing linearly with the time trend. However, this stochastic process can be made stationary simply by removing the time trend through regression or an equivalent procedure. For this reason $y_t = 5 + 0.7y_{t-1} + 0.25T + \epsilon_t$ is considered to be a trend stationary stochastic process. In contrast, $y_t = 5 + y_{t-1} + \epsilon_t$ is a difference stationary stochastic process because the current specification is non stationary but the first difference, $y_t - y_{t-1} = 5 + \epsilon_t$, is stationary since it has has a fixed mean equal to 5. Determining whether a stochastic process is trend stationary or difference stationary plays an important role in the empirical applications to follow.

This textbook focuses on food and energy commodity prices and as such it is useful to ask if the prices of these types of commodities are stationary or non-stationary in general. Publicly traded stocks such as Exxon Mobil are usually non-stationary since reversion to a fixed mean would be eliminated through profit seeking. Nominal commodity prices are also usually non stationary when measured over a sufficiently long time horizon due to the effect of inflation. Real commodity prices are usually a mix of stationary and non stationary. This is because commodities have a real cost of production and if supply is relatively elastic this production cost defines the long run mean price of the commodity.

A third important concept in our study of stochastic processes is autocorrelation. Suppose we have determined that two daily time series variables (e.g., the futures prices for oil and wheat) are each stationary after removing a deterministic trend. Because these variables are stationary it is acceptable to regress the first variable on the second variable. If we do this we will almost certainly find that the regression residual has a high degree of autocorrelation (if we used monthly data instead of daily data the autocorrelation will be lower). Autocorrelation does not systematically bias our coefficient estimates but it does bias

downward the estimated standard errors. This bias generally means that the t statistic and p values will over state the statistical significance of the estimated coefficients.

The standard method to eliminate autocorrelation from the regression residual is to include lagged values of the variable on the right side of the regression equation. However, choosing the correct number of lags is tricky because too few lags will not eliminate the autocorrelation to an acceptable level and too many lags will inflate the standard errors of the coefficient estimates. Methods have been developed to use statistical measures such at the Akaike information criterion to optimally choose the number of lags. A general approach to ensuring the time series is both stationary and free of autocorrelation is obtained by estimating an autoregressive integrated moving average (ARIMA) model. In Chapter 6 a regression model with ARIMA errors is used to estimate the statistical relationship between energy and food prices.

The last important concept in our study of stochastic processes is seasonality. Seasonality is a specific form off non-stationarity and as such it should be removed before time series variables are used in regressions. In non-time series econometric models it is common to include seasonal dummy variables as a way to control for seasonality. In time series models the preferred approach is to use seasonal differencing to control for seasonal non-stationarity. An example of a seasonal difference is using the difference between the month j price of year t and the month j price of year $t-1$ for $j = 1...12$ in the regression equation. A seasonal ARIMA model adds seasonal differences to a standard ARIMA model.

In the next section four time series data sets are visually analyzed to assess the status of trends and seasonality (autocorrelation is generally difficult to identify through visual inspection). Beginning at the time of the Great Recession, nominal price of bananas appears to be trend stationary and the real price appears to be stationary. When working with logged prices, using the consumer price index (CPI) to deflate the price of a commodity such as bananas is equivalent to removing a linear trend if the CPI itself is trend stationary. A visual plot of the CPI is suggestive of a trend stationary time series but a visual plot of the detrended CPI makes this assessment much less certain. A formal test of stationarity of the CPI is deferred to Chapter 5.

A plot of the price of crude oil reveals a high likelihood of non-stationary due to a stochastic trend. Not surprisingly, there are no obvious signs of seasonality in the price of crude oil. In contrast, the producer price index for potatoes illustrates a time series with seasonality and no obvious deterministic trend.

Before moving to the next section, it is useful to first load the R packages which will be used in this section. As noted in Chapter 3, the *tidyverse* and *lubridate* packages facilitate "tidy" coding. The *tsibble* and *feasts* packages are specialized time series packages. The *gridExtra* package allows for side-by-side plots and the *fredr* package is for used for importing the Federal Reserve Economic Data via an API.

```
pacman::p_load(tidyverse, here, lubridate, tsibble, feasts, gridExtra, fredr, reshape2)
```

5.2 Trends

In this section the four U.S. data series (bananas, the CPI, crude oil and the farm price of potatoes) are visually examined for obvious signs of trends and seasonality. These four data sets were cleaned in Chapter 3 and stored in four corresponding *.RDS* files as *tsibble* objects. Let's read in these RDS files now.

```
bananas <- readRDS(here("data/ch4","bananas.RDS"))
head(bananas)
```

```
## # A tsibble: 6 x 2 [1M]
##      month Price
##      <mth> <dbl>
## 1 2002 Sep  0.49
## 2 2002 Oct  0.54
## 3 2002 Nov  0.4
## 4 2002 Dec  0.63
## 5 2003 Jan  0.56
## 6 2003 Feb  0.46
```

```
cpi_mnth <- readRDS(here("data/ch4","cpi_mnth.RDS"))
head(cpi_mnth)
```

```
## # A tsibble: 6 x 2 [1M]
##      month   CPI
##      <mth> <dbl>
## 1 1990 Jan  128.
## 2 1990 Feb  128
## 3 1990 Mar  129.
## 4 1990 Apr  129.
## 5 1990 May  129.
## 6 1990 Jun  130.
```

```
crude_daily <- readRDS(here("data/ch4","crude_daily.RDS"))
head(crude_daily)
```

```
## # A tsibble: 6 x 7 [1D]
##   Date        Open  High   Low Close Adjusted_close Volume
##   <date>     <dbl> <dbl> <dbl> <dbl>          <dbl>  <dbl>
```

```
## 1 2017-01-03  54.2  55.2  52.1  52.3          52.3 727793
## 2 2017-01-04  52.5  53.4  52.2  53.3          53.3 512641
## 3 2017-01-05  53.4  54.1  52.8  53.8          53.8 517362
## 4 2017-01-06  53.7  54.3  53.3  54.0          54.0 528333
## 5 2017-01-09  53.8  53.8  51.8  52.0          52.0 564893
## 6 2017-01-10  51.8  52.4  50.7  50.8          50.8 632573
```

```
ppi_potat <- readRDS(here("data/ch4","ppi_potat.RDS"))
head(ppi_potat)
```

```
## # A tsibble: 6 x 2 [1M]
##      month   PPI
##      <mth> <dbl>
## 1 2000 Jan  104.
## 2 2000 Feb  103.
## 3 2000 Mar  104.
## 4 2000 Apr  102.
## 5 2000 May  104
## 6 2000 Jun  103
```

Before plotting the various data series it is useful to note that for the purpose of statistical analysis prices tend to enter in logged format. This is because a price distribution generally has a long right tail and so the log of the price series is usually much closer to having a normal distribution in comparison to the price series itself. If the CPI is used as a deflator when creating a real price series we have $P_t^{real} = 100P_t/CPI_t$. The log of the real price series is given by $ln(P_t^{real}) = ln(100) + ln(P_t) - ln(CPI_t)$. This equation shows that when working with logs, deflating a price series with the CPI is similar to removing a linear trend from a price series if the CPI itself is following a linear trend (if the CPI is trend stationary then the log of CPI will also be trend stationary). This correspondence between deflation and detrending is revisited in the next section.

5.2.1 Nominal and Real Banana Prices

Our goal for this section is to plot the log of the nominal price and the log of the real monthly price of bananas in order to make a visual assessment of stationarity. To do this we must add the CPI to the banana data frame and then use it to deflate the nominal banana price. Taking logs of the nominal and real price completes the setup. Let's keep things simple by assigning the first observation (September of 2002) as the base year for the real price of bananas.

```
base <- cpi_mnth$CPI[1] # set the base year as the first observation

bananas <- bananas %>% inner_join(cpi_mnth, by = "month") %>%
  mutate(lnPrice = log(Price),
         lnPreal = log(Price/CPI*base)) %>%
  dplyr::select(month,lnPrice,lnPreal)
head(bananas)
```

```
## # A tsibble: 6 x 3 [1M]
##      month lnPrice lnPreal
##      <mth>   <dbl>   <dbl>
## 1 2002 Sep  -0.713  -1.06
## 2 2002 Oct  -0.616  -0.968
## 3 2002 Nov  -0.916  -1.27
## 4 2002 Dec  -0.462  -0.817
## 5 2003 Jan  -0.580  -0.939
## 6 2003 Feb  -0.777  -1.14
```

To plot both schedules on the same graph with a legend it is useful to use the *melt()* function from the *reshape2* package.

```
bananas_melt <- reshape2::melt(bananas, id.var='month')

ggplot(bananas_melt, aes(x=month, y=value, col=variable)) + geom_line()
```

Figure 5.1 shows that similar to the price of most other commodities the price of bananas surged in the early to mid 2000s. Interestingly, the price of bananas experienced only a brief crash during the 2008 - 2009 Great Recession. Since this time the nominal price of bananas increased steadily at the general rate of inflation whereas the real price of bananas has remained relatively flat. Based on a visual assessment alone, it would appear that since 2008 - 2009 the nominal price of bananas is trend stationary and the real price is stationary - possibly with a slight trend. There is no visual evidence of stochastic trends in either data series.

Let's plot the monthly CPI beginning in the year 2000 in order to visually assess the stationarity properties of this time series. We will use the *autoplot()* function from the *ggplot2* package to create the plot. Note that the *autoplot()* function is part of the *ggplot2* package but when working with *tsibble* objects it is necessary to have the *feasts* package loaded in order to use *autoplot()*.

```
n <- nrow(cpi_mnth)
autoplot(cpi_mnth[120:n,], CPI) +
  labs(title = "Monthly U.S. CPI: All Urban Consumers",
       subtitle = "Index 1982-1984=100", x="")
```

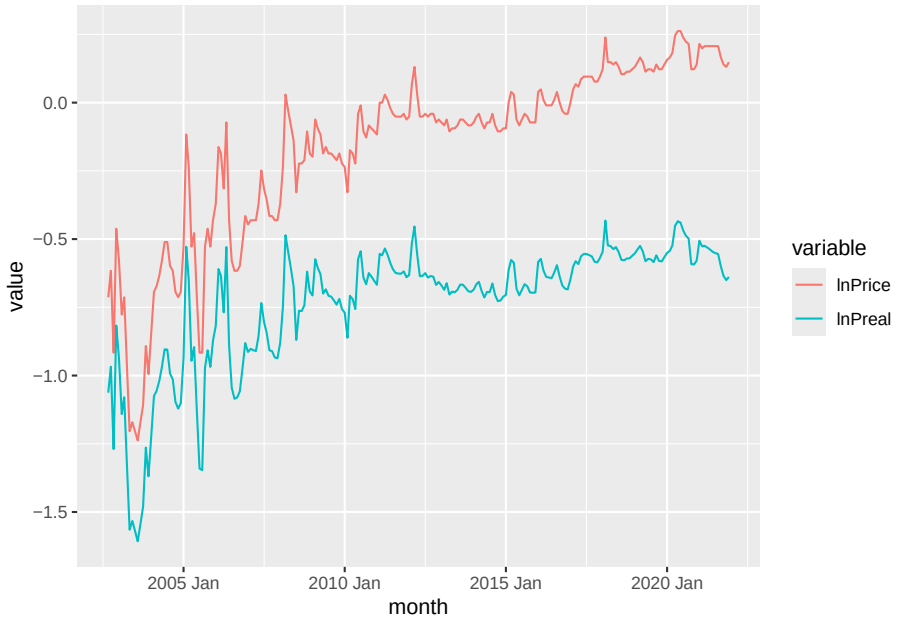

Figure 5.1: Monthly Nominal and Real Global Price of Bananas

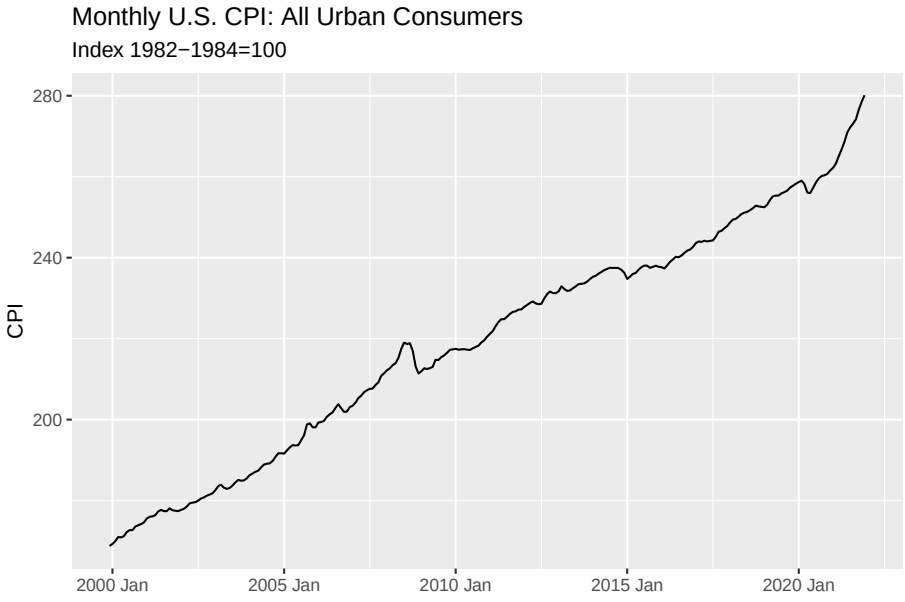

Figure 5.2: Monthly U.S. Consumer Price Index (CPI)

Figure 5.2 reveals that the trend in the CPI is approximately linear. This outcome is expected because the CPI is essentially a measure of inflation, which for the past 20 years has tended to grow at between 1 and 4 percent per year. It is of interest to examine a detrended version of the CPI. The trend can be removed by regress the CPI on a time trend and then adding the mean value of the CPI to the regression residuals. The time trend regression is as follows.

```
cpi_mnth <- cpi_mnth %>%
  mutate(trend = row_number())
cpi_lm <- lm(CPI ~ trend, data = cpi_mnth)
summary(cpi_lm)$coefficients[,1:4]
```

```
##                Estimate  Std. Error  t value Pr(>|t|)
## (Intercept) 127.8659946 0.277816587 460.2533        0
## trend         0.3643515 0.001250663 291.3267        0
```

The next step is to add the regression residuals to our main data frame and then create a new column which is the detrended CPI series.

```
resid <- as_tibble(cpi_lm$resid) %>%
  rename(resid=value)
cpi_mnth <- dplyr::bind_cols(cpi_mnth,resid)

cpi_mean <- mean(cpi_mnth$CPI)

cpi_mnth <- cpi_mnth %>%
  mutate(CPI_detr = resid  + cpi_mean)
```

This detrended CPI can now be plotted.

```
autoplot(cpi_mnth, CPI_detr) +
  labs(title = "Monthly U.S. CPI with Deterministic Time Trend Removed",
       subtitle = "Index 1982-1984=100",
       y = "CPI", x="")
```

The detrended CPI which is shown in Figure 5.3 has considerable variation and it is not at all clear that this residual series is stationary. The strong trend in Figure 5.2 was clearly masking this variation. In Chapter 5 we will revisit the CPI data series and formally test for stationarity.

5.2.2 Daily Price of Crude Oil

Let's plot the closing daily futures prices for crude oil.

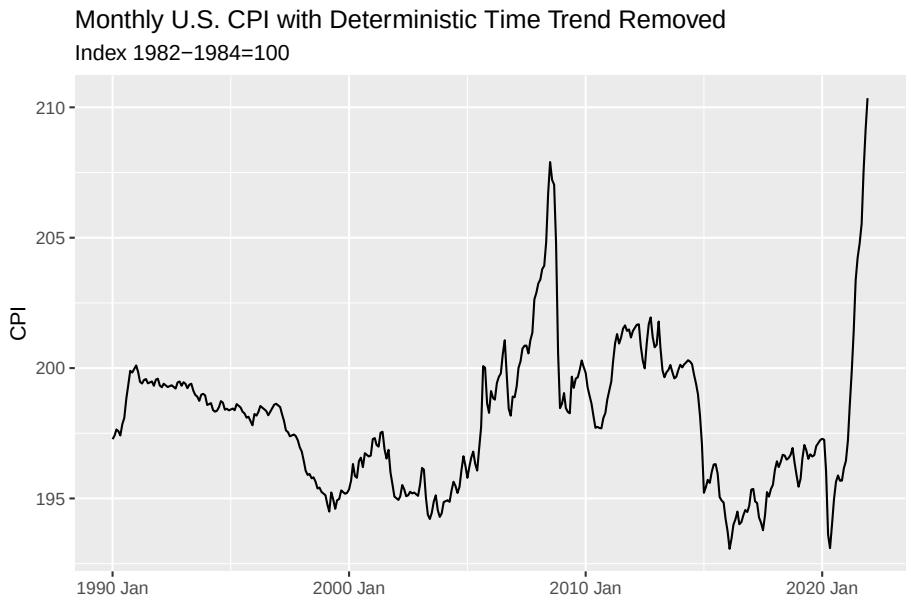

Figure 5.3: Monthly Detrended U.S. Consumer Price Index (CPI)

```
autoplot(crude_daily, Close) +
  labs(title = "Closing Price of WTI Crude Oil Futures",
       subtitle = "Rolling Through Expiring Contracts",
       y = "USD $/barrel", x = "")
```

This plot of daily crude oil plots in Figure 5.4 does not have an obvious deterministic trend or seasonality. However, it does appear to have stochastic trends, which means that price is following a random walk rather than being continually attracted toward its long term mean. In Chapter 5 will conduct formal testing for the presence of a unit root in order to confirm if the price of crude oil is indeed a random walk. To do some exploratory analysis let's estimate the following standard one-lag autoregressive model: $y_t = \alpha + \beta y_{t-1} + \epsilon_t$.

```
crude_lm <- lm(Close~lag(Close),data=crude_daily)
summary(crude_lm)$coefficients[,1:4]
```

```
##              Estimate  Std. Error    t value  Pr(>|t|)
## (Intercept) 0.2793832 0.170494689   1.638662 0.1015333
## lag(Close)  0.9953425 0.002967672 335.395024 0.0000000
```

Notice that the estimate of β is 0.9953, which is very close to 1. This is our

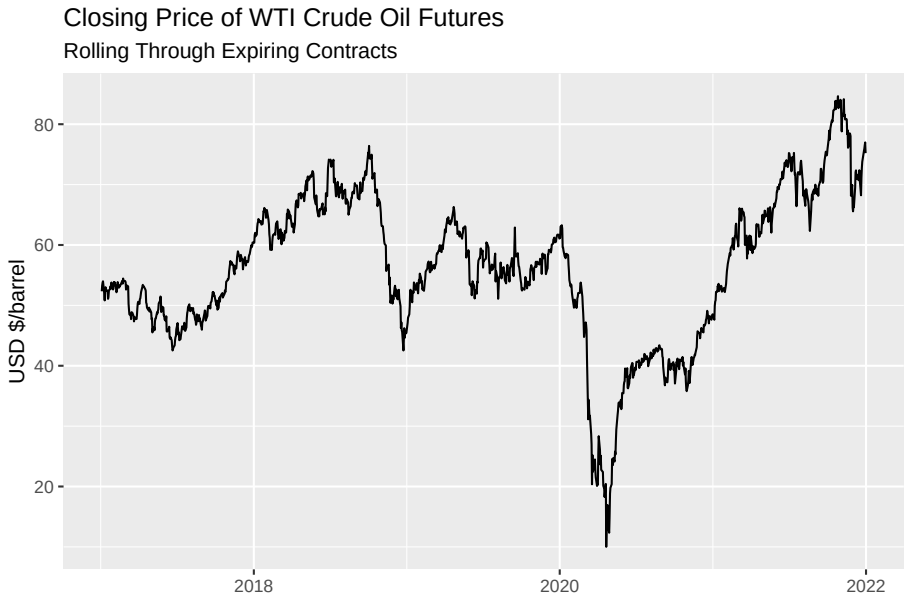

Figure 5.4: Daily Price of Crude Oil: 2017 - 2022

first clue that the daily price of crude oil has a unit root and as such is non stationary due to the presences of stochastic trends.

Another sign that the daily price of crude oil is non-stationary is that price shocks from many periods ago continue to have influence on the current price. With stationary data the price shock is eventually forgotten because the price is attracted to a constant mean in the long run. To demonstrate how this works, lets regress the price of crude oil on its lag 500 days earlier.

```
crude_lm <- lm(Close~lag(Close,500),data=crude_daily)
summary(crude_lm)$coefficients[,1:4]
```

```
##                    Estimate Std. Error   t value      Pr(>|t|)
## (Intercept)     104.7217176 3.49699668  29.94619 8.540971e-131
## lag(Close, 500)  -0.8643152 0.06015272 -14.36868  1.343086e-41
```

Notice from the previous regression that the 500 day lag is a highly statistically significant. As noted, this long memory outcome is an indicator of non-stationarity. Later in this chapter a plot of the crude oil autocorrelation function emphasizes this feature.

5.2.3 Monthly U.S. Potato Producer Price Index (PPI)

In this section we examine the existence of trends in the producer price index (PPI) for U.S. Russet potatoes. This data was previously read in as *ppi_potat.RDS* Let's plot the potato price series:

```
autoplot(ppi_potat, PPI) +
  labs(title = "Monthly U.S. PPI for Potatoes",
       subtitle = "Index Dec 1991=100",x="")
```

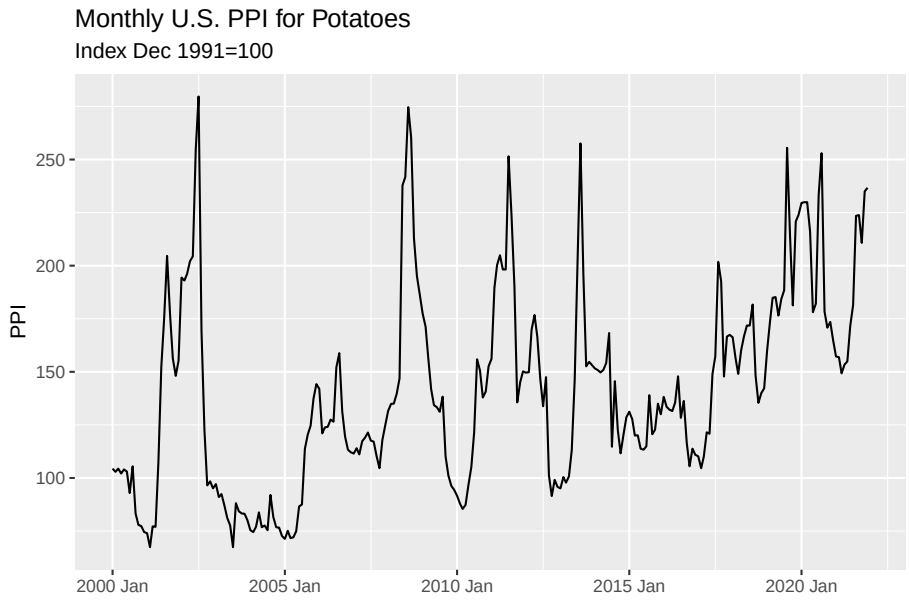

Figure 5.5: Monthly U.S. Producer Price Index (PPI) for U.S. Russet Potatoes

It is obvious from Figure 5.5 that the potato PPI is much more volatile than the much broader CPI. We should expect seasonality in the potato PPI, with the lowest price occurring in the late fall when the potatoes are harvested, and the highest price occurring in the late summer when the old crop is transitioning to the new crop. Later in this chapter the seasonal component of the producer PPI for potatoes is examined in some detail.

5.3 Autocorrelation

In this section we examine the concept of autocorrelation in a time series. After formally defining autocorrelation we note that more frequently collected data

(e.g., daily) generally has higher levels of autocorrelation than less frequently collected data (e.g., monthly). We show that this holds for the case oil by first aggregating the daily crude oil price into weekly and monthly averages and then comparing plots of autocorrelation for the weekly and monthly prices. It is worth noting that when working with daily prices it is common practice to aggregate to the weekly or monthly level as a means to reduce autocorrelation rather than including a large number of lags.

5.3.1 Autocorrelation Defined

Autocorrelation is a measure of the correlation between the current price and the price lagged by k periods. The level of autocorrelation as a function of k is the *autocorrelation* function. A plot of the autocorrelation function is a *correlogram*.

Let $(x_1, x_2, ..., x_T)$ denote the observe realization of a financial time series such as the weekly average price of crude oil. Current values are influenced by past values and so it is natural to define the *autocorrelation* of a time series as a function of the lag length, k:

$$\rho_k = \frac{Cov(x_t, x_{t-k})}{[V(x_t) \cdot V(x_{t-k})]^{1/2}}$$

In this equation, $Cov()$ is the covariance function and $V()$ is the variance function. Not surprisingly, the number of lags for which autocorrelation is significant depends on the frequency of the time series. For example, there is likely to be many statistically significant lags with weekly data and relatively few significant lags with monthly data.

5.3.2 Plots of Daily, Weekly and Monthly Crude Oil Prices

The code below aggregates the daily prices of crude oil into weekly and monthly averages.

```
crude_oil_wk <- crude_daily %>%
  dplyr::select(Date,Close) %>%
  mutate(Year_Week = yearweek(Date)) %>%
  index_by(Year_Week) %>%
  summarise_all(mean)
```

and

```
crude_oil_mth <- crude_daily %>%
  dplyr::select(Date,Close) %>%
  mutate(Year_Month = yearmonth(Date)) %>%
  index_by(Year_Month) %>%
  summarise_all(mean)
```

A plot of these weekly and monthly average prices are as follows:

```
autoplot(crude_oil_wk, Close) +
  labs(title = "Weekly Average Closing Price of WTI Crude Oil Futures",
       subtitle = "Rolling Through Expiring Contracts",
       y = "USD $/barrel", x="")
```

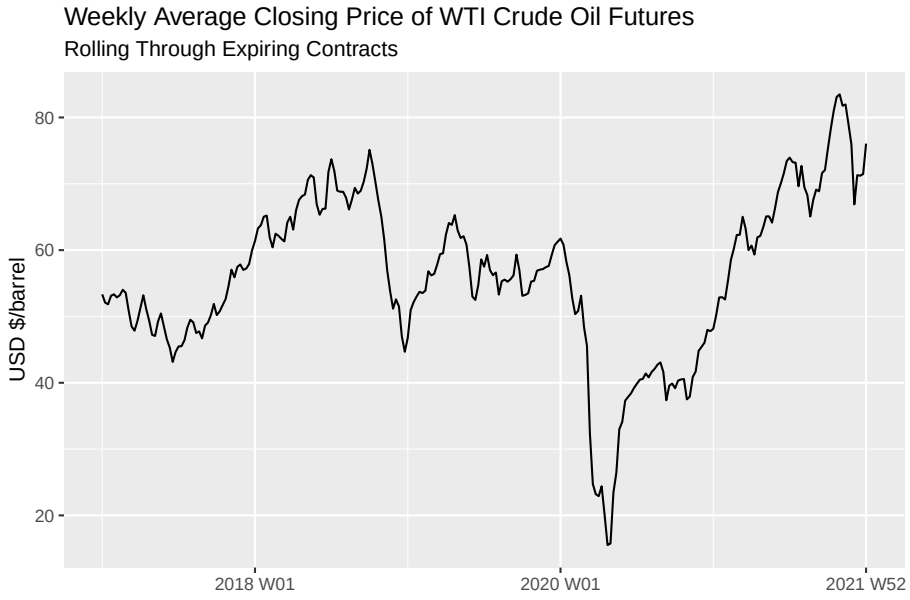

Figure 5.6: Weekly Average of Crude Oil

and

```
autoplot(crude_oil_mth, Close) +
  labs(title = "Monthly Average Closing Price of WTI Crude Oil Futures",
       subtitle =  "Rolling Through Expiring Contracts",
       y = "USD $/barrel",x="")
```

In comparison to the daily price of crude oil which is illustrated in Figure 5.4,

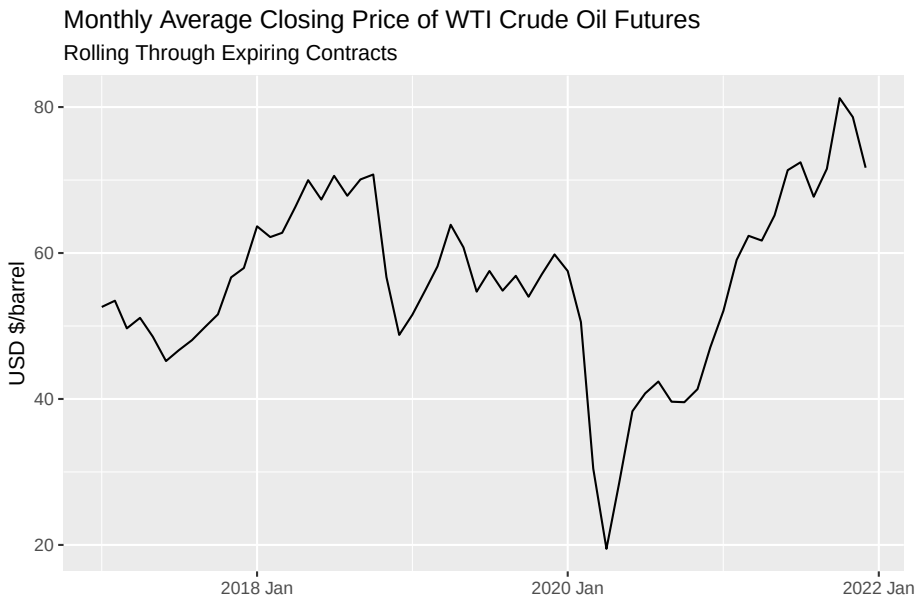

Figure 5.7: Monthly Average of Crude Oil

the plots of weekly and monthly averages of crude oil prices are much coarser. Not surprisingly, the coarseness translates into a lower level of autocorrelation.

5.3.3 Crude Oil Correlograms

To better understand how data frequency influences the level of autocorrelation it is useful to examine the correlograms for weekly and monthly crude oil prices. The correlogram for weekly prices together with the corresponding 95 percent confidence intervals can be generated as follows:

```
crude_oil_wk %>%
  ACF(Close) %>%
  autoplot() + labs(title = "Crude Oil Correlogram with Weekly Prices")
```

Notice in Figure 5.8 that the autocorrelation function drops off very slowly. The slow drop is largely the result of the non-stationarity of the data. This outcome is expected in light of the results from the previous section where it was shown that the lag or price was still highly statistically significant 500 days part.

The correlogram for crude oil aggregated to the monthly level is as follows:

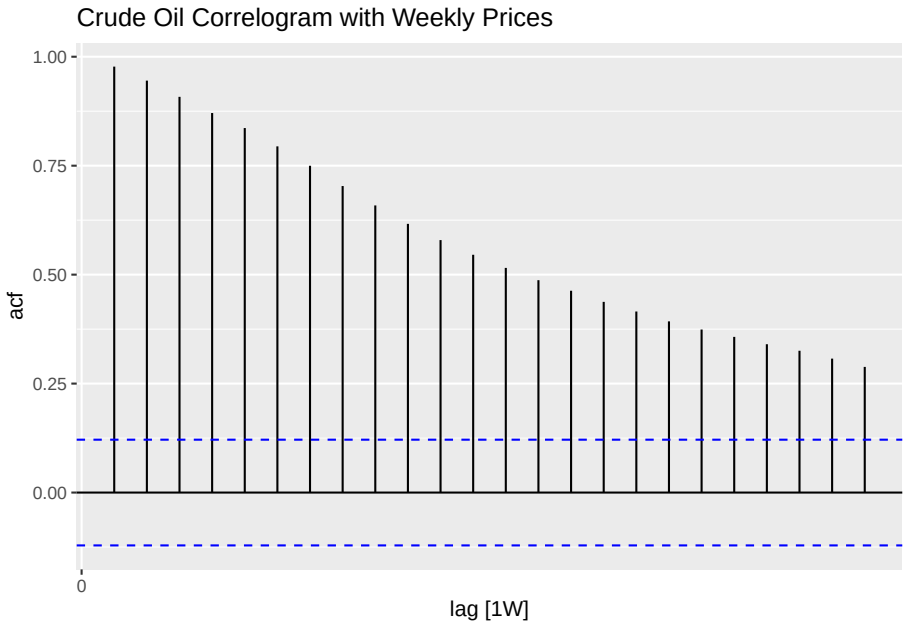

Figure 5.8: Autocorrelation of Weekly Crude Oil Prices

```
crude_oil_mth %>%
  ACF(Close) %>%
  autoplot() + labs(title = "Crude Oil Correlogram with Monthly Prices")
```

A comparison of Figures 5.8 and 5.9 confirms that the autocorrelation falls off much faster with the monthly average data versus the weekly average data. Nevertheless, the fact that the monthly autocorrelation does not appear to be converging toward zero is suggestive that monthly crude oil prices are non-stationary.

To better understand how non-stationary affects the pattern of autocorrelation it is useful to examine a correlogram for a stationary version of crude oil prices. Let's work with the weekly return, which is calculated as the weekly difference in the log of prices (the log difference is a measure of percent change in continuous time). The following code calculates the weekly return and adds it as a new column in the *crude_wk* data frame:

```
crude_wk <- crude_oil_wk %>%
  mutate(rtn_crude = difference(log(Close))) %>%
  select(Year_Week,Close,rtn_crude)
```

After deleting the first row of *crude_wk* which contains "NA" due to first dif-

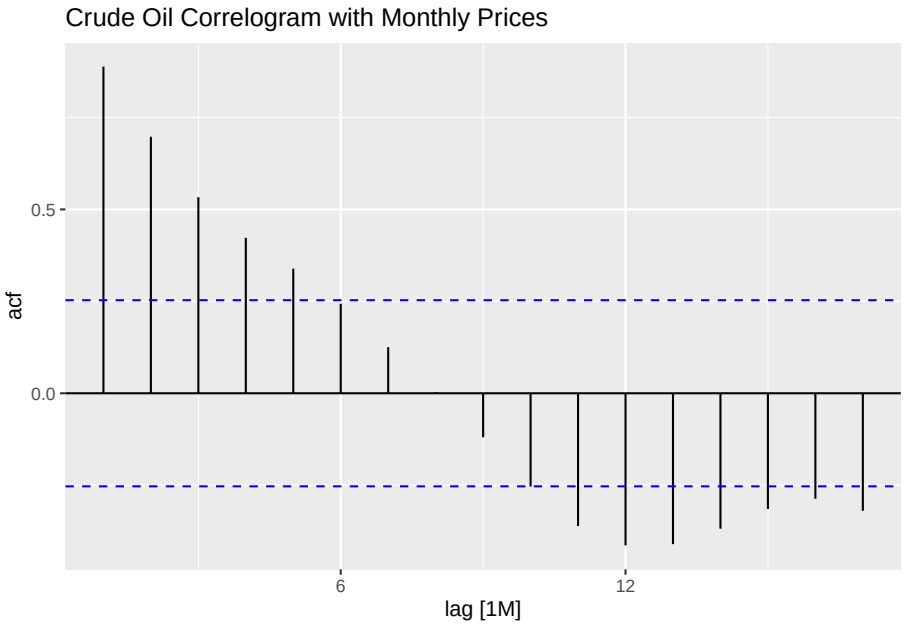

Figure 5.9: Autocorrelation of Monthly Crude Oil Prices

ferencing, the correlogram for the newly-created return variable is generated as follows:

```
acf_wk <- crude_wk[-1,] %>%
  ACF(rtn_crude) %>%
  autoplot()
acf_wk
```

Figures 5.10 makes it clear that the autocorrelation estimates for crude oil returns are much smaller and die out much more quickly as compared to the previous case of crude oil prices. As will be explained in the next module, a key feature of stationary data is that autocorrelations fade away to zero over a relatively small number of periods.

5.3.4 Test for Autocorrelation

It is also useful to formally test for the presence of autocorrelation. It is common to use a Ljung–Box test, which is part of the 'portmanteau' family of tests. This test, which is part of the *feats* package in R, requires the number of lags to be specified. If the time series to be tested is a set of residuals from an ARIMA model then the degrees of freedom (dof) must be set equal to $m - p - q$ where

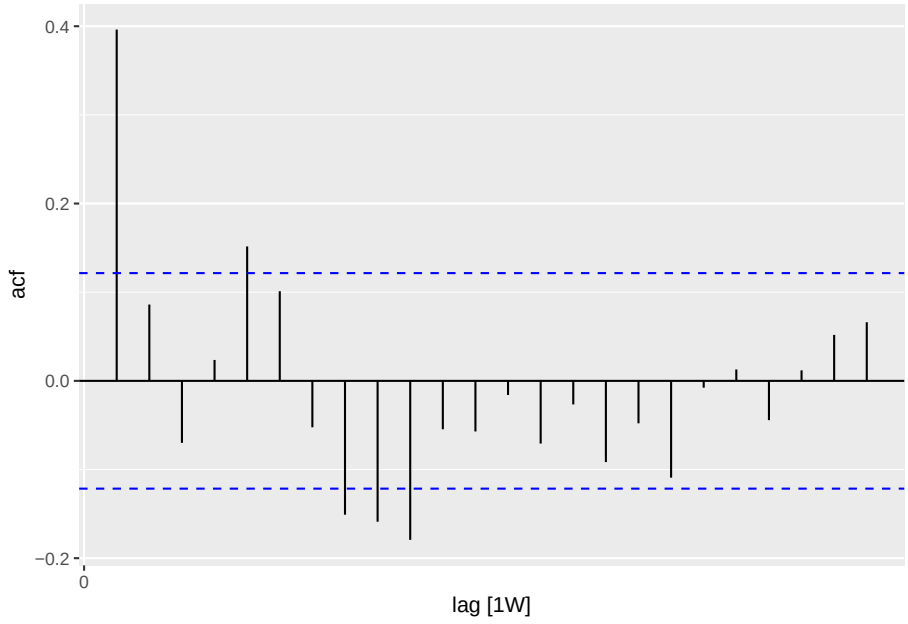

Figure 5.10: Autocorrelation of Weekly Crude Oil Returns

m is the number of lags, and p and q are the parameters of the ARIMA model. In other cases it is sufficient to omit the *dof* parameter.

For weekly data with a large number of weeks (i.e., hundreds) it is common to set the lag equal to 20. Using this rule, the Ljung test statistic can be generated as follows:

```
ljung_box(crude_wk[-1,]$rtn_crude, lag = 20, dof = 0)
```

```
##      lb_stat    lb_pvalue
## 8.585456e+01 3.897471e-10
```

The null hypothesis with the Ljung Box test is no autocorrelation. The p value from the test output is close to zero and so the null can be rejected and we can conclude that autocorrelation is present. This outcome was expected given the results which are implied by the previous correlogram.

5.3.5 Lagged Variables to Eliminate Autocorrelation

Autocorrelation must generally be eliminated before estimating a time series model such as a cointegration relationship, a vector autoregression (VAR) or

vector error correction model (VECM). A standard way to eliminate autocorrelation is to include lagged values of the variable on the right hand side of the regression. Let's regress the crude oil return for week t on the values of crude oil return for weeks $t-1$, $t-2$ and $t-3$. These lagged values are added to the *crude_wk* tsibble as follows:

```
crude_wk <- crude_wk %>% mutate(L.rtn_crude = lag(rtn_crude,1),
                                L2.rtn_crude = lag(rtn_crude,2),
                                L3.rtn_crude = lag(rtn_crude,3))
```

The reqression in question is created with the following code:

```
modL3 <- lm(rtn_crude ~ L.rtn_crude + L2.rtn_crude + L3.rtn_crude, data = crude_wk[-c(1:3),])
summary(modL3)$coefficients[,1:4]
```

```
##                  Estimate  Std. Error    t value     Pr(>|t|)
## (Intercept)   0.001022419 0.003252237  0.3143743 7.534959e-01
## L.rtn_crude   0.424488774 0.062759913  6.7636928 9.272711e-11
## L2.rtn_crude -0.046454974 0.068153490 -0.6816228 4.961010e-01
## L3.rtn_crude -0.088510663 0.062736184 -1.4108391 1.595205e-01
```

The regression results reveal that the first two lags are statistically significant and the third lag is insignificant. This suggests that we should drop the third lag and re-estimate the model with two lags.

```
modL2 <- lm(rtn_crude ~ L.rtn_crude + L2.rtn_crude, data = crude_wk[-c(1:2),])
summary(modL2)$coefficients[,1:4]
```

```
##                  Estimate  Std. Error    t value     Pr(>|t|)
## (Intercept)   0.001051794 0.003246766  0.3239514 7.462404e-01
## L.rtn_crude   0.432170606 0.062554704  6.9086828 3.891281e-11
## L2.rtn_crude -0.085465522 0.062531334 -1.3667631 1.729034e-01
```

With the two-lag model both lags are statistically significant, which means that no further adjustments are necessary. We can now check to ensure that no autocorrelation remains in the residuals. The following code collects the residuals, adds "NAs" to ensure the residuals have the same length as the original data and then add the residuals to the *crude_wk* tsibble:

```
crude_wk <- crude_wk %>% mutate(L2.resid = c(NA, NA, NA, modL2$residuals))
```

A correlogram of the residuals can be generated as follows: {{r resid_correl, warning=FALSE, fig.cap='Correlogram of Regression Residuals',fig.align='center'}
crude_wk[-c(1:3),] %>% ACF(L2.resid) %>% autoplot()

This correlogram reveals much smaller autocorrelation in the residuals as compared to unadjusted prices. To formally test whether the autocorrelation is no longer present we can once again use the Ljeung Box test.

```
ljung_box(crude_wk[-4,]$L2.resid, lag = 10, dof = 0)
```

```
##     lb_stat   lb_pvalue
## 20.95390608  0.02141728
```

The p value of 0.062 indicates that the null of no autocorrelation cannot be rejected at the 5 percent level of significance. This confirms that the two-lag regression has effectively eliminated the autocorrelation from the regression residuals.

5.3.6 Partial Autocorrelation

The partial autocorrelation function (PACF) is also commonly used in the statistical analysis of time series data. This function measures only the direct correlation between the current price and the price k periods ago. More details about what is meant by the direct and indirect effects is provided below.

The correlogram for the PACF is generated similar to the correlogram for the ACF. It is useful to create a side-by-side comparison of the ACF (left side) and the PACF (right side) for the weekly return in crude oil market:

```
pacf_wk <- crude_wk[-1,] %>% PACF(rtn_crude) %>% autoplot()
grid.arrange(acf_wk, pacf_wk, ncol = 2)
```

Figure 5.11 shows that the correlation for the one-week lag is approximately 0.4 for both the PACF and the ACF. For the two week lag the correlation is negative for the PACF and positive for the ACF. This result emerges because as the name suggests the PACF examines only the direct correlation between the current return and the lagged return while controlling for indirect impacts from intermediate lags. In this case the shock to a return affects the outcome two weeks later both directly and indirectly through the outcome one week later. The ACF function measures both the direct and indirect correlation whereas the PACF measures only the direct correlation.

5.4 Seasonality

Figure 5.5 previously showed that the producer price index (PPI) for U.S. potatoes had both a linear trend and seasonality. The *STL* function in R's *feats*

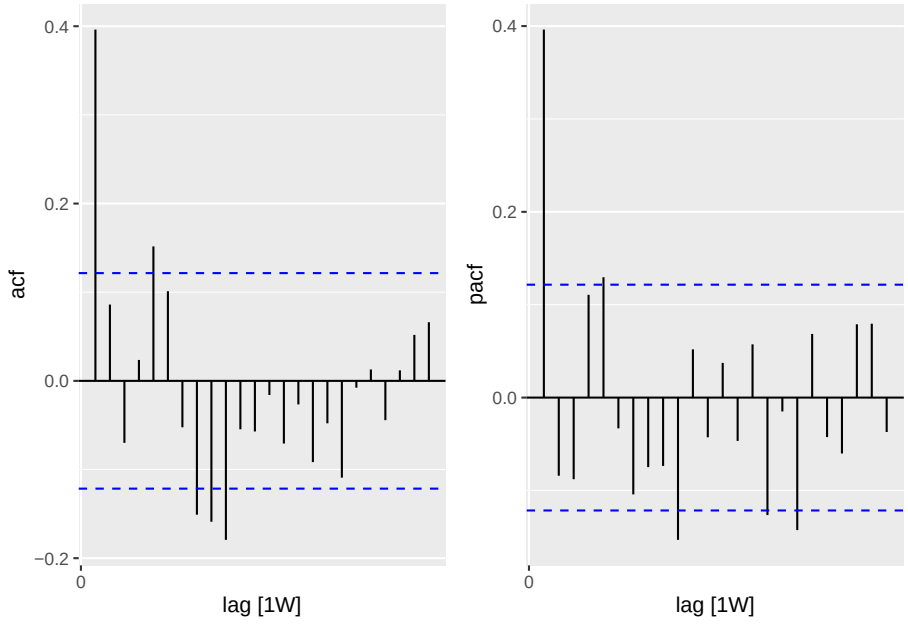

Figure 5.11: ACF versus PACF for Weekly Crude Oil Prices

package is designed to decompose time series into trend, seasonal and residual components. For the case of the potato PPI, the seasonality is more apparent with quarterly data and so let's begin our analysis by aggregating the monthly PPI values into quarterly values.

```
ppi_qtr <- ppi_potat %>%
  mutate(Year_Qtr = yearquarter(month, fiscal_start = 1)) %>%
  index_by(Year_Qtr) %>%
  summarise_all(mean) %>%
  mutate(Qtr=lubridate::quarter(month)) %>%
  select(Year_Qtr,Qtr,PPI)
```

We can call the *STL* function to create the decomposition. The *window* parameter indicates the number of observations which should be used during the decomposition procedure. These will be set at *inf* (i.e., "infinity") to ensure that the trend line is linear and the seasonality is constant over time.

```
dcmp <- ppi_qtr %>%
  model(STL(PPI ~ trend(window = Inf) + season(window = Inf)))
components(dcmp) %>%
  autoplot() + labs(x="")
```

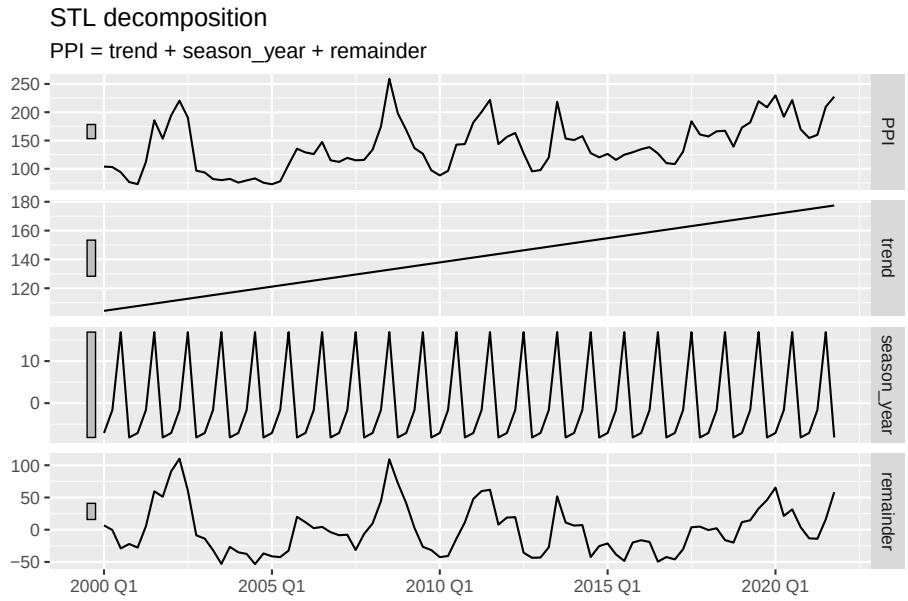

Figure 5.12: Decomposition of Potato PPI

Figure 5.12 shows that the linear trend component and the seasonality component of the potato PPI are significant features of the potato PPI. Both of these components must be removed in order to make the potato PPI stationary and thus suitable for use in a regression model. The bottom panel of Figure 5.12 show the residuals which remain after the trend and seasonality components have been removed from the potato PPI series. It is likely that this residual series has significant autocorrelation.

Lets redo the decomposition twice, first with the first difference of the potato PPI and second with the fourth difference. The reason for doing this is explained below. We begin by adding the first and fourth differences to the data frame. When doing this the first four rows are sliced off to ensure that *NA* values are not present in the final result.

```
ppi_qtr <- ppi_qtr %>%
  mutate(D.PPI=difference(PPI),
         D4.PPI=difference(PPI,4)) %>%
  slice(-c(1:4))
head(ppi_qtr)
```

```
## # A tsibble: 6 x 5 [1Q]
##   Year_Qtr   Qtr   PPI  D.PPI D4.PPI
```

```
##      <qtr> <int> <dbl>  <dbl>  <dbl>
## 1  2001 Q1     1  72.9  -3.70 -31.0
## 2  2001 Q2     2 112.   39.4    9.23
## 3  2001 Q3     3 186.   73.5   91.9
## 4  2001 Q4     4 153.  -32.5   76.7
## 5  2002 Q1     1 194.   41.3  122.
## 6  2002 Q2     2 220.   25.9  108.
```

Taking the first difference of the potato PPI will eliminate a linear trend but it will have minimal impact on seasonality. Let's look at the decomposition for the first differenced data to see if this is the case.

```
dcmp_diff1 <- ppi_qtr %>%
  model(STL(D.PPI ~ trend(window = Inf) + season(window = Inf)))
components(dcmp_diff1) %>%
  autoplot() + labs(x="")
```

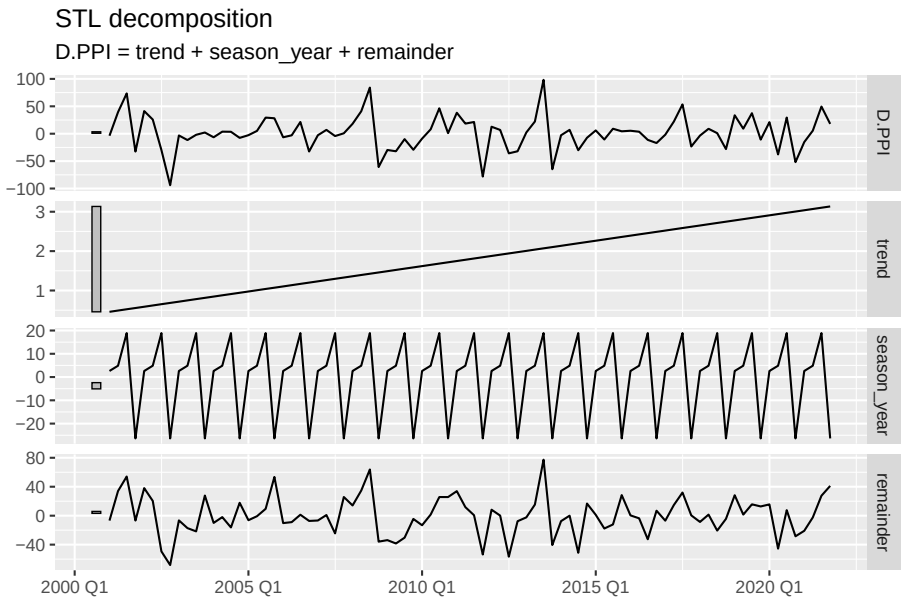

Figure 5.13: Decomposition of Difference in Potato PPI

Being mindful of the scale of the vertical axis, Figure 5.13 shows that the trend is largely gone but the seasonality remains strong.

Now let's do the decomposition with the potato PPI which with a four quarter difference (i.e., the difference for values one year apart). Since the seasonality

in potato prices is annual, this differencing procedure is expected to largely eliminate the seasonality. Let's see if this is the case.

```
dcmp_diff4 <- ppi_qtr %>%
  model(STL(D4.PPI ~ trend(window = Inf) + season(window = Inf)))
components(dcmp_diff4) %>%
  autoplot()
```

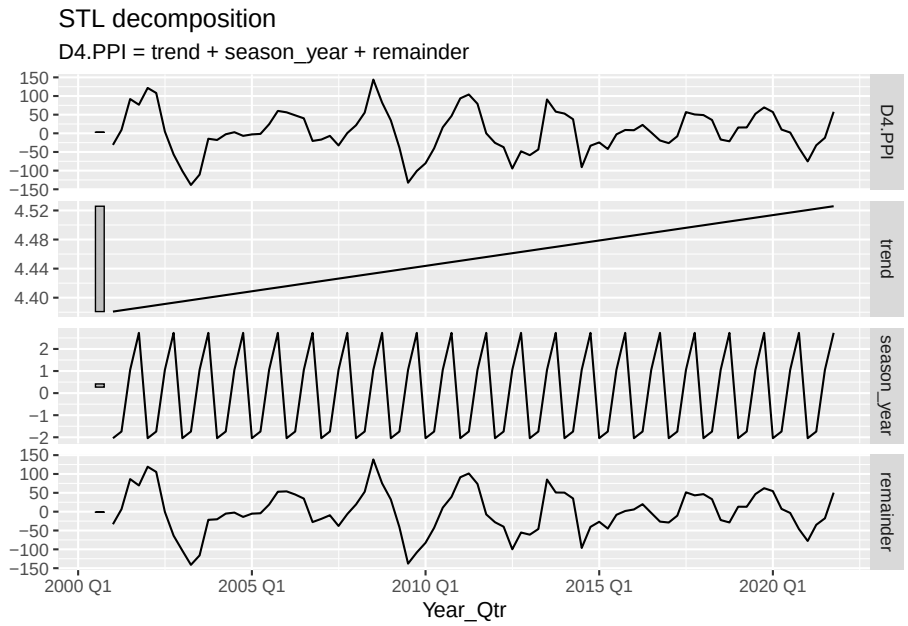

Figure 5.14: Decomposition of Difference in Potato PPI

Once again being mindful of the scale of the vertical axis, Figure 5.14 shows the expected result that the seasonality has largely been eliminated.

5.5 Summary

This chapter provided a review of the key characteristics of the stochastic processes which are relevant for this textbook. The prices analyzed in the remaining chapters of this textbook are generally in the food and energy categories. As previously noted, unlike the price of publicly traded stock which are almost always non-stationary, food and energy commodities tend to have a mixture of trend stationary and difference stationary outcomes.

Commodity prices which are difference stationary can be analyzed by working with the first difference of the price series. However, as will be shown in later

chapters, including the first difference of a commodity price in a regression rather than the price itself loses valuable information about long run properties of the price series. Fortunately, there is one important exception when a pair of non-stationary time series can be included in a regression. As will be shown in Chapter 8, if a pair of time series are cointegrated of the same order then despite being non stationary, they can be used in regression models. Of particular interest in Chapter 8 is the extent that the price of energy commodities and the price of food commodities are cointegrated.

Agricultural commodities which have seasonal production often have prices which exhibit seasonality. As was noted in this chapter, seasonality is a special type of non-stationarity. Seasonality can be eliminated while preserving the original data series (e.g., the use of seasonal dummies) or can be eliminated through seasonal differencing. The seasonal ARIMA model which is featured in Chapter 6 is the best way to implement seasonal differencing in a time series application.

Most time series are autocorrelated, and the level of autocorrelation is particularly high for high frequency data such as daily commodity prices. Unlike the case of non-stationary data, autocorrelation does not cause spurious regression outcomes. However, autocorrelation does cause downward biased standard errors of the estimated coefficients and so the autocorrelation must always be removed from time series variables. The models which are estimated in the various empirical applications within this textbook typically use lagged values of the variable as a way to eliminate the autocorrelation.

Chapter 6

Stationarity

There was considerable discussion about stationarity in Chapter 4. In this chapter a more formal treatment of stationarity is provided using vegetable oils as the case study data. Most importantly, the well-known augmented Dickey-Fuller (ADF) procedure is used to formally whether or not a time series is stationary. The ADF test is featured because it is relatively easy to understand and implement. In real-world applications the more powerful KPSS test is generally used. The KPSS test is briefly discussed at the end of this chapter.

The next section briefly discusses stationarity in the context of macroeconomic and commodity price analysis. In section 6.2 stationarity is formally defined with respect to the underlying probability distribution and the stability of the moments of that distribution. Section 6.3 establishes the connection between non-stationarity of a time series and spurios regression outcomes. The implications of using non-stationary variables in a forecasting model is examined in Section 6.4. In Section 6.5 the Augmented Dickey Fuller (ADF) test is used to formally test for a unit root (i.e., non stationarity) in a series of commodity-related time series. Shortcomings of the ADF approach are highlighted in Section 6.6, and a summary and concluding remarks are provided in Section 6.7.

6.1 Background

Nelson and Plosser [1982b] published a highly influential paper titled "Trends and random walks in macroeconomic time series: Some evidence and implications". They showed that macroeconomic variables are typically either trend stationary or difference stationary. If the variable in question is trend stationary then an economic shock to that variable will have only a temporary impact on other economic variables because the shocked variable will be gradually drawn back toward its long run trend line. In contrast, if the variable is difference

stationary then there is no long run trend line to return to, in which case shocks to one variable have a permanent impact on the outcomes of other variables. Nelson and Plosser [1982b] showed that the unit root (non-stationary) null hypothesis could be rejected, and thus stationarity confirmed, for only one out of the 14 macroeconomic variables examined.

Stationarity is also very important in the study of commodity prices. If commodity prices have unit roots (i.e., are non stationary) then there is no long term reversion to the mean and our ability to forecast commodity prices will be poor. In contrast, if commodity prices are trend stationary, then there will exist reversion to the mean and relatively accurate forecasting is possible. Using a comprehensive data set on commodity prices, Winkelried [2021] provides statistical evidence which supports the claim that real commodity prices are generally stationary. This empirical finding is consistent with theoretical models of short and long run commodity pricing. In the empirical applications of this textbook commodity prices are nominal and are generally found to be non-stationary.

Stationarity is also important when financial variables such as commodity prices are included in econometric models. If the variables are non-stationary due to unit roots then the regression outcomes will be spurious, which means "misleading". This is because non-stationary random variables are characterized by stochastic trends, and it is not uncommon for the stochastic trends of two or more variables to be coincidentally moving in the same direction, even though the variables are independent of each other. A good example is testing whether the price of energy impacts the price of food in studies which examine the food-for-fuel debate. If the variables which are included in the energy and food price regression are non-stationary then conclusions about the impact of a shock in the energy market on food prices will typically be incorrect.

6.2 Stationarity Overview

A stochastic process is said to be stationary if the corresponding time series is governed by a probability distribution which does not change over time. For example, suppose $z_t = \mu + e_t$ where e_t is a random error term which, for each period, is independently drawn from a probability distribution with mean 0 and standard deviation σ. It should be clear that z_t is stationary because the probability that $z_t < \bar{z}$ is the same regardless of the time period. Throughout the analysis below we will refer to an error term such as e_t which is independently draw from a stable probability distribution as white noise.

If a process is stationary then a shock in the current period must gradually fade away and eventually disappear. This must be the case because if a shock in the current period did not fade away then clearly the underlying probability distribution is not be constant over time. The best example of a non-stationary stochastic process for which the shock does not fade way is a simple random walk: $z_t = z_{t-1} + e_t$, where, once again, e_t is assumed to be white noise.

Stationarity also means that the moments of the time series do not change over time. For example, a financial time series with an inflationary trend is not stationary because the mean of the time series is gradually increasing over time. Similarly, if the variance of the time series is increasing over time then the series is not stationary. Finally, a financial time series with seasonality (e.g., the monthly price of lamb) is non stationary because the mean of the time series cycles throughout the year.

The previous concepts can now be made more formal. A time series is said to be *strictly stationary* if the joint probability distribution for a pair of observations, (z_t, z_{t-s}), depends on s, which is the number of periods which separate the two observations, but does not depend on t, which is the time index. For example, if a time series of annual data is strictly stationary then the probability distribution which governs the pair of outcomes in 1980 and 1985 must be the same as the probability distribution for the 2000 and 2005 pair of outcomes.

Most researchers assume the data is normally distributed and adopt the notion of *weak stationarity* (also called *covariance stationarity*). In this case for the pair (z_t, z_{t-s}) the means, variances and covariance depends on s but not on t. Specifically, $E(z_t) = \mu$, $var(z_t) = \sigma_z^2$ and $cov(z_t, z_{t-s}) = \sigma_s$ for all t and s. This definition can be extended to more than two observations but little is gained by doing so in this current context.

It is common to avoid the units of measure by analyzing the correlation rather than the covariance. The autocorrelation parameter for a time series which is covariance stationary can be expressed as

$$\rho_s = corr(z_t, z_{t-s}) = \frac{cov(z_t, z_{t-s})}{\sigma_y^2} \tag{6.1}$$

In equation (6.1) weak stationarity requires that $\rho_s \to 0$ as $s \to \infty$.

6.3 Spurious Regression

In this section we will examine the concept of a spurious regression. Spurious means that the regression results will lead to an incorrect/misleading conclusion due to a model specification error (i.e., not all of the least squares assumptions are valid). In the first of three sections we examine a spurious regression which results from a deterministic time trend. The middle section links a spurious regression outcome to stochastic trends. The last section is used to demonstrate how a spurious regression can lead to incorrect conclusions regarding the strength of the empirical evidence in the food for fuel debate.

6.3.1 Spurious Regressions and Deterministic Trends

A good example of a time series which appears to have a deterministic trend is the consumer price index (CPI). This is because the CPI is a price index of a fixed basket of goods where the price of each good is increasing steadily over time at the general rate of inflation plus a sector-specific rate of inflation. A formal test for trend stationarity in the U.S. CPI is conducted below.

In preparation for the analysis of spurious regression let's also examine monthly data on seasonally-adjusted industrial production by U.S. electric and gas utilities (shortened to "industrial production"). These data sets were cleaned in Chapter 3 and can now be read as individual RDS files. Before reading these RDS files we need to load the relevant packages into R.

```
pacman::p_load(tidyverse, here, lubridate, tsibble, feasts, gridExtra, urca, forecast)
```

```
cpi_mnth <- readRDS(here("data/ch5","cpi_mnth.RDS"))
head(cpi_mnth)
```

```
## # A tsibble: 6 x 2 [1M]
##      month   CPI
##      <mth> <dbl>
## 1 1990 Jan  128.
## 2 1990 Feb  128
## 3 1990 Mar  129.
## 4 1990 Apr  129.
## 5 1990 May  129.
## 6 1990 Jun  130.
```

```
industrial <- readRDS(here("data/ch5","industrial.RDS"))
head(industrial)
```

```
## # A tsibble: 6 x 2 [1M]
##      month industrial
##      <mth>      <dbl>
## 1 1990 Jan       71.4
## 2 1990 Feb       70.5
## 3 1990 Mar       71.5
## 4 1990 Apr       72.0
## 5 1990 May       72.4
## 6 1990 Jun       73.1
```

The *inner_join* function is used to combine the CPI and industrial production data frames. Let's also add a time trend for use in the regressions below.

```
cpi_indust <- cpi_mnth %>% inner_join(industrial, by="month") %>%
  mutate(Trend = row_number())
head(cpi_indust)
```

```
## # A tsibble: 6 x 4 [1M]
##      month   CPI industrial Trend
##      <mth> <dbl>      <dbl> <int>
## 1 1990 Jan  128.       71.4     1
## 2 1990 Feb  128        70.5     2
## 3 1990 Mar  129.       71.5     3
## 4 1990 Apr  129.       72.0     4
## 5 1990 May  129.       72.4     5
## 6 1990 Jun  130.       73.1     6
```

The CPI and Industrial Production time series can now be plotted side by side:

```
cpi_plot <- cpi_indust %>% autoplot(CPI) +
  labs(y = "U.S. CPI: Base = 1982-1984", x="Date")
indust_plot <- cpi_indust %>% autoplot(industrial) +
  labs(y = "U.S. Industrial Production Index: Base = 2017", x="Date")
```

```
grid.arrange(cpi_plot, indust_plot, ncol = 2)
```

The dual plots reveals a strong upward trend in both the CPI and Industrial Production time series. Theory suggests a very weak statistical relationship between the two data series because utility pricing tends to be regulated, and utility output (e.g., electricity) makes up a small percentage of the CPI. A regression of the CPI on Industrial Production reveals the following:

```
modA <- lm(CPI ~ industrial, data=cpi_indust)
summary(modA)$coefficients[,1:4]
```

```
##                Estimate Std. Error   t value      Pr(>|t|)
## (Intercept) -144.128417 7.76755631 -18.55518  2.971707e-55
## industrial     3.681497 0.08309699  44.30361 1.356939e-152
```

```
summary(modA)$adj.r.squared  # Adjusted R Squared
```

```
## [1] 0.8366605
```

The regression results reveal a strong goodness of fit (the adjusted R squared is 0.8314) and a highly significant slope. This regression is clearly spurious versus

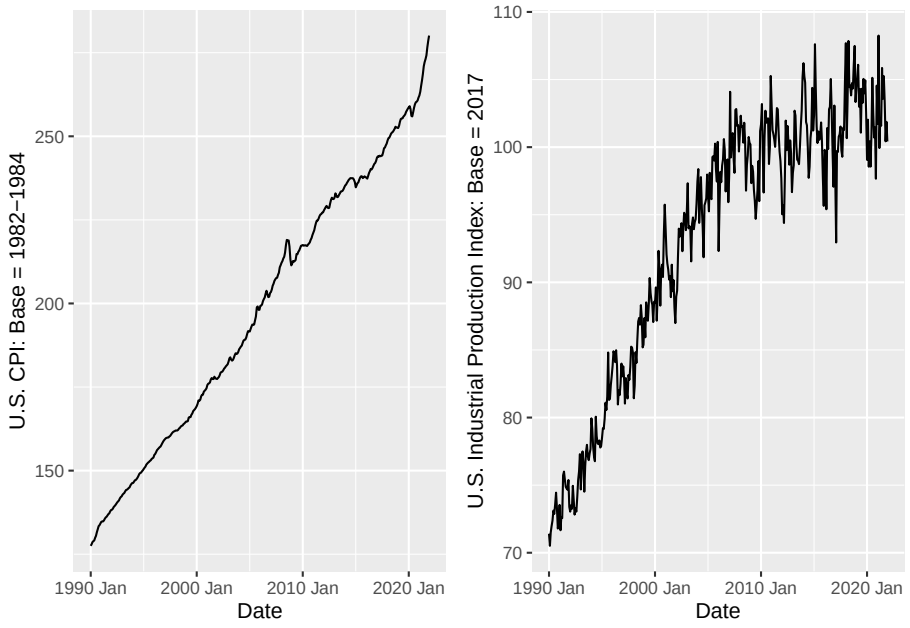

Figure 6.1: U.S. Monthly CPI and Industrial Production

causal because it is the strong positive coincidental trend in both time series which is the cause of the strong statistical relationship. If there happens to be a small causal relationship it is fully masked by the strong spurious relationship.

The left plot in Figure 6.1 suggests that the CPI may be *trend stationary*. If a linear time trend is removed from a trend stationary time series then the remaining time series will be stationary. It is less clear that Industrial Production, which is plotted on the right side of Figure 6.1, is trend stationary. The Federal Reserve inflation targets implies a reasonably predictable upward trend for the CPI data series whereas there is no equivalent external stabilizer in the case of industrial production.

To formally show that a regression of the CPI on Industrial Production is spurious it is sufficient to re-estimate the model but this time with a time trended included as a covariate. The revised results (see below) reveal an insignificant relationship between the CPI and Industrial Production, after controlling for the time trend. Thus, the original regression without the time trend is necessarily spurious due to common time trends.

```
modB <- lm(CPI ~ industrial + Trend, data=cpi_indust)
summary(modB)$coefficients[,1:4]
```

```
##                Estimate  Std. Error     t value      Pr(>|t|)
```

```
## (Intercept) 126.7840078 2.666213392  47.5520857 2.670130e-162
## industrial    0.0140658 0.034471621   0.4080401  6.834735e-01
## Trend         0.3631817 0.003128372 116.0928955 1.932503e-299
```

6.3.2 Spurious Regressions and Stochastic Trends

A time series with a deterministic trend can be made stationary by removing the deterministic trend. In contrast, a time series with a stochastic trend cannot be made stationary through detrending. Rather these types of series are difference stationary, which means that stationarity can be achieved by taking one or more differences. Stochastic trends generally emerge if the series is a random walk (possibly with drift) due to a unit root. Prices for publicly traded stocks such as Google are generally believed to have a unit root by virtue of the efficient market hypothesis. As noted in the Introduction, commodity prices are expected to be stationary due to long term mean reversion but in empirical results often fail to reject a unit root for the major commodities.

A time series which is currently stationary is said to be integrated of order 0 and is thus labeled *I(0)*. A time series which can be made stationary by taking the first difference is said to be integrated of order 1 and is thus labeled *I(1)*. An *I(2)* time series must be differenced twice to be made stationary and so forth.

A good example of a time series which has stochastic trends due to an underlying unit root is a simple random walk. A random walk is modeled by starting with a first order autoregressive (AR) stochastic process, which can be expressed as $z_t = a_1 z_{t-1} + \varepsilon_t$. The characteristic equation of this AR(1) process is given by $m - a_1 = 0$ where m is the root of the equation. The *AR(1)* is a simple random walk when $a_1 = 1$. This is because $m = 1$ is required to solve the $m - a_1 = 0$ characteristic equation.

It is useful to generate a pair of independent random walks in order to visually make the connection between a time series with a unit root and the presence of stochastic trends. After the stochastic trends have been identified one of the random walks will be regressed on the second random walk to demonstrate that stochastic trends typically give rise to a spurious regression.

```
set.seed(4)
rndata <- tsibble(
  Year = 1970:2021,
  P1 = 100 + cumsum(rnorm(n=52, mean=0, sd=3)),
  P2 = 100 + cumsum(rnorm(n=52, mean=0, sd=3)),
  index = Year
)
```

```
grid.arrange(autoplot(rndata,P1), autoplot(rndata,P2), ncol = 2)
```

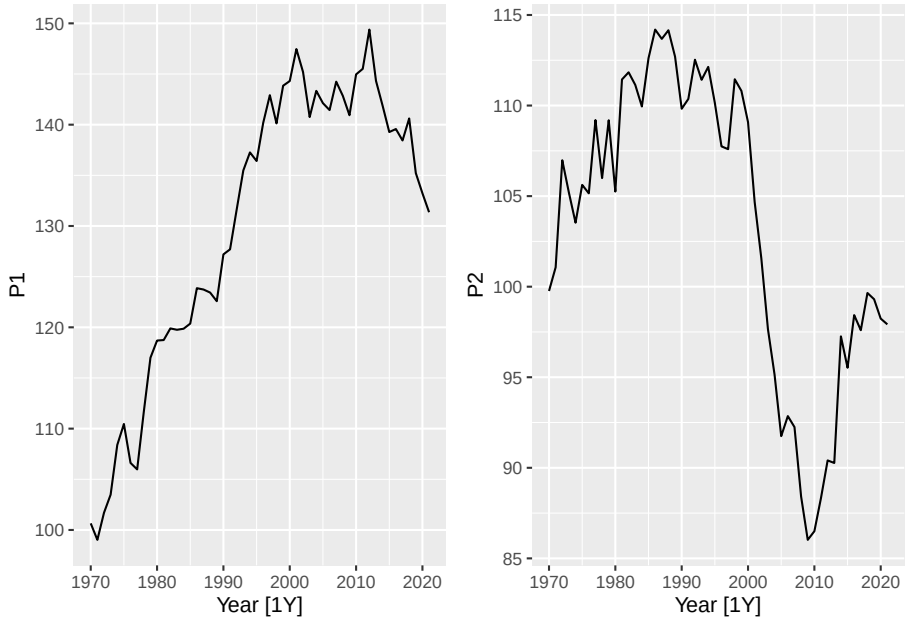

Figure 6.2: Plots of Two Independent Random Walks

The stochastic trends are obvious in both plots. If the trends coincidentally move in the same direction or in opposite directions, then the two random walks may have a significant positive or negative correlation, even though they are fully independent of each other. To examine this claim, let's regress the second random walk on the first one:

```
spur <- lm(P2 ~ P1, data=rndata)
summary(spur)$coefficients[,1:4]
```

```
##                Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) 136.7364136 9.45628634 14.459843 1.709916e-19
## P1           -0.2574038 0.07235353 -3.557585 8.304030e-04
```

```
summary(spur)$adj.r.squared  # Adjusted R Squared
```

```
## [1] 0.186037
```

This spurious regression is much less pronounced than the regression between the CPI and industrial production, which was featured in the previous section. Nevertheless, within this pair of random walks the stochastic trends are sufficiently strong to create a significant spurious regression outcome.

As discussed in Chapter 4, non-stationary time series have correlograms within which the correlations typically fade away relatively slowly. If one non-stationary variable is regressed on another, then the non-stationary properties of each variable typically appear in the regression residual. Figure 6.3 below is a correlogram of the random walk regression residuals. The slow-to-fade correlogram is consistent with the long-memory (slope fade) property of non-stationary time series.

```
rndata %>%
  mutate(Res = spur$residuals) %>%
  ACF(Res) %>%
  autoplot()
```

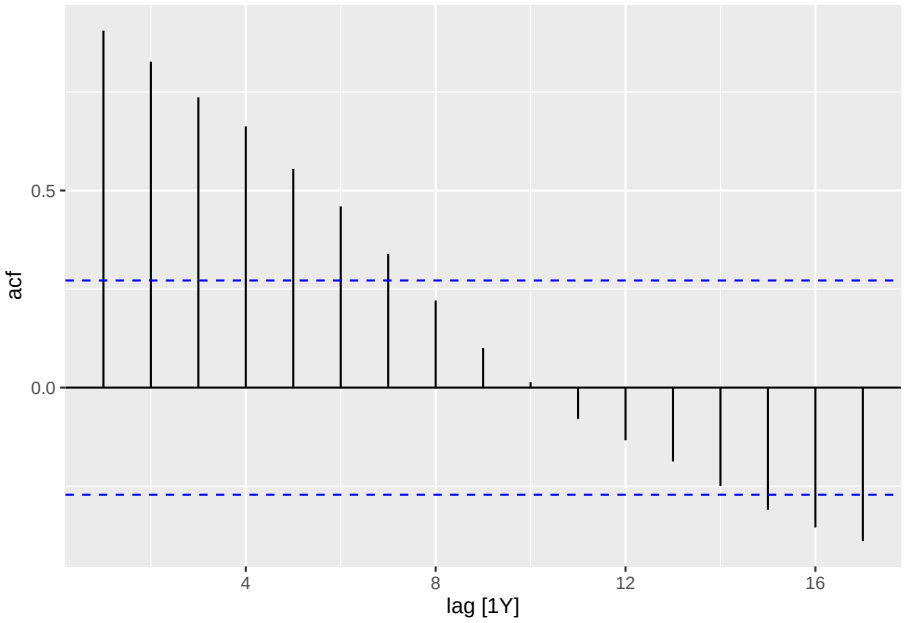

Figure 6.3: Correlogram of Regression Residuals

As noted above, a random walk is a difference stationary time series since the first difference of a random walk is stationary. Indeed, $z_t - z_{t-1} = \varepsilon_t$ is stationary because ε_t is white noise. Let's add the first differences of the pair of random walks to the original data frame and then plot this new series:

```
rndata <- rndata %>%
  mutate(D.P1 = difference(P1)) %>%
  mutate(D.P2 = difference(P2))
```

```
P1diff <- rndata[-1,] %>% autoplot(D.P1) +
  labs(y = "First Difference of Random Walk 1", x="Time Period")
P2diff <- rndata[-1,] %>% autoplot(D.P2) +
  labs(y = "First Difference of Random Walk 2", x="Time Period")
```

```
grid.arrange(P1diff, P2diff, ncol = 2)
```

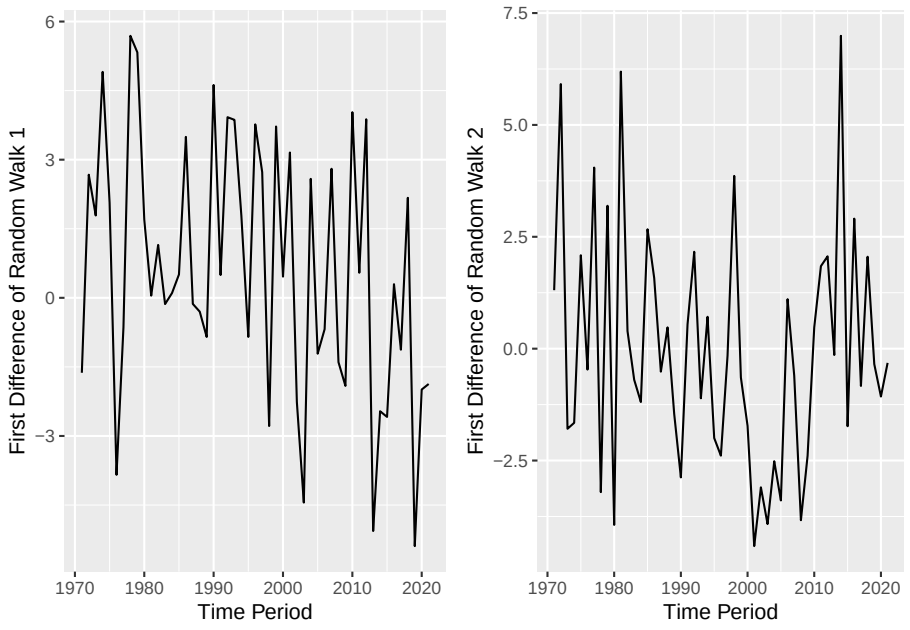

Figure 6.4: Plots of First Difference of Two Independent Random Walks

The plots of the first differences reveal that the stochastic trends which are caused by the unit roots are no longer present. If this pair of differenced random walks were regressed on each other we would obtain an insignificant relationship. This is because the spurious relationship which was present without the differencing is no longer present.

6.3.3 Spurious Correlation in the Food for Fuel Debate

Suppose we are interested in the statistical relationship between the price of crude oil and the price of wheat. The so-called food for fuel debate tells us that a relationship may exist because crude oil is used to produce gasoline, corn is used to produce ethanol (which is a gasoline substitute) and wheat substitutes for corn as a feed for livestock. There are two problems with regressing the price of

wheat on the price of oil as way to examine the food for fuel hypothesis. The first is that if the goal is to establish a causal relationship then this simple regression is likely to suffer from omitted variable bias. Second, and most relevant for time series analysis, is that the price of wheat and the price of oil may contain stochastic trends. The non-stationarity which results from the stochastic trends will result in a spurious regression and thus misleading conclusions based on the regression results.

A formal test of non-stationarity of wheat and crude oil prices due to underlying unit roots is deferred to the next section. In this section we will shed light on non-stationarity by examining the wheat and crude oil regression results and by viewing the correlograms of the individual time series and the regression residuals. Let's begin by reading in the saved RDS file (created in Chapter 3) which combines the monthly prices of wheat and crude oil.

```
wht_oil <- readRDS(here("data/ch5","wht_oil.RDS"))
```

Let's examine a side-by-side plot of this pair of time series:

```
 wht_plot <- wht_oil %>% autoplot(P_wht) +
  labs(y = "Price of Wheat ($/bu)", x="Date")
oil_plot <- wht_oil %>% autoplot(P_oil) +
  labs(y = "Price of Crude Oil ($/barrel)", x="Date")
```

```
grid.arrange(wht_plot, oil_plot, ncol = 2)
```

This pair of graphs reveals the likely presence of stochastic trends in both time series. Suppose we ignored this problem and choose to regress the price of wheat on the price of oil in an attempt to determine if the price of oil affects the price of wheat.

```
 wht_oil_reg <- lm(P_wht ~ P_oil, data=wht_oil)
summary(wht_oil_reg)$coefficients[,1:4]
```

```
##               Estimate Std. Error  t value     Pr(>|t|)
## (Intercept) 97.008520 3.82091180 25.38884 6.120871e-84
## P_oil        1.613823 0.06831949 23.62170 1.253715e-76
```

```
summary(wht_oil_reg)$adj.r.squared  # Adjusted R Squared
```

```
## [1] 0.5931776
```

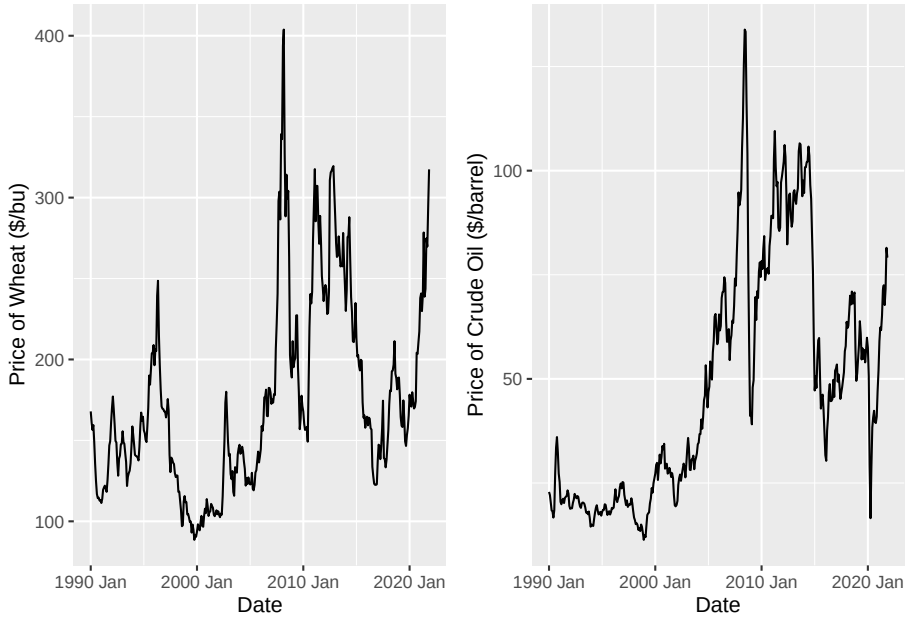

Figure 6.5: Monthly Price of Wheat and Crude Oil

These regression results reveal a highly significant relationship between the price of crude oil and the price of wheat. Someone who was not aware that stochastic trends are a cause of nonstationarity, which in turn give rise to spurious regression outcomes, may very well jump to the incorrect conclusion that there is strong evidence which supports the food for fuel debate.

If the price of oil really does have a strong causal impact on the price of wheat then we should expect that the month-to-month difference in the price of oil will have a significant causal impact on the month-to-month difference in the price of wheat. Let's test this by regressing the first difference in the price of wheat on the first difference in the price of oil.

```
wht_oil <- wht_oil %>%
  mutate(D.P_wht = difference(P_wht)) %>%
  mutate(D.P_oil = difference(P_oil))

wht_oil_diff_reg <- lm(D.P_wht ~ D.P_oil, data=wht_oil)
summary(wht_oil_diff_reg)$coefficients[,1:4]
```

```
##              Estimate Std. Error   t value   Pr(>|t|)
## (Intercept) 0.3543062  0.7209562 0.4914392 0.62339956
## D.P_oil     0.2518000  0.1519411 1.6572208 0.09830017
```

```
summary(wht_oil_diff_reg)$adj.r.squared  # Adjusted R Squared
```

```
## [1] 0.004562762
```

In this regression the difference in the price of oil is not significant at the 5 percent level, and is barely significant at the 10 percent level. As shown in the previous section the first differencing eliminated the stochastic trends and in doing so the strength of the relationship between the price of oil and the price of wheat weakened significantly. At best there is very weak evidence in the support of the food for fuel debate.

6.4 Stationarity and Forecasting

Time series models are often used for forecasting. It may be argued that it is acceptable to use non-stationary time series to estimate a forecasting model because when forecasting all that matters is the correlation, and it is not important if the correlations are spurious versus non-spurious. This line of thinking is not correct. It is important that the forecasted values converge toward the long-run mean value of the time series as the forecasting time horizon expands. This type of convergence is not possible if the underlying time series are non-stationary.

To better understand the problem with forecasting with non-stationary data let's revisit the pair of random walks which are illustrated in Figure 6.2 and were later regressed against each other: $P_2 = \alpha + \beta P_1 + e$. We can recover the intercept (α) and slope (β) coefficients from the P_2 on P_1 regression and use them to generate fitted values of P_2 as a function of P_1. In the original data frame the fitted values of P_2 are stacked below the actual values of P_2 (versus adding a new column) for the purpose of multi-line graphing with a legend.

```
spur <- lm(P2 ~ P1, data=rndata)
intercept <- summary(spur)$coefficients[1,1]
slope <- summary(spur)$coefficients[2,1]

rnfor <- rndata %>% mutate(P2 = intercept + slope*P1)

rnstack <- as_tibble(rndata) %>%
  mutate(Type = 'Actual P2') %>%
  bind_rows(as_tibble(rnfor) %>%
              mutate(Type = 'Fitted P2'))
```

Figure 6.6 shows that the fitted P_2 values do a poor job tracking the actual P_2 values even though the estimated slope coefficient is highly statistically significant. The poor in-sample forecasting performance can be attributed to the

lack of a long run mean for both P_1 and P_2. In other words, a forecast which depends on data with stochastic trends will generally have very little predictive capacity.

```
ggplot(rnstack,aes(y = P2,x = Year,color = Type)) +
  geom_line() +
  ggtitle("P2 Fitted and Actual Values")
```

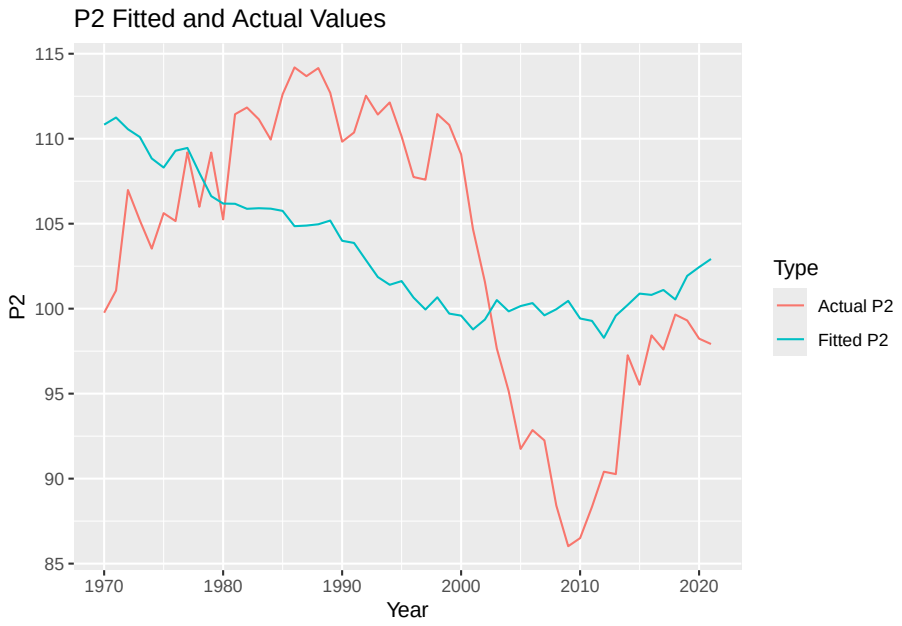

Figure 6.6: Forecast Performance of Random Walk Oil

Another way to view the poor forecasting performance of a non-stationary random variable such as the P_2 random walk is to generate prediction intervals. A 90 percent prediction interval is the range of future outcomes which is expected to contain 90 percent of the random outcomes for that time interval. Let's plot the prediction interval for P_2 at both the 80 percent and 95 level of confidence.

```
P2for <- rwf(rndata$P2, h=50, level = c(80, 95))
autoplot(P2for)
```

Figure 6.7 shows that the width of the prediction interval widens rapidly as the forecasting time horizon increases. This outcome is consistent with the poor forecasting performance of a random walk, as illustrated in Figure 6.6.

Figure 6.8 shows the prediction interval for a stationary first order autoregressive stochastic process of the type: $z_t = 20 + 0.8z_{t-1} + e_t$. In this case the inter-

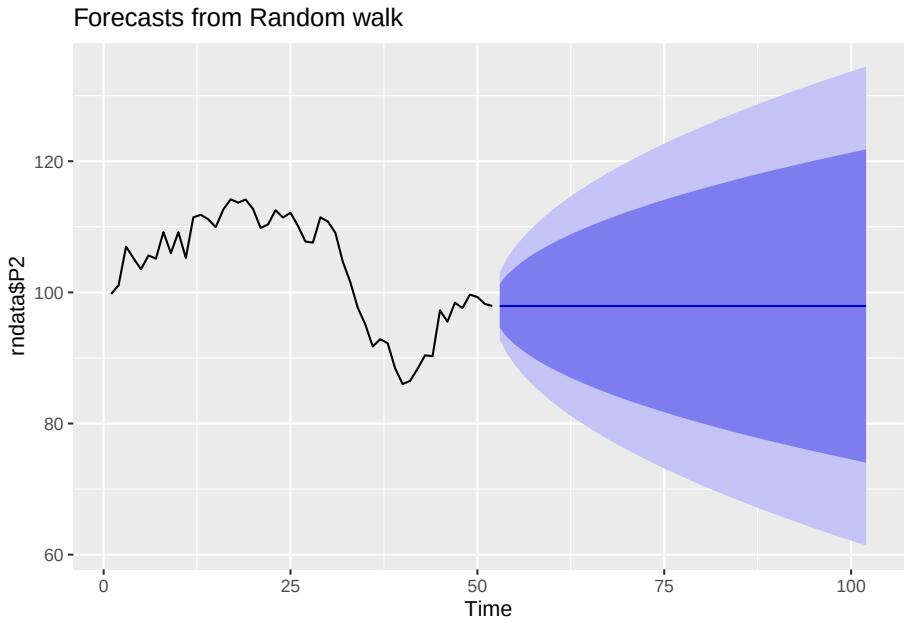

Figure 6.7: 80 and 95 Percent Prediction Intervals for Random Walk

val quickly stabilizes around a relatively narrow band with maximum width of about 5. Contrast this to the previous random walk where the interval width had reached 40 within an approximate 30 month forecasting time horizon. Forecasting is clearly much more accurate when the time series is stationary versus non-stationary.

```
set.seed(5)
n <- 50
e <- rnorm(n)
y0 <- rep(100, n)
for (i in seq.int(2, n))
  y0[i] <- 20 + 0.8 * y0[i-1] + e[i]

model <- auto.arima(ts(y0))
fcst <- forecast(model, h=30)
autoplot(fcst, ylab = 'AR(1) Process')
```

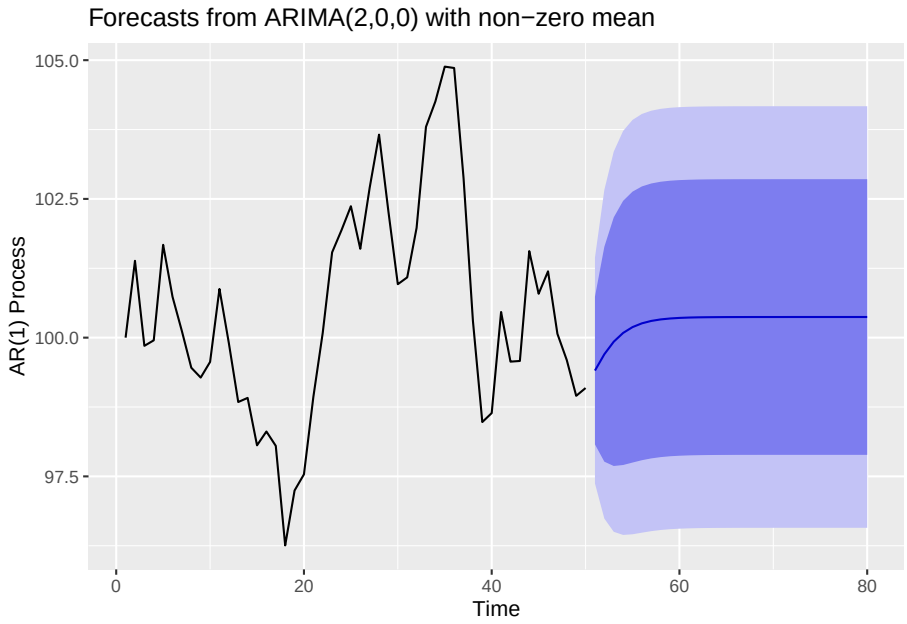

Figure 6.8: Confidence Interval for AR(1) Process

6.5 Testing for Stationarity

Plots of the ACF and PACF (i.e., the correlograms) reveal a great deal about the stationarity properties of the time series. Nevertheless, visual identification is usually not sufficiently precise and so formal statistical testing is required.

The standard test for stationarity is equivalent to testing for a unit root. The testing procedure is well explained in the Wikipedia entry for this topic: https://en.wikipedia.org/wiki/Dickey%E2%80%93Fuller_test.

A simple autoregressive stochastic process with one lag can be written as $z_t = \rho z_{t-1} + e_t$. If ρ is positive but strictly less than 1 then this time series is stationary with a mean of zero. In other words, the absence of a unit root implies that z_t is stochastically drawn towards its long run mean of zero, regardless of the starting value of z_t . If instead ρ is equal to 1 (i.e., has a unit root) then z_t is non-stationary because it has no long term mean value to which it is drawn.

In the next section we will test for the stationarity of a random walk using the well-known Dickey-Fuller test [Dickey and Fuller, 1979]. In Section 6.5.2 the augmented Dickey-Fuller (ADF) test is applied to data which has an obvious trend. This includes the producer price of eggs in the Canadian province of Alberta and the U.S. CPI. Section 6.5.3 is used to test for stochastic trends in the wheat and oil markets. Here we confirm that the unit root null hypothesis

cannot be rejected and so we conclude that both of these commodities are non-stationary.

6.5.1 Dickey-Fuller Test: Random Walk Data

The time series illustrated in both panels of Figure 6.2 are simple random walks without drift (i.e., $z_t = z_{t-1} + e_t$ with $z_0 = 100$). Even though we know the data has a unit root it is still useful to formally test the null hypothesis that a unit root is present in order to illustrate the Dickey-Fuller method. In this first specification a drift term (i.e., an equation intercept), a deterministic time trend and higher order lags are all absent. Consequently, the simple version of the Dickey-Fuller test can be used.

To construct the simple version of the test subtract z_{t-1} from both sides of $z_t = \rho z_{t-1} + e_t$ to obtain:

$$\Delta z_t = (\rho - 1) z_{t-1} + e_t = \delta z_{t-1} + e_t \tag{6.2}$$

where Δ is the first difference operator and $\delta = \rho - 1$. We can now test the hull hypothesis that $\delta = 0$ since this is equivalent to testing the unit root ($\rho = 1$) null hypothesis.

For reasons which out of scope for this chapter, it is not possible to use a standard t test when testing for the significance of the δ coefficient in equation (6.2). Instead of using the standard critical t value and the associated p value, we use a critical value which comes from a Dickey-Fuller table. The *urca* package which we will use to conduct the Dickey-Fuller test automatically converts the Dickey-Fuller critical t values into Dickey-Fuller associated p values. These p values can be interpreted in the usual way.

The *ur.df* function in the *urca* package generates the Dickey-Fuller testing statistics. The function is described on page 44 of https://cran.r-project.org/web/packages/urca/urca.pdf. The form of the function is as follows:

ur.df(y, type = c("none", "drift", "trend"), lags = 1, selectlags = c("Fixed", "AIC", "BIC"))

For this current test we want *type="none"* because our simple random walk has neither a drift or trend. There are no lags on the right side of equation (6.2) and so we will use *lags = 0* and omit the *selectlags* option.

We previously generated two independent random walks, $P1$ and $P2$. We will conduct the Dickey-Fuller unit root test on $P1$. To begin suppose that instead of using the *ur.df()* function we estimated equation (6.2) with a regular least squares regression.

```
rndata <- rndata %>%
  mutate(L.P1 = lag(P1))

olsrw <- lm(D.P1 ~ 0 + L.P1, data=rndata)
summary(olsrw)$coefficients[,1:4]
```

```
##    Estimate  Std. Error     t value    Pr(>|t|)
## 0.003858484 0.002995233 1.288208189 0.203603210
```

```
critical_t <- qt(1-0.05/2, 51)
critical_t
```

```
## [1] 2.007584
```

The regression summary shows that the estimate of δ is 0.003858, the calculated t value is 1.2882 and the critical t value is $\tau = 2.0076$. In a regular hypothesis test we would compare the calculated t value of 1.2882 with the critical value of 2.0076 and in doing so conclude that δ is not statistically significant at the 5 percent level.

As previously noted we cannot use the $\tau = 2.0076$ critical value to test the unit root null hypothesis because the critical t values must be taken from the Dickey-Fuller table. We can now use the *ur.df()* function to generate both the regression results and the Dickey-Fuller critical t values.

```
dfrw <- ur.df(rndata$P1, type = "none", lags = 0)
summary(dfrw)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.9347 -1.8604 -0.0946  2.2739  5.2807
##
## Coefficients:
```

```
##         Estimate Std. Error t value Pr(>|t|)
## z.lag.1 0.003858   0.002995   1.288    0.204
##
## Residual standard error: 2.795 on 50 degrees of freedom
## Multiple R-squared:  0.03212,    Adjusted R-squared:  0.01277
## F-statistic: 1.659 on 1 and 50 DF,  p-value: 0.2036
##
##
## Value of test-statistic is: 1.2882
##
## Critical values for test statistics:
##      1pct  5pct 10pct
## tau1 -2.6 -1.95 -1.61
```

The estimate of δ and the calculated t value is the same as what was calculated above, which is what we were expecting. This method gives the ADF critical t values. Notice that for the 5 percent test, the absolute Dickey-Fuller t value is $\tau = 1.95$, which is a little less than the $\tau = 2.0076$ critical value which comes from the standard regression.

The Dickey-fuller testing summary shows that the calculated t value of 1.288 is less than all three of the critical t values (i.e., for the 1, 5 and 10 percent tests). It is therefore not possible to reject the unit root null hypothesis. This outcome is, of course, expected because the data we are testing was constructed to have a unit root.

We know that a random walk is difference stationary. This means that we should be able to reject the unit root null hypothesis if we test the first difference of the P_1 random walk variable.

```
dfrwdiff <- ur.df(rndata$D.P1[-1], type = "none", lags = 0)
summary(dfrwdiff)@testreg$coefficients
```

```
##           Estimate Std. Error   t value     Pr(>|t|)
## z.lag.1 -0.8867099  0.1420905 -6.240458 9.958655e-08
```

```
summary(dfrwdiff)@teststat
```

```
##                tau1
## statistic -6.240458
```

```
dfrwdiff@cval
```

```
##      1pct  5pct 10pct
## tau1 -2.6 -1.95 -1.61
```

With this test the absolute t statistic of 6.2404 is well above the set of absolute critical t values (2.6 for 1%; 1.95 for 5% and 1.61 for 10%) and so the unit root null hypothesis can indeed be rejected with a high degree of confidence.

6.5.2 Dickey-Fuller Test for Possible Trend Stationary Data

Testing real world data is much more complicated than using the simple Dickey-Fuller test for a unit root in a random walk, which was examined in the previous section. In this section we will focus on testing for stationarity when a series has an obvious trend, such as the U.S. CPI, which is illustrated in Figure 6.1. Before considering this case we will test for trend stationarity in the producer price of eggs in the Canadian province of Alberta. This data series was chosen because the pricing of Canadian eggs is heavily influence by marketing board quotas and the associated administrative pricing. The long-term trend in producer prices for Canadian eggs is therefore expected to reflect long-term relatively stable inflation in farm input prices.

Let's begin by reading in the data, which is stored in the RDS file which was created in Chapter 3. After the data arrives the price variable is logged and five lags and a trend line are added to the *tsibble* object.

```
m <- readRDS(here("data/ch5", "eggs.RDS")) %>%

  mutate(lnp = log(price),
         L.lnp = lag(lnp,1),
         D.lnp = difference(lnp),
         LD1.lnp = lag(D.lnp,1),
         LD2.lnp = lag(D.lnp,2),
         LD3.lnp = lag(D.lnp,3),
         LD4.lnp = lag(D.lnp,4),
         LD5.lnp = lag(D.lnp,5),
         trnd = row_number())
```

Figure 6.9 shows the relatively stable upward trend in the monthly Alberta producer price for the large grade of eggs over the 1985 to 2021 period. Despite the strong influence of Alberta's egg marketing board it can be seen that there is still a reasonable amount of variability in the producer price of eggs. Consequently, trend stationarity for this price series is possible but not guaranteed.

```
m %>% autoplot(price) +
  labs(y = "Monthly Alberta Producer Egg Price", x="Date")
```

When adapting the Dickey-Fuller model as given by equation (6.2) to accommodate this price series we must add an intercept term, a deterministic time trend and account for higher order autocorrelation. The new equation is

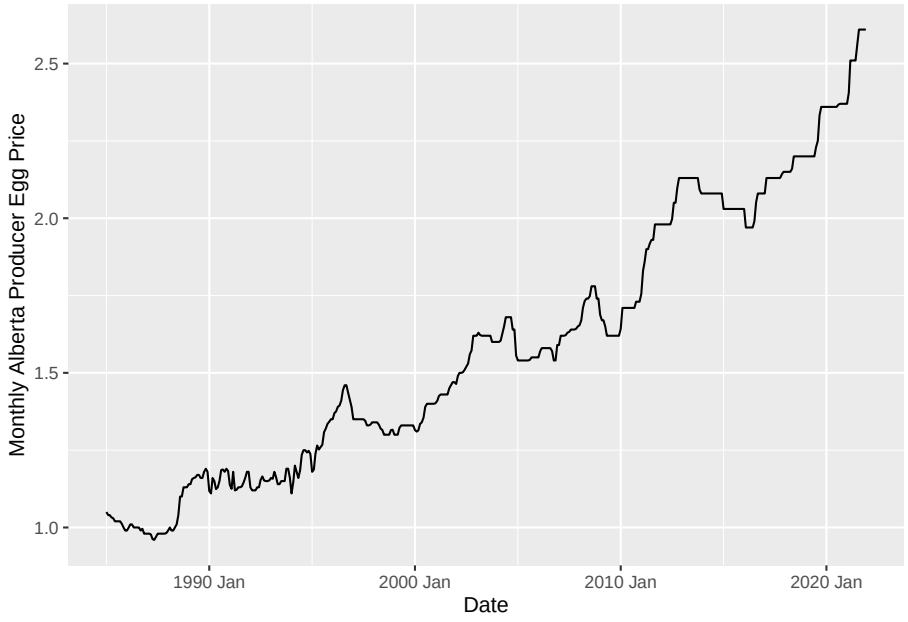

Figure 6.9: Graded Large

$$\Delta z_t = a_0 + a_1 t + \delta z_{t-1} + \sum_{j=1}^{k} \gamma_j \Delta z_{t-j} + e_t \tag{6.3}$$

Lag Selection

An important parameter in equation (6.3) is k, which is the number of lags of Δz_t to include on the right side of the regression. To verify the importance of k let's use the egg price data which we previously imported to estimate equation (6.3) with lags ranging from zero to four. We will then use the set of estimated equations to observe how the calculated t statistic varies across the different specifications.

The estimated equation with no lags is as follows:

```
adf0 <- lm(m$D.lnp ~ m$L.lnp + m$trnd)
summary(adf0)$coefficients[,1:4]
```

```
##                  Estimate   Std. Error    t value    Pr(>|t|)
## (Intercept) -2.277275e-04 1.283011e-03 -0.1774945 0.859201638
## m$L.lnp     -3.789602e-02 1.300381e-02 -2.9142245 0.003747580
## m$trnd       8.252672e-05 2.710253e-05  3.0449818 0.002466646
```

In this particular case the t statistic for the δ coefficient (i.e., the coefficient on the lagged price variable, "m$L.lnp") is -2.914. Using this same procedure, a full set of t statistics can be calculated for the ADF tests which are specified with a varying number of lags. The test statistics are calculated as follows:

```
adf0 <- lm(m$D.lnp ~ m$L.lnp + m$trnd)
adf1 <- lm(m$D.lnp ~ m$L.lnp + m$trnd + m$LD1.lnp)
adf2 <- lm(m$D.lnp ~ m$L.lnp + m$trnd + m$LD1.lnp + m$LD2.lnp)
adf3 <- lm(m$D.lnp ~ m$L.lnp + m$trnd + m$LD1.lnp + m$LD2.lnp + m$LD3.lnp)
adf4 <- lm(m$D.lnp ~ m$L.lnp + m$trnd + m$LD1.lnp + m$LD2.lnp + m$LD3.lnp + m$LD4.lnp)
```

The third row of the table below shows the calculated t statistics for the estimated δ coefficient assuming $k = 0, 1, 2, 3, 4$. Notice that these t values range from an absolute low of 2.9142 with no lags to an absolute high of 3.7100 with four lags. Clearly the choice of number of lags is important because it can affect whether the unit root null hypothesis is rejected versus not rejected.

```
c0 <- as.data.frame(summary(adf0)$coefficients[2,1:4]) %>%
  rename(delta_lag0 = 1)
c1 <- as.data.frame(summary(adf1)$coefficients[2,1:4]) %>%
  rename(delta_lag1 = 1)
c2 <- as.data.frame(summary(adf2)$coefficients[2,1:4]) %>%
  rename(delta_lag2 = 1)
c3 <- as.data.frame(summary(adf3)$coefficients[2,1:4]) %>%
  rename(delta_lag3 = 1)
c4 <- as.data.frame(summary(adf4)$coefficients[2,1:4]) %>%
  rename(delta_lag4 = 1)
lagreg <-cbind(c0,c1,c2,c3,c4)
lagreg
```

```
##             delta_lag0    delta_lag1   delta_lag2   delta_lag3    delta_lag4
## Estimate   -0.03789602 -0.0458590656 -0.040940873 -0.043593223 -0.0496050188
## Std. Error  0.01300381  0.0128635145  0.013042924  0.013229137  0.0133703227
## t value    -2.91422450 -3.5650494626 -3.138933604 -3.295243037 -3.7100838813
## Pr(>|t|)    0.00374758  0.0004038252  0.001810797  0.001064034  0.0002342367
```

A popular way to select the number of lags to include in an ADF test is to choose the model which generates the lowest value of the Akaike information criterion (AIC) or a related measure of statistical information. Lets modify the above table by adding a new row which contains the AIC statistic for each of the five lag configurations.

```
lagfull <- rbind(lagreg,c(AIC(adf0), AIC(adf1), AIC(adf2), AIC(adf3), AIC(adf4)))
rownames(lagfull)[rownames(lagfull) == "5"] <- "AIC"
lagfull
```

```
##               delta_lag0    delta_lag1    delta_lag2    delta_lag3
## Estimate   -3.789602e-02 -4.585907e-02 -4.094087e-02 -4.359322e-02
## Std. Error  1.300381e-02  1.286351e-02  1.304292e-02  1.322914e-02
## t value    -2.914225e+00 -3.565049e+00 -3.138934e+00 -3.295243e+00
## Pr(>|t|)    3.747580e-03  4.038252e-04  1.810797e-03  1.064034e-03
## AIC        -2.606707e+03 -2.620834e+03 -2.617477e+03 -2.611271e+03
##               delta_lag4
## Estimate   -4.960502e-02
## Std. Error  1.337032e-02
## t value    -3.710084e+00
## Pr(>|t|)    2.342367e-04
## AIC        -2.609457e+03
```

The table shows that the AIC statistic takes on its smallest value of -0.00262 when there is one lag in the regression (i.e., second column). This means we can proceed with the ADF test with $k = 1$.

Fortunately, the *ur.df()* function is able to automatically choose the number of lags to include based on the minimization of the AIC or a related information criteria. Let's do the ADF with this automatic lag selection. The following output from the *ur.df()* function verifies that one lag is optimal.

```
adfauto <- ur.df(m$lnp, type = "trend", selectlags = "AIC")
summary(adfauto)@testreg$coefficients
```

```
##                  Estimate   Std. Error    t value     Pr(>|t|)
## (Intercept) -5.414483e-04 0.0012531221 -0.4320794 6.658963e-01
## z.lag.1     -4.585907e-02 0.0128635145 -3.5650495 4.038252e-04
## tt           9.757056e-05 0.0000268025  3.6403521 3.047811e-04
## z.diff.lag   2.235063e-01 0.0465664576  4.7997278 2.183436e-06
```

Critical t Selection (Decision Tree)

The previous test results show that with the optimal number of lags included in the model, the t statistic which is used to test the unit root null hypothesis is -6.240. The next step is to choose the critical t value for which this t statistic must be compared. In Section 6.5.1 there was only one critical t value which was included in the ADF test result and so selection of the critical t value was not an issue. The current test requires distinguishing between a stationary series with a deterministic trend and a non-stationary series with a stochastic trend (i.e., a random walk with drift).

It is important to keep in mind that the ADF test result is biased if the model is not correctly specified. A misspecification includes including a trend term in the econometric model when a deterministic trend is not present in the series,

and failing to include a trend term when the underlying stochastic process does have a deterministic trend. For this reason it is necessary to use a multi-stepped structured approach to testing the unit root null hypothesis and arriving at a conclusion regarding whether the series has a deterministic trend versus a stochastic trend.

Figure 1 below is a decision tree for conducting an ADF test when the appropriate structure of the ADF testing model (e.g., with or with the trend term) is uncertain. If the first difference in the time series, Δy_t, possibly contains a trend then the starting point is the left side of the decision tree. In cases such as testing a residual from a regression where it is not possible for Δy_t to contain a time trend then it is acceptable to start at the right side. If in doubt always start on the left side of the decision tree since the algorithm will quickly lead the user to the right side if statistical testing shows there is no trend in Δy_t.

For this particular case the series potentially contains a deterministic trend in Δy_t, and so we will begin at the top left corner of the decision tree.

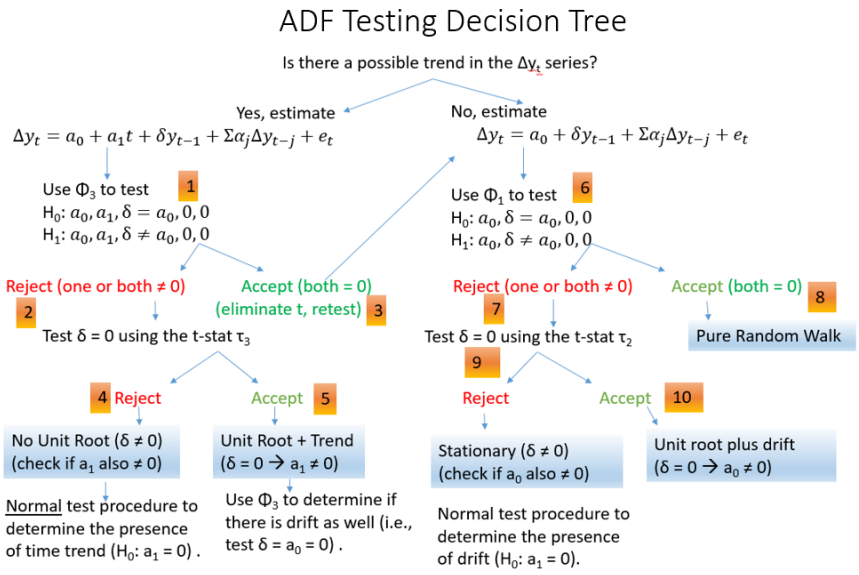

Figure 6.10: Figure 10: Decision Tree for Conducting an Augmented Dickey-Fuller (ADF) test.

The decision tree uses four test statistics, $\{\phi_1, \phi_3\}$ and $\{\tau_2, \tau_3\}$, and the critical values which correspond to these test statistics. The test statistics for $\{\phi_3, \phi_2, \tau_3\}$ are generated when the *trend* option is used in the *ur.df()* function and the test statistics for $\{\phi_1, \tau_2\}$ are generated when the *drift* option is used in the *ur.df()* function. The decision tree does not use the τ_1 test statistic, which is generated when the *none* option is used with the *ur.df()* function. Note that the

decision tree is not used when the *none* option because without a trend or drift component there is only one τ_1 test statistic to be compared with one critical value. The none option is mostly commonly used when testing the stationarity of a regression residual.

The ϕ_1 and ϕ_3 test statistics are used to test joint null hypotheses comprised of the unit root coefficient and the intercept and time trend coefficients. The τ test statistics are used to test the singular null hypothesis comprised of the δ (unit root) coefficient. The left side of the decision tree corresponds to the ϕ_3 joint null and τ_3 singular null with the unit root, drift and trend components. The right side of the decision tree corresponds to the ϕ_1 joint null and the τ_2 singular null with the unit root and drift components.

The augmented Dickey-Fuller (ADF) equation for the left side of the diagram is as follows:

$$\Delta z_t = a_0 + a_1 t + \delta z_{t-1} + \sum_{j=1}^{k} \gamma_j \Delta z_{t-j} + e_t$$

In this case the ϕ_3 test statistic corresponds to the $a_1 = \delta = 0$ null hypothesis and the τ_3 test statistic corresponds to the $\delta = 0$ null hypothesis.

The augmented Dickey-Fuller (ADF) equation for the right side of the diagram is as follows:

$$\Delta z_t = a_0 + \delta z_{t-1} + \sum_{j=1}^{k} \gamma_j \Delta z_{t-j} + e_t$$

In this scenario there is no time trend and the ϕ_1 test statistic corresponds to the $a_0 = \delta = 0$ null hypothesis. Similarly, the τ_2 test statistic corresponds to the $\delta = 0$ null hypothesis

The decision tree tells us to begin by testing the ϕ_3 null hypothesis (see step 1), which is equivalent to jointly assuming there is a unit root and no deterministic trend. We obtain the F statistic and the critical F value from the ϕ_3 column in the ADF testing summary.

```
summary(adfauto)@teststat
```

```
##                tau3     phi2     phi3
## statistic -3.565049 7.014924 6.633251
```

```
adfauto@cval
```

```
##       1pct  5pct 10pct
## tau3 -3.98 -3.42 -3.13
## phi2  6.15  4.71  4.05
## phi3  8.34  6.30  5.36
```

From the above test results we have a ϕ_3 F statistic equal to 6.633 and a ϕ_3 critical F value equal to 6.30 for the 5 percent confidence test. This means we should reject the joint null hypothesis of a unit root ($\delta = 0$) and no significant deterministic time trend ($a_1 = 0$). Rejecting a joint null means that one or both of $\delta \neq 0$ and $a_1 \neq 0$ may be true. This means that additional testing is required in order to fully assess the stationary status of this price series.

We are now at step 2 in the decision tree. We are told to test the $\delta = 0$ hypothesis by conducting the τ_3 hypothesis test. This involves comparing the t statistic for δ with the reported critical t value, which is labeled τ_3. From the testing output above the t statistic is equal to -3.565 and the critical t value is equal to -3.42 for the 5 percent test. We can therefore reject the $\delta = 0$ hypothesis and conclude there is no unit root (i.e., the data is stationary).

We are now at step 4 in the decision tree. We still can't be sure there is a deterministic time trend because the ϕ_3 test told us that one or both of $\delta = 0$ and $a_1 = 0$ do not hold. If instead we were not able to reject $\delta = 0$ then we could verify the presence of a deterministic time trend given the outcome in step 1.

Returning to step 4, because we know the data is stationary it is acceptable to test for the presence of a time trend using normal testing procedures. Specifically, we would look for a p value less than 0.05 for the time trend coefficient when regressing *lnp* on its lags and a time trend. In this particular case it can be shown with normal testing that the time trend is positive and statistically significant. We can now conclude that the producer price of eggs in Alberta is trend stationary.

Is the U.S. CPI Trend Stationary?

Let's use the decision tree in Figure 10 to determine the stationarity status of the U.S. CPI, which is illustrated in Figure 6.1. We begin by using the automatic lag selection feature of the *ur.df()* function.

```
cpitrnd <- ur.df(cpi_indust$CPI, type = "trend", selectlags = "AIC")
summary(cpitrnd)@testreg$coefficients
```

```
##                Estimate  Std. Error   t value     Pr(>|t|)
## (Intercept)  2.508116587 1.248717955  2.008553 4.529531e-02
## z.lag.1     -0.018488750 0.009775960 -1.891246 5.935579e-02
## tt           0.006987165 0.003559709  1.962847 5.039688e-02
## z.diff.lag   0.519457139 0.045714130 11.363164 6.115882e-26
```

Similar to the case of producer egg prices in Alberta, one lag is optimal when conducting the ADF test for U.S. CPI. Now lets look at the null hypothesis testing statistics.

```
cpitrnd <- ur.df(cpi_indust$CPI, type = "trend", selectlags = "AIC")
 summary(cpitrnd)@teststat
```

```
##                tau3     phi2     phi3
## statistic -1.891246 15.58188 2.475847
```

```
cpitrnd@cval
```

```
##       1pct  5pct 10pct
## tau3 -3.98 -3.42 -3.13
## phi2  6.15  4.71  4.05
## phi3  8.34  6.30  5.36
```

As before we start with step 1 and conduct the ϕ_3 hypothesis test. The above statistics shows that the F statistic for the $\delta = 0$ and $a_1 = 0$ joint hypothesis is equal to 2.476, and the critical F value is 6.30 for the 5 percent test. In this case we fail to reject the joint null hypothesis and so we should move to step 3 and conclude that $\delta = 0$ and $a_1 = 0$ both hold. Knowing that $a_1 = 0$ we should eliminate the time trend from the model and conduct a second ADF test (even though we concluded that $\delta = 0$ with this current test we may arrive at different conclusions with the new test which eliminates the time trend).

We are now at step 6 in the decision tree. The ϕ_1 test involves testing the joint null hypothesis that there is a unit root ($\delta = 0$) and no significant drift term in the autoregressive process ($a_0 = 0$). The ADF test and test statistics are as follows.

```
cpidrft <- ur.df(cpi_indust$CPI, type = "drift", selectlags = "AIC")
summary(cpidrft)@testreg$coefficients
```

```
##                 Estimate   Std. Error    t value     Pr(>|t|)
## (Intercept) 0.0697263618 0.1271925407  0.5481954 5.838805e-01
## z.lag.1     0.0006601192 0.0006320726  1.0443724 2.969788e-01
## z.diff.lag  0.5003741255 0.0448361552 11.1600587 3.371689e-25
```

```
summary(cpidrft)@teststat
```

```
##               tau2     phi1
## statistic 1.044372 21.28621
```

```
cpidrft@cval
```

```
##       1pct  5pct 10pct
## tau2 -3.44 -2.87 -2.57
## phi1  6.47  4.61  3.79
```

The test results reveals that the F statistic from the regression is 21.286 and the critical F value is 4.61 at the 5 percent level. We can therefore easily reject this joint null hypothesis and conclude that one or both of $\delta = 0$ and $a_0 = 0$ fail to hold. Thus, we move to step 7 in the decision tree and perform the test named τ_2. The t statistic for the δ coefficient is equal to 1.0444 and the critical t value is -2.87. We clearly fail to reject this hypothesis and so we can conclude there is a unit root in the series. We previously concluded that one or both of $\delta = 0$ and $a_0 = 0$ failed to hold. Since we have established that $\delta = 0$ we can conclude that $a_0 \neq 0$. Consistent with step 10, we can conclude that the U.S. CPI has a unit root (i.e., random walk) with drift.

6.5.3 Testing for Stochastic Trends

Figure 6.5 shows the price of wheat (left panel) and the price of crude oil (right panel). There is no obvious deterministic trend in either series and so it is acceptable to begin the assessment for stationarity at step 6 in the decision tree (see Figure 10). The ADF test for the log of the price of wheat is as follows:

```
whtdrft <- ur.df(log(wht_oil$P_wht), type = "drift", selectlags = "AIC")
summary(whtdrft)@testreg$coefficients
```

```
##                Estimate Std. Error   t value     Pr(>|t|)
## (Intercept)  0.11365408 0.05339461  2.128568 3.393616e-02
## z.lag.1     -0.02197237 0.01043716 -2.105205 3.593281e-02
## z.diff.lag   0.22840186 0.05038272  4.533338 7.802726e-06
```

```
summary(whtdrft)@teststat
```

```
##                tau2    phi1
## statistic -2.105205 2.31287
```

```
whtdrft@cval
```

```
##       1pct  5pct 10pct
## tau2 -3.44 -2.87 -2.57
## phi1  6.47  4.61  3.79
```

These test results reveal that once again a single lag in Δz_t is optimal for the ADF test. Following step 6 in the decision tree we test the ϕ_1 joint hull hypothesis: $\delta = 0$ and $a_0 = 0$. The F statistic from the regression is 2.3129 and the critical F value is 4.61 at the 5 percent level. Thus, it is not possible to reject ϕ_1 and we move to step 8 in the decision tree. Given that $\delta = 0$ and $a_0 = 0$ both hold we can conclude that the log of the price of wheat is a pure random walk (i.e., a unit root with no drift). This outcome is expected given the general pricing pattern of the price of wheat in Figure 6.5.

The stationarity outcome for the log of the price of crude oil is similar. The test of ϕ_1 in step 6 reveals that the F statistic of 2.2154 is well below the critical F value of 4.61 at the 5 percent level. We can therefore conclude that the log of the price of oil is a random walk with no drift.

```
oildrft <- ur.df(log(wht_oil$P_oil), type = "drift", selectlags = "AIC")
summary(oildrft)@testreg$coefficients
```

```
##                Estimate  Std. Error   t value     Pr(>|t|)
## (Intercept)  0.06088997 0.029000498  2.099618 3.642493e-02
## z.lag.1     -0.01587915 0.007763429 -2.045378 4.150944e-02
## z.diff.lag   0.29236263 0.049215141  5.940502 6.450434e-09
```

```
summary(oildrft)@teststat
```

```
##                tau2     phi1
## statistic -2.045378 2.215454
```

```
oildrft@cval
```

```
##       1pct  5pct 10pct
## tau2 -3.44 -2.87 -2.57
## phi1  6.47  4.61  3.79
```

6.6 Low Power of the ADF Test and Alternatives

An important shortcoming of ADF testing is that it has relatively low statistical power when the series is stationary but close to having a unit root. Unfortunately this describes many financial time series including commodity prices. It is useful to briefly review the concept of statistical power to explain the implications of this shortcoming of ADF testing.

A type I error is the probability of rejecting a null hypothesis when it is actually true (i.e., a false positive). A type II error is the failure to reject a null hypothesis which is actually false (i.e., a false negative). The probability of a type I error, which is denoted α, is a measure of statistical significance (e.g., $\alpha = 0.05$ when the confidence level is 95 percent). The probability of a type II error is denoted β. Statistical power, which is the inverse of making a type II error, is measured by $1 - \beta$. With ADF testing, the higher the probability of not rejecting the unit root null hypothesis which is actually false, the lower the power of the test. Lower power with ADF testing means that we will too frequently fail to reject the unit root null hypothesis when the series is in fact stationary.

There are several variations of the ADF test which aim to raise the power of the test of the unit root null hypothesis. In R's URCA package there are five alternative testing procedures:

- ur.ers(): Elliott, Rothenberg and Stock Unit Root Test
- ur.kpss(): Kwiatkowski et al. Unit Root Test
- ur.pp(): Phillips and Perron Unit Root Test
- ur.sp(): Schmidt & Phillips Unit Root Test
- ur.za(): Zivot & Andrews Unit Root Test

Let's re-examine the U.S. CPI test for a unit root with the *ur.df()* function. When the time trend is included the testing results are as follows:

```
cpitrnd <- ur.df(cpi_indust$CPI, type = "trend", selectlags = "AIC")
 summary(cpitrnd)@teststat
```

```
##                tau3     phi2     phi3
## statistic -1.891246 15.58188 2.475847
```

```
cpitrnd@cval
```

```
##       1pct  5pct 10pct
## tau3 -3.98 -3.42 -3.13
## phi2  6.15  4.71  4.05
## phi3  8.34  6.30  5.36
```

Notice the the estimated coefficient on the lagged z_t variable is very close to 0, which means that the estimated root is only slightly less than one. This is precisely the situation where the ADF has relatively low power. Based on the testing statistics we concluded that U.S. CPI follows a random walk with drift. Knowing that this particular test has low power it may be the case that the U.S. CPI is actually trend stationary but the test was not powerful enough to detect this outcome.

The ADF-GLS test, which belongs to the Elliott, Rothenberg and Stock (ERS) family of testing procedures, is more powerful than the regular ADF testing. Without going into detail, the ADF-GLS test works with first-differenced data, which implies that the data is detrended prior to testing. The null hypothesis is that z_t is a random walk, possibly with drift. The alternative hypothesis is that z_t is stationary around a deterministic trend. Unlike the regular ADF test, there is just one test statistic.

The testing statistics for the ADF-GLS test when applied to U.S. CPI is as follows:

```
test <- ur.ers(cpi_indust$CPI, type = "DF-GLS", model = "trend", lag.max = 4 )
summary(test)
```

```
##
## ###############################################
## # Elliot, Rothenberg and Stock Unit Root Test #
## ###############################################
##
## Test of type DF-GLS
## detrending of series with intercept and trend
##
##
## Call:
## lm(formula = dfgls.form, data = data.dfgls)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.83746 -0.18889 -0.00338  0.22438  2.01203
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## yd.lag       -0.017829   0.008675  -2.055 0.040558 *
## yd.diff.lag1  0.617088   0.051556  11.969  < 2e-16 ***
## yd.diff.lag2 -0.202795   0.060628  -3.345 0.000906 ***
## yd.diff.lag3  0.032059   0.060741   0.528 0.597946
## yd.diff.lag4  0.082948   0.052995   1.565 0.118378
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4842 on 374 degrees of freedom
## Multiple R-squared:  0.2947, Adjusted R-squared:  0.2853
## F-statistic: 31.26 on 5 and 374 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -2.0552
```

```
##
## Critical values of DF-GLS are:
##                  1pct  5pct 10pct
## critical values -3.48 -2.89 -2.57
```

The test statistic, which is equal to -2.0552, is less in absolute value than the test's critical value, which is equal to -2.89 at the 5 percent level. Thus, even though the test statistic and critical value are closer together, we still cannot reject the null hypothesis and so continue to conclude that the U.S. CPI is a random walk with drift.

6.7 Summary and Conclusions

Assessing the stationarity properties of a time series should be the starting point for all types of time series analysis. Examining the pattern of the autocorrelation and partial autocorrelation functions in correlograms typically provide strong hints regarding whether the time series is likely to be stationary. The Augmented Dickey-Fuller (ADF) test is the most widely used method for testing for stationarity, largely because it is easy to understand and estimated. Unfortunately, the ADF test has relatively low power when a stationary time series is close to having a unit root. Recent testing procedures have been developed to address this deficiency.

An important focus of this chapter was distinguishing between a series with a deterministic trend and one with a stochastic trend. To address this issue explicitly, we examined two data series that had a similar appearance but which ended up having different testing outcomes. Specifically, we rejected the unit root null hypothesis for the case of Alberta eggs but were not able to reject the unit root null for the case of the U.S. CPI. What this means is that we can include the price of eggs in an econometric model provided that we also include a separate explicit time trend variable. However, we cannot include the U.S. CPI in an econometric model because including an explicit time trend would not make the CPI stationary. One solution is to construct a model which uses the first difference of the CPI rather than the level because the first difference is stationary.

Including non-stationary variables in an econometric model typically results in spurious regression outcomes. Such outcomes are driven by coincidental stochastic trends rather than causal relationships. For this reason conclusions based on spurious regression outcomes can be highly misleading. We saw this when we regressed the price of wheat on the price of oil while not accounting for the fact that both variables are non-stationary. The coincidental stochastic trends made it appear that shocks in the oil market have a large impact on the price of wheat. After eliminating the stochastic trends through first differencing we

discovered that the actual relationship between the price of wheat and the price of oil is rather weak.

Time series often have structural breaks which are caused by sudden, large and non-reversible shocks. Structural breaks can make a time series non-stationary even if the series itself is stationary before and after the break. The procedures for estimating stationary while accounting for structural breaks is imporant but rather complex. The issue of structural breaks is examined in some detail in a later chapter in this textbook.

Additional useful readings

table 4.2 in Enders, Applied Econometric Time Series (4e) https://dokumen.tips/documents/1-unit-root-tests-methods-and-problems-roger-perman-applied-econometrics-lecture.html?page=27

https://stats.stackexchange.com/questions/24072/interpreting-rs-ur-df-dickey-fuller-unit-root-test-results

See https://www.reed.edu/economics/parker/312/tschapters/S13_Ch_1.pdf
See https://cran.r-project.org/web/packages/strucchange/strucchange.pdf

Chapter 7

Structural Breaks

A structural break is an unexpected change in one or more parameters of a stochastic process (e.g., mean, variance or trend). Structural breaks in commodity prices are often associated with a specific policy event such as the signing of a new trade agreement. Failing to account for structural breaks in time series can lead to incorrect conclusions. For example, if the trend in a price series switches from a negative slope to a positive slope due to a policy-induced structural break, then a regression may produce a zero-slope outcome. Similarly, the presence of one or more structural breaks may cause a researcher to conclude that a time series has a unit root when one doesn't actually exist. There has been a lot of debate in the literature regarding whether macro economic variables such as GDP are stationary with breaks, or are inherently non-stationary.

In this chapter we restrict our attention to structural breaks in the mean of a time series rather than both the mean and a trend. The primary data set we will analyze is the gross margin earned by U.S. electricity producers, which is expected to be free of trends. In this case there are various reasons for the emergence of a structural break such as the introduction of a carbon tax on the natural gas which is used to produce the electricity. Conversely, a break may result with a change in a key regulation which governs electricity pricing. If a researcher has information about the reason for the break then structural break analysis can be used to assess the validity of that reason. More commonly, researchers do not have information about the timing of the break and so the timing of the break is estimated and the significance of the break is tested simultaneously.

Before analyzing the main electricity margin data, we will spend some time with a gasoline pricing data set. This data set has some obvious breaks and the breaks can be traced back to the introduction of new taxes. This data is used to motivate the concept of a structural break by using a t test for the equality of the mean gasoline price before and after the break. The same test is conducted through an analysis of variance (ANOVA) in order to motivate the wide-spread

use of the F test in structural break analysis. In this chapter we will estimate an autoregressive model with one lag (i.e., an AR1), in which case there are two parameters which are allowed to break. With multiple breaks and a known break date we are able to use the Chow F test procedure to test for the null hypothesis of no structural break.

The bulk of this chapter uses the electricity margin data to analyze structural breaks when the break date is unknown. In the case of a single break, we will use a *supF* procedure. This procedure involves calculating a F statistic for every possible break date, and then using the maximum value from this set of F statistics to both identify the most likely date of the break and test for its statistical significance. Allowing for multiple breaks is much more complicated because the number of different break date combinations is typically excessively high. To address this problem, a dynamic programming procedure is used to identify the break points which lead to the lowest possible residual sum of squares (RSS). The BIC criterion is then used to select the optimal number of breaks to including for significance testing.

A common use of structural break analysis is conducting a stationarity test which allows for a structural break as part of the testing procedure. As previously noted, failing to account for structural breaks can result in incorrectly failing to reject the unit root null. Differencing a time series as a means to achieve stationary is an incorrect procedure if structural breaks are not accounted for. There are many sophisticated procedures for testing for a unit root while allowing for one or more structural breaks. We will analyze the electricity margin data with one of the earliest methods for testing for a unit root while allowing for a single structural break.

The next section is used to introduce structural break analysis more generally and to briefly review the literature. The preliminary analysis which utilizes the gasoline pricing data is conducted in Section 7.2.The focus of Section 7.3 is the *supF* test with a single break. The case of multiple structural breaks is analyzed in Section 7.4. Section 7.5 concludes the formal analysis with a detailed examination of unit root testing in the presence of a structural break. A summary of the chapter is provided in 7.6

7.1 Introduction

Structural breaks in an economic time series is often the result of a government policy such as a new or revised tax. The price of gasoline is likely to have structural breaks because in most jurisdictions gasoline has several taxes, some of which are revised over time. For example, in 2022 drivers in Vancouver, British Columbia paid about 70 cents per liter in combined gasoline taxes. British Columbia introduced a provincial carbon tax in 2008 and by 2012 that tax was 11 cents per liter. In addition, the tax allocated to transit steadily increased to its current 2022 level of about 18.5 cents per liter. In contrast, drivers in

Winnipeg, Manitoba only began to pay a carbon tax in 2020, and there is no dedicated gasoline tax for transit. This suggests the the price premium paid by Vancouver drivers relative to Winnipeg drivers will have "jumped" upward when the carbon and transit taxes were introduced in British Columbia. Of interest is whether or not this suspected structural break can be observed in the data.

After loading the packages required for this chapter, the price premium data can be read in from a saved *RDS* file.

```
#rm(list = ls())
pacman::p_load(tidyverse, here, lubridate, tsibble, reshape2, gridExtra, urca, strucchange,aod)

gas_city <- readRDS(here("data/ch6","gas_city.RDS"))
```

Figure 7.1 shows that indeed the Vancouver - Winnipeg price premium jumped upward during the 2008 to 2012 period in response to a provincial carbon tax which was introduced in British Columbia but not in Manitoba. Similarly, the increase in the metro Vancouver transit tax, which was 18.5 cents per liter in 2022, caused the price premium to increase again after 2015. The premium shrunk beginning in 2020 with the introduction of the federal carbon tax in Manitoba.

```
ggplot(gas_city,aes(x=month,y=premium) ) + geom_line() +
  labs(y = "Cents per Liter", x="")
```

Without knowing the markets which generated the data in Figure 7.1 it may be tempting to view the problem as a structural break in both the trend line as well as the intercept. Given the nature of gasoline markets in Canada and the scope for arbitrage we don't expect the price premium to be trending upward over time. Another possibility is that the structural break is gradual rather than abrupt. There are models which allow for gradual rather than abrupt structural change but these models are too advanced for this textbook. If trends are ruled out then a visual assessment of Figure 7.1 suggests an approximate mean price premium of 10 cents a liter before 2009, 20 cents a liter between 2010 and 2015 and 30 cents a liter after 2015. Structural break analysis is important because it replaces this type of subject assessment with estimates from rigorous statistical analysis.

For this particular case of differential gasoline taxes we have a fairly good idea of when the structural breaks occur. If the break date is known with certainty then we can use the test for a structural break proposed by Chow [1960]. The widely-used Chow test for a structural break is an F test on the null hypothesis that the estimated coefficients on the pre and post break segments are equal. The null is rejected if the residual sum of squares of the restricted model where no break

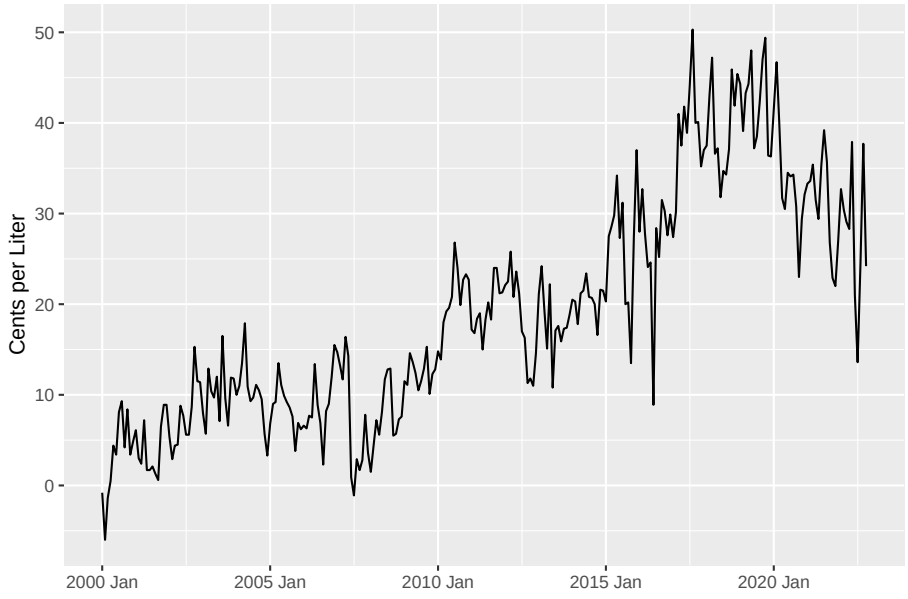

Figure 7.1: Monthly Price Premium for Gasoline in Vancouver versus Winnipeg

is allowed is sufficiently above the residual sum of squares of the unrestricted model where the break is allowed.

In most models we do not know the precise date of the break. Quandt [1960] was one of the first to tackle this problem. Quandt proposed to calculate a Chow F statistic for every possible break, and then estimate the date of the break as that which corresponds to the maximum value within the set of F statistics. This is a sensible approach but for many years the distribution of the Quandt statistic (i.e., maximum F) was unknown because the break date parameter appears in the alternative hypothesis but not in the null hypothesis. Andrews [1993] and Andrews and Ploberger [1994] eventually identified the asymptotic distribution of the test statistic under the null distribution, and with this distribution in hand a table of critical values was created. These critical values are typically well above the critical values for the Chow test, which means that rejecting the null hypothesis of no structural change is relatively less likely. Hansen [1997] made testing easier by providing asymptotic P Values for the Quandt test statistic.

In Figure 7.1 there is likely more than one structural break, due to the differential timing of the carbon tax implementation, and the extensive use of a gasoline transit tax in Vancouver but not in Winnipeg. Bai and Perron [1998] used a dynamic programming procedure to find the break points which minimize the residual sum of squares assuming m break points. The BIC criterion together

with the derived asymptotic critical values can then be used to choose the optimal number of breaks and the statistical significance of these breaks. Liao and Suen [2006] and Jin and Miljkovic [2010] date multiple structural breaks in global oil prices and relative farm prices, respectively.

If the time series is modeled in levels then the slope of the trend line is the mean rate of growth of the variable in question. This means that a structural break in the slope of the trend line is equivalent to a structural break in the mean growth rate. Ben-David and Papell [1998] define a "slow-down" as a downward structural break in the trend line. They estimate that the majority of the 74 countries analyzed at some point had a slowdown in their per capital GDP. Using tests for multiple structural breaks in the trend line, Benati [2007] identify a deceleration in labour productivity beginning in the early 1970s and a strong resurgence in growth beginning in the mid-1990s. More generally, Stock and Watson [1996] conclude that a large number of macro economic time series have significant structural breaks.

Structural breaks have important implications for detecting a unit root in a time series. Recall from Chapter 5 that Nelson and Plosser [1982a] showed that many macro economic time series have a unit root. Perron [1989] noted that a break in a trend produces a serial correlation which is similar to a unit root. When accounting for a single structural break, Perron [1989] was able to reject the unit root on many of the time series which were designated as having a unit root by Nelson and Plosser [1982a]. Zivot and Andrews [1992] showed that the results of Perron [1989] fail to hold if the timing of the break is endogenized. This revised testing procedure has a dedicated function (*ur.za()*) in the *urca* package. Lee and Strazicich [2003] built on the Zivot and Andrews [1992] method by allowing for two structural breaks in the unit root test, and also by constructing a test statistic within which the break date parameters are present in both the null and alternative hypothesis.

As was shown in Chapter 5, time series which are integrated of order 1 can be made stationary by taking the first difference. However, if failure to reject the unit root is due to structural breaks rather than a true unit root, then taking a first difference is not an appropriate strategy. What is needed instead is the appropriate use of dummy variables to accommodate the break points. Wang and Tomek [2007] note that based on economic theory agricultural commodity prices should be stationary. They show that by accounting for structural breaks, it is possible to reject the unit root null for many of the commodity prices which were originally believed to be non-stationary. Maslyuk and Smyth [2008] use weekly data on WTI and Brent crude oil spot and futures prices to test for unit roots while allowing for one and two structural breaks.

7.2 Preliminaries

The remainder of this chapter will examine structural breaks in the "spark spread" in the U.S. Pacific Northwest. The spark spread is the gross margin earned by a plant which produces electricity from natural gas. It is calculated by adjusting the price of natural gas by a heat-rate conversion factor and then subtracting this adjusted gas price from the price of electricity. The spark spread is in unit of dollars per megawatt hour (MWh). The electricity prices are *Mid-C Hub* wholesale daily prices from the Intercontinental Exchange (ICE), which have been aggregated to monthly averages and then seasonally adjusted. The natural gas prices are Henry Hub monthly averages from the Federal Reserve of Economic Data (FRED), which were also seasonally adjusted. The Henry Hub is near the Gulf Coast, which means that prices at this location trade at a premium. Using daily natural gas prices for the years 2015, 2016 and 2017 (these are the only years for which this data is available), the Pacific Northwest price discount for natural gas was estimated to equal about 5.5 percent.

7.2.1 Data Visualization and t test

We begin by reading in the data, which was previously pre-cleaned and stored as *power.RDS*.

```
power <- readRDS(here("data/ch6","power.RDS"))
```

The left and right plots in Figure 7.2 respectively show the seasonally-adjusted, monthly average wholesale prices of electricity in the Pacific Northwest (PNW) and natural gas at the Gulf Coast. The more volatility electricity price reflects the fact that electricity is a fully non-storable commodity. Both series show the surge leading up to the 2007-2008 great recession and the dramatic crash in prices with the onset of the recession. Notice that electricity prices but not natural gas prices surged again in 2019/2020. In general, these features reflect the relatively high degree of regulation in U.S. electricity markets and the relatively low degree of regulation in U.S. natural gas markets.

```
plotA <- ggplot(power,aes(x=month,y=elect_sa) ) + geom_line() +
  labs(y = "$/MWh", x="")

plotB <- ggplot(power,aes(x=month,y=gas_sa)) + geom_line() +
  labs(y = "$/mmBTU", x="")

grid.arrange(plotA, plotB, ncol = 2)
```

Figure 7.3 below shows the calculated spark spread when producing electricity from natural gas. The calculated spread uses an EIA-recommended heat rate of

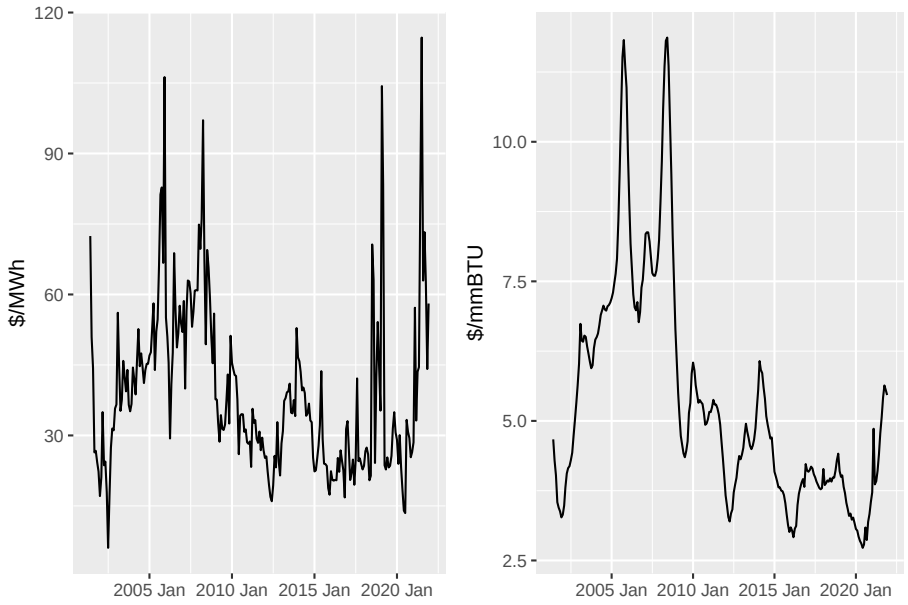

Figure 7.2: Monthly PNW Electricity Prices and Gulf Coast Natural Gas Prices

7000 BTU per kilowatt hour (KWh) of electricity generated. This is equivalent to 7 million BTU per megawatt hour (MWh) of electricity. Natural gas is measured in millions of BTU and so when calculating the spark spread the price of natural gas was multiplied by 7 to convert the gas price to a MWh equivalent. The calculated spread also accounts for the approximate 5 percent lower natural gas price on the U.S. west coast versus the U.S. Gulf Coast.

```
ggplot(power,aes(x=month,y=SprkSprd) ) + geom_line() +
  labs(y = "$/MWh", x="")
```

Figure 7.3 shows that the spark spread is highly volatile, even though the underlying prices have been seasonally adjusted. This spread must cover the other variable and fixed operating costs of the plant. Negative values for the spread reflect times when the plants are operating at a loss. Plants which have a heat rate of less than 7 will be relatively more profitable.

Of interest in this chapter are potential structural breaks in the spark spread. Figure 7.3 shows that the mean spread may have jumped to a higher level beginning in early 2018. Let's "guess" that the structural break happened in March of 2018. For now we will assume that this break date is known with certainty. The bulk of this chapter is devoted to the more realistic case presented later where the specific break date is unknown.

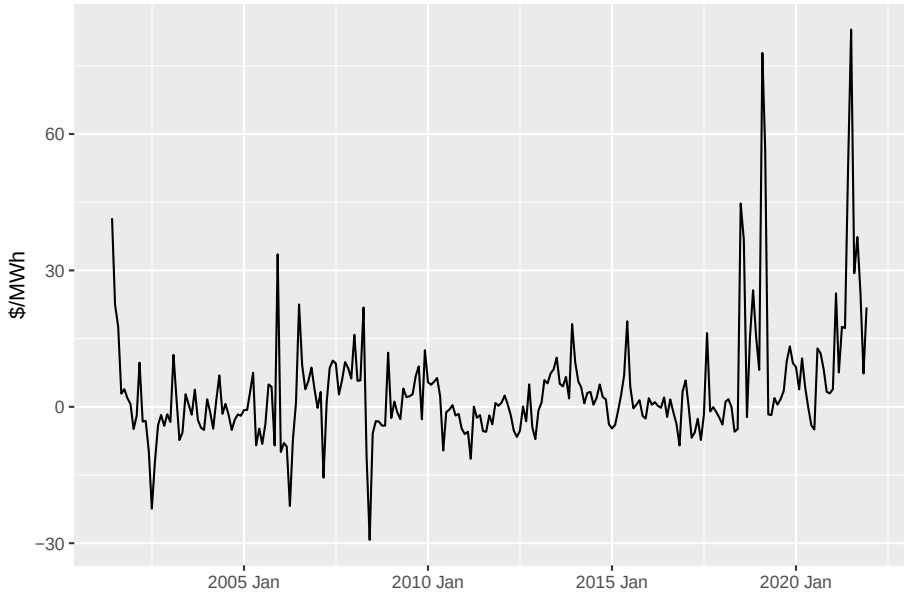

Figure 7.3: Calculate Monthly Spark Spread in the Pacific Northwest

7.2.2 t and F tests

Before turning to the formal testing of structural breaks, let's use the popular t test to assess the null hypothesis that the mean spark spreads before and after the March 2018 break are equal. To keep our analysis as simple as possible, let's use the same number of observations before and after the break. This means choosing the start date of the segment 1 sample (May of 2014) in order to match the 46 months in the segment 2 sample. We will also assume the variance of the spark spread is the same before and after the break. We can see from Figure 7.3 that this is not the case, but we will deal with the issue of heteroskedasticity later in this chapter.

```
seg1 <- power %>% filter_index("2014-05"~ "2018-02")
seg2 <- power %>% filter_index("2018-03"~ . )
```

If we assume that the spark spread is normally distributed then we can use a standard two-sample test for the null hypothesis of zero difference in means. The formula for the t statistic of this test can be expressed as

$$t_{stat} = \frac{\bar{x}_1 - \bar{x}_2}{\left(\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}\right)} \tag{7.1}$$

In equation (7.1), $\bar{x}_1$ is the mean value of the segment before the March 2018 break and $\bar{x}_2$ is the mean value for the post March 2018 segment. The s_i standard deviation variables are defined in a similar way, and n_i is the sample size for segment i (given our simplification, $n_1 = n_2 = 46$). Let's calculate this t statistic now.

```
t_denom <- (sd(seg1$SprkSprd)^2/nrow(seg1)+sd(seg2$SprkSprd)^2/nrow(seg2))^0.5
t_num <- mean(seg1$SprkSprd)-mean(seg2$SprkSprd)
t_num/t_denom
```

```
## [1] -4.705618
```

A much easier way to calculate this same t-statistic is to use the *t.test()* function. The *t.test()* function also gives the p value for the null hypothesis of equal means. In the results below the p value is very small and as such the null hypothesis of equal means on either side of March 2018 can be rejected. This is our first evidence that there may be a structural break in this data.

```
t_stat2 <- t.test(seg1$SprkSprd, seg2$SprkSprd, var.equal=TRUE)
t_stat2$statistic
```

```
##         t
## -4.705618
```

```
t_stat2$p.value
```

```
## [1] 9.112082e-06
```

An alternative way to test the null hypothesis of equal means before and after the break is to note that the ratio of the between-group variability to the within-group variability follows an F-distribution when the null hypothesis is true. This approach is the widely-used analysis of variance (ANOVA) procedure. The formula for the F statistic is relatively simple for this case of equal sample sizes.

$$F_{stat} = n_i \frac{S^2_{mean}}{0.5(S^2_1 + S^2_2)} \tag{7.2}$$

In equation (7.2), S^2_{mean} is the standard deviation of the pair of means for segments 1 and 2 (i.e., the between group variation). We can calculate this F statistic as follows.

```
mean1 <- mean(seg1$SprkSprd)
mean2 <- mean(seg2$SprkSprd)
var1 <- var(seg1$SprkSprd)
var2 <- var(seg2$SprkSprd)
var_mean <- var(c(mean1,mean2))
avg_var <- mean(c(var1,var2))
F <- nrow(seg1)*var_mean/avg_var
F
```

```
## [1] 22.14284
```

This F statistic follows a F distribution with 1 degree of freedom for the numerator and *n-1* degree of freedom for the denominator. This allows a critical F value to be calculated and together the F statistic and the critical F value can be used to derive the p value for the hypothesis test. We can use the ANOVA function, *aov()*, to generate this p value. The *aov()* function is equivalent to a linear regression with a dummy variable which takes on a value of zero before the break and one after the break.

```
power2 <- power %>%
  filter_index("2014-05"~.) %>%
  mutate(brk = ifelse(month>=yearmonth(as.Date("2018-03-01")),1,0))
anova <- aov(SprkSprd~brk,data=power2)
summary(anova)
```

```
##             Df Sum Sq Mean Sq F value   Pr(>F)
## brk          1   4929    4929   22.14 9.11e-06 ***
## Residuals   90  20036     223
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

As expected, the p value for the F test of the no-break null hypothesis is equal to the p value of the previous t test. This is because the two sample t test and an ANOVA with two groups are equivalent procedures. In fact, if you take the square root of the above F statistic you will obtain the previously-derived t statistic.

To conclude this section note that in equation (7.2) there is an alternative way to measure the variance of the means for segments 1 and 2, as measured by S^1_{mean}. Let RSS_i denote the residual sum of squares (RSS) of segment i and let RSS_{12} denote the residual sum of squares (RSS) of the two segments combined. From the definition of variance it follows that $RSS_i = (n_i - 1)S_i^2$. The following relationship holds: $n_i S^2_{mean} = RSS_{12} - RSS_1 - RSS_2$. In words, the variance of the means of the two segments (multiplied by n_i) can be measured as the

difference in the RSS for both segments combined and the sum of the two individual RSS measures. As will be shown in the next section, this alternative interpretation is central in the formal testing for structural breaks.

7.3 Single Break

This section begins our formal analysis of structural breaks. We begin with the case of a single break, first when the break date is known with certainty and then when the break date is unknown. Section 7.4 considers the case of multiple breaks with uncertain timing.

For this section and the remainder of this chapter assume that the spark spread follows an autoregressive stochastic process with one lag (i.e., an AR1). One lag may not be sufficient to fully eliminate the autocorrelation in this variable. Later in the analysis a robust covariance matrix is used during estimation and testing to account for both residual autocorrelation and heteroskedasticity.

Let y_t denote the spark spread in month t. In the absence of any structural break, the AR1 model can be expressed as

$$y_t = a + by_{t-1} + e_t \qquad t = 1, 2, ..., T \tag{7.3}$$

With one structural break at date t_B the AR1 can be written as

$$y_t = \begin{cases} a_1 + b_1 y_{t-1} + e_t (t = 1, 2, ...t_b - 1) \\ a_2 + b_2 y_{t-1} + e_t (t = t_b, t_b + 2, ...T) \end{cases} \tag{7.4}$$

An alternative way to specify the model with one structural break is to add a dummy variable to the standard model. Specifically, let the dummy d_t take on a value of 0 for all months before the structural break and a value of 1 for all months including and after the structural break. In this case, the AR1 model with one structural break can be expressed as.

$$y_t = a + by_{t-1} + \delta_1 d_t + \delta_2 d_t y_{t-1} + v_t \tag{7.5}$$

There are two ways to express the null hypothesis of no structural break at date t_B. With reference to equation (7.4) the null hypothesis is $a_1 = a_2$ and $b_1 = b_2$. The alternative hypothesis is that there is a structural break with $a_1 \neq a_2$ or $b_1 \neq b_2$ or both $a_1 \neq a_2$ and $b_1 \neq b_2$. With reference to the dummy variable model which is given by equation (7.5), the null hypothesis of no structural break is $\delta_1 = \delta_2 = 0$. The alternative hypothesis is $\delta_1 \neq 0$ or $\delta_2 \neq 0$ or both $\delta_1 \neq 0$ and $\delta_2 \neq 0$.

7.3.1 Chow Test

To test the null hypothesis of no structural break when the break date is known with certainty we use the Chow F test. One version of the Chow F test corresponds to the $a_1 = a_2$ and $b_1 = b_2$ null hypothesis and the second version corresponds to the $\delta_1 = \delta_2 = 0$ null hypothesis. Of course, the test result outcomes are same for both versions. With $k = 2$ coefficients to be estimated and sample sizes for the pre-break and post-break segments equal to n_1 and n_2, respectively, expressions for the two Chow F statistics are as follows.

$$F_a = \frac{(RSS_f - RSS_1 - RSS_2)/k}{(RSS_1 + RSS_2)/(n_1 + n_2 - 2k)} \tag{7.6}$$

and

$$F_b = \frac{(RSS_r - RSS_u)/k}{RSS_u/(n - 2k)} \tag{7.7}$$

For the first version of the Chow test, RSS_1 and RSS_2 are the residual sum of squares from a regression involving the top and bottom half of equation (7.4), respectively, and RSS_f is the residual sum of squares from a regression involving equation (7.3). For the second version of the Chow test, RSS_u is the residual sum of squares from an equation (7.5) regression and RSS_r is the residual sum of squares from an equation (7.3) regression. The f subscript denotes the "full model", the u subscript denotes the unrestricted model and the r subscript denotes the restricted model. It follows that $RSS_f = RSS_r$ since they are the residual sum of squares from the same regression. As well, $RSS_u = RSS_1 + RSS_2$ since estimating the dummy variable model is equivalent to individually estimating the segment 1 and segment 2 models and then adding the RSS outcomes.

It should be clear from equations (7.6) and (7.7) that a Chow test is a F test of linear restrictions on the coefficients of a standard regression model. The Chow test which is given by equations (7.4) and (7.6) is the most popular way to specify the Chow procedure. An important advantage of the second (dummy variable) method is that the estimated model contains potentially useful decomposition information about the break. For example, if the estimated coefficient on the d_t variable is significant and the estimated coefficient on the $d_t y_{t-1}$ variable is not significant, then we will know that the structural break consists exclusively of a break in the mean of the data series. Based on the results from Section 7.2 keep in mind that in the absence of the autoregressive component of the model the Chow test reduces to a standard two-sample t test or two group ANOVA F test (i.e., the p values are identical).

Let's derive the two different versions of the Chow F statistic for the null hypothesis of no structural break in the spark spread in March of 2018. We begin

by constructing data sets which correspond to the pre-break and post-break segments.

```
power <- power %>%
  mutate(L.SprkSprd=lag(SprkSprd),
         brk = ifelse(month>=yearmonth(as.Date("2018-03-01")),1,0)) %>%
  slice(-1)
seg1 <- power %>% filter_index(. ~ "2018-02")
seg2 <- power %>% filter_index("2018-03"~.)
n <- nrow(seg1)+nrow(seg2)
```

We can now use the estimated outcomes of equations (7.3) and (7.4) to calculate values for RSS_1, RSS_2 and RSS_f. These RSS outcomes together with equation (7.6) are used to derive the Chow F statistic for the first method of the Chow test.

```
lmrss1 <- lm(SprkSprd~L.SprkSprd,data=seg1)
rss1 <- sum(lmrss1$residuals^2)
rss1
```

```
## [1] 9310.994
```

```
lmrss2 <- lm(SprkSprd~L.SprkSprd,data=seg2)
rss2 <- sum(lmrss2$residuals^2)
rss2
```

```
## [1] 14945.5
```

```
lmrssf <- lm(SprkSprd~L.SprkSprd,data=power)
rssf <- sum(lmrssf$residuals^2)
rssf
```

```
## [1] 26618.64
```

```
F_a <- 0.5*(rssf-rss1-rss2)/((rss1+rss2)/(n-4))
F_a
```

```
## [1] 11.78324
```

Let's repeat this procedure for the second Chow method but this time we will obtain expressions for RSS_u and RSS_r from equations (7.3) and (7.5), and then use these RSS expressions to calculate the Chow F statistic as given by equation (7.7). As expected, the calculated F statistic from this method is the same as that from the first method.

```
lmrssu <- lm(SprkSprd~L.SprkSprd+brk+brk*L.SprkSprd,data=power)
rssu <- sum(lmrssu$residuals^2)
rssr <- rssf
F_b <- 0.5*(rssr-rssu)/(rssu/(n-4))
F_b
```

```
## [1] 11.78324
```

To test the null hypothesis of no structural break in March of 2018 we need the critical value for a $F_{0.05}$ statistic with 2 and $n-2$ degrees of freedom. We can also calculate the corresponding p value.

```
qf<-qf(0.05, df1=2, df2=n-2, lower.tail=FALSE)
qf
```

```
## [1] 3.032816
```

```
pf <- pf(F_a, df1=2, df2=n-2, lower.tail = FALSE)
pf
```

```
## [1] 1.302843e-05
```

```
# http://users.stat.umn.edu/~sandy/courses/8311/handouts/ch06.pdf
```

The critical F value (3.033) is below our calculated F statistic (11.783) and so we can reject the null hypothesis of no structural break assuming $\alpha = 0.05$. The associated p value (1.3×10^{-5}) shows that our evidence for rejection of the no break null hypothesis is moderately strong.

To conclude this section let's calculate the Chow F statistic using the *sctest()* function from the *strucchange* package.

```
rowbreak <- which(power$month == yearmonth(as.Date("2018-03-01")))
sctest(SprkSprd ~ L.SprkSprd, data=power, type = "Chow", point = rowbreak-1)
```

```
##
##  Chow test
##
## data:  SprkSprd ~ L.SprkSprd
## F = 11.783, p-value = 1.308e-05
```

7.3.2 Quandt Statistic and supF

Let's now turn to the more realistic case where the break date is not known. It is tempting to treat the problem as first estimating the break date in stage one and then testing for the significance of the break in stage two. It is clear from the literature (e.g., Hansen [2001]) that this approach is not appropriate. Instead, the break date and the test statistic for estimating the no-break null hypothesis must be derived together.

Quandt [1960] was the first to tackle the problem of testing for a structural break with an unknown break date. He calculated a Chow F statistic for all feasible break points and then designated the date with the highest F statistic as the most likely break date in the time series. This maximum value of the F statistic is commonly referred to as the Quandt statistic. However, the Quandt statistic could not be used to test the no-break null hypothesis test because the critical values for this statistic were unknown. Andrews [1993] and Andrews and Ploberger [1994] were the first to solve this problem through use of asymptotic theory.

In order to understand how the Quandt problem was eventually resolved it is necessary to digress and discuss some key asymptotic results. First, a F distribution is well approximated by a Chi square distribution when the number of sample observations is large. Specifically, if $\tilde{F}$ is a F distribution with k and $n-2k$ degrees of freedom, then as n becomes very large the test statistic $k\tilde{F}$ converges to a Chi square distribution with $n-2k$ degrees of freedom.

Second, the F test statistic is a special case of a likelihood ratio. This is because if we take the ratio of the maximized likelihood functions, one corresponding to the unrestricted model and the other corresponding to the restricted model, most of the terms cancel and we are left with an expression which can be written as a monotone function of the F statistic. In large samples the likelihood ratio is distributed as a Chi square distribution, which is an expected result given the previously-discussed relationship between the F distribution and the Chi square distribution. Having a test statistic which is distributed as a Chi square is needed in order to derive asymptotic testing results.

Let $F_n(t_B)$ be the likelihood ratio test statistic for the null hypothesis of no structural break when the break date is t_B. Let t_L and t_H be the first and last feasible breakpoints in the data (these are needed to ensure a minimum length for each segment). In R's *strucchange* package the default settings are $t_L = 0.15n$ and $t_H = 0.85n$. We can now define the *supF* statistic as

$$SupF_n = \sup_{t_L \leq t_B \leq t_L} F_n(t_B) \tag{7.8}$$

For the purpose of this chapter, the supremum (i.e., *sup*) operator in equation (7.8) can be interpretted as the maximum value of $F_n(t_B)$ out of the set

$F(t_L), F(t_L+1), ..., F(t_H)$. Critical values for the *supF* test statistic and the corresponding p values are used to test the no-break null hypothesis.

7.3.3 Fstats Collection

As noted in the previous section, in large samples the likelihood ratio test statistic and the F statistic from the Chow test converge toward the Chi Square distribution. We will therefore assume that the $F_n(t_B)$ function in the *supF* test is the Chi square test statistic conditioned on a structural break at date t_B. Recall that $k\tilde{F}$ converges toward a Chi square distribution with $n-2k$ degrees of freedom. Given that $k=2$ we should expect the Chi square test statistic to be double the size of the F test statistic. This is indeed the case because the Chi square test statistic with a break at date t_B can be expressed as

$$F_c(t_B) = \frac{(RSS_r(t_B) - RSS_u(t_B))}{(RSS_u(t_B))/(n_1+n_2-2k)} \tag{7.9}$$

If we calculate $F_c(t_B)$ for all $t_B \in [t_l, t_H]$ then we have a collection of Chi square statistics, which we will call *Fstats*. The *supF* test statistic (equivalent to the Quandt statistic in this particular case) is the maximum value within the *Fstats* set.

In order to generate the set of *Fstats* it is useful to work with matrices rather than data frames. Let's set the first break at 24 months from the start (i.e., $t_L = 24$) and the last break 24 months from the end (i.e.,. $t_H = n-24$).

```
y <- as.matrix(power[,4])
n <- length(y) - 1
X <- cbind(matrix(1,n+1,1),lag(y))
y <- y[-1]
X <- X[-1,]
t_L <- 24
t_H <- n - 24
span <- (t_L:t_H)
np <- length(span)
```

The next step is calculate the RSS for each break date between t_L and t_H. Looping consists of incrementing the t_b break variable through all values in the *span* vector. For each loop, the *lm.fit()* function will regress the data in the y matrix on the data in the X matrix, first for months $1:t_B$ (this is segment 1) and then for months $(t_B+1):n$ (this is segment 2). For each segment the residuals are recovered and combined into a single vector of length n. The sum of the square of these n residuals is the single RSS measure which is stored for that particular loop. When the looping is complete the RSS vector will be filled with one RSS value for each $t_B \in [t_L, t_H]$.

```
RSS <- rep(0,np)
for(i in 1:np)
{
  X1 <- as.matrix(X[(1:span[i]),])
  X2 <- as.matrix(X[((span[i]+1):n),])
  m1 <- lm.fit(X1,y[1:span[i]])
  m2 <- lm.fit(X2,y[((span[i]+1):n)])
  u <- c(m1$residuals, m2$residuals)
  RSS[i] <- sum(u^2)
}
```

The final step in calculating the distribution of *Fstats* is to calculate the RSS for the unrestricted model and then use equation (7.9) to generate the desired F test statistics.

```
m <- lm.fit(X,y)
rss_m <- sum(m$residuals^2)
RSS_r <- rep(rss_m,np)
RSS_u <- RSS
Fstat <- (RSS_r-RSS_u)/(RSS_u/(n-4))
```

We can now add the full set of saved *RSS* values to the *data1* data frame and then plot these values in order to visually identify the optimal break date. Figure 7.4 clearly identifies the estimated break point as occurring in early 2018. This result is expected based on what we learned by examining the initial data in Figures 7.2 and 7.3. Mid 2018 is about the time when the spark spread turned upward.

```
data1 <- power[(t_L+1):(t_H+1),1]  # add 1 due to lag truncation

Fstat_tib <- tibble(Fstat=Fstat)
data1 <- bind_cols(data1,Fstat_tib)

ggplot(data1, aes(month, Fstat)) +
  geom_line() +
  labs(y = "Chi Square Statistic", x="Break Date")
```

Let's ask R to choose the exact date at which *Fstat* is maximized. Note that it is necessary to add $t_L - 1$ to the optimal date because dates 1 through $t_B - 1$ were excluded from the evaluation. The calculations below show that the precise break date is June of 2018. This optimized date is reasonably close to the March 2018 date which we assumed in Section 7.2.

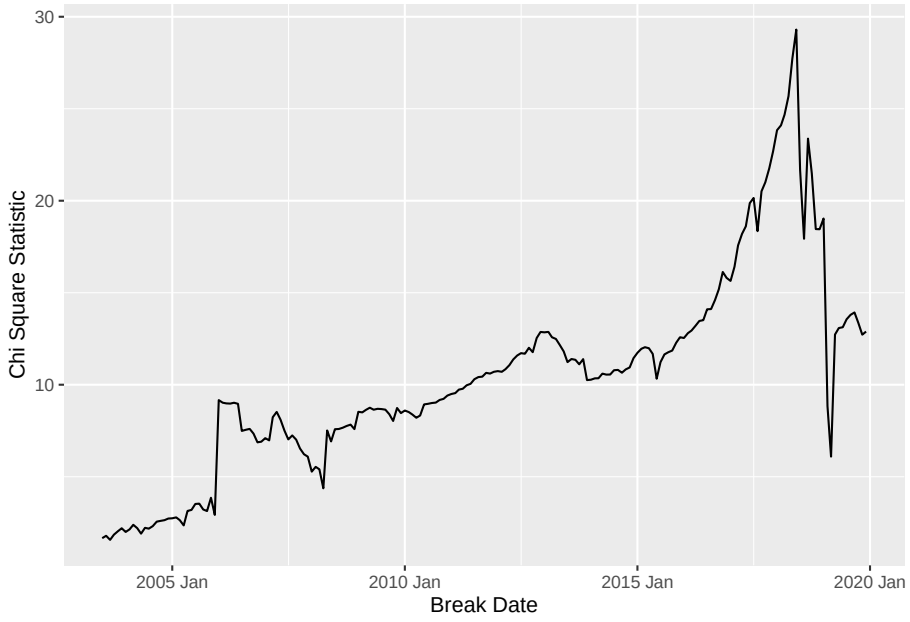

Figure 7.4: Chi Square Test Statistic as a Function of Break Date

```
max_fstat = max(Fstat,na.rm = TRUE)
max_fstat
```

```
## [1] 29.30272
```

```
est_break = which(Fstat==max_fstat) + t_L
est_break
```

```
## [1] 204
```

```
est_break_date <- data1[est_break-t_L,1]
#est_Break_date <- ymd("1997-02-01") %m+% months(est_break)
est_break_date
```

```
## # A tsibble: 1 x 1 [1M]
##      month
##      <mth>
## 1 2018 Jun
```

To conclude this section let's use the *Fstats()* function in the *Strucchange* package to generate the set of *Fstats* which are plotted in Figure 7.4. To confirm

that the outcomes match, the difference in the outcomes are shown to equal zero.

```
Fstat2 <- Fstats(y~X-1, from = t_L, to = t_H)
head(Fstat2$Fstats-data1$Fstat)
```

```
## Time Series:
## Start = c(0, 25)
## End = c(0, 30)
## Frequency = 245
## [1] 0 0 0 0 0 0
```

7.3.4 Testing the No-Break Null

In order to test the no structural break null hypothesis it is necessary to add a boundary to Figure 7.4. Regions for which the *Fstats* plot rises above the boundary are dates for which the null hypothesis of no structural break can be rejected. We are particularly interested if the *supF* statistic, which is the maximum height of the graph, is above the boundary because this will imply that the no break null hypothesis can be rejected at the optimal break date. As noted in the Introduction, Andrews [1993] and Andrews and Ploberger [1994] derived the boundary for the *supF* test.

It is not possible to explicitly calculate the "Andrews" boundary and so we will use the *boundary()* function from the *strucchange* package to generate the desired values.

```
boundary_supF <- boundary(Fstat2,alpha = 0.05)
head(boundary_supF)
```

```
## Time Series:
## Start = c(0, 25)
## End = c(0, 30)
## Frequency = 245
## [1] 12.10482 12.10482 12.10482 12.10482 12.10482 12.10482
```

The *Fstats* and the boundary can now be converted into tibbles and plotted. It is clear from Figure 7.5 that the no-break null can be rejected, beginning in early 2016 and ending in late 2018.

```
supF_tib <- tibble(boundary = boundary_supF)
data1 <- bind_cols(data1,supF_tib)

plot_Fstat <- melt(data1, id.vars = "month", variable.name = "series")
```

```
ggplot(plot_Fstat, aes(month, value)) +
  geom_line(aes(colour = series)) +
  labs(y = "Chi Square Test Statistic and Boundary", x="Break Date")
```

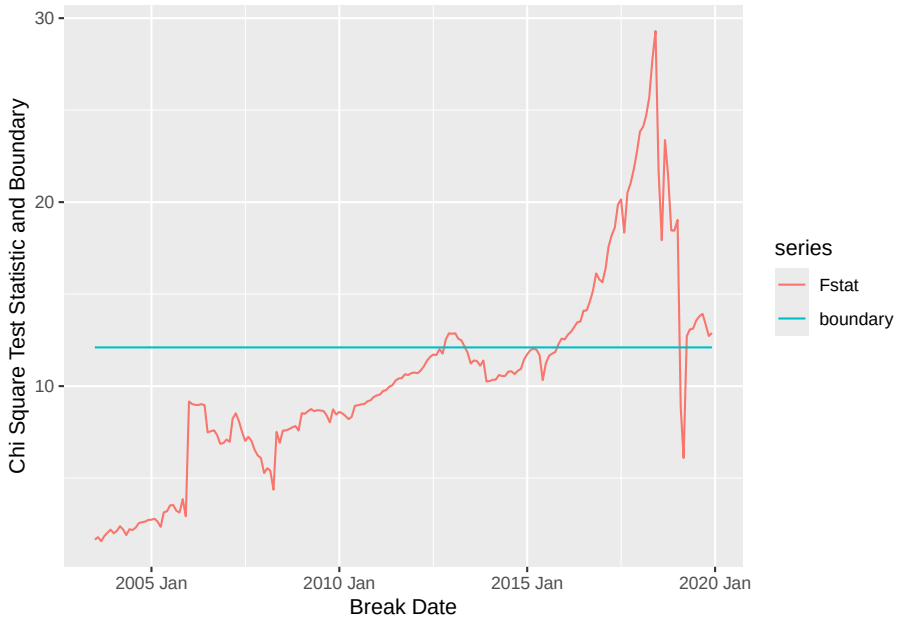

Figure 7.5: Plot of Fstats and Critical Boundary

Even though we are confident the no-break can be rejected, it is nevertheless useful to test the specific hypothesis at the June 2018 *supF* break date. A test of a structural break at the optimal break point is implemented using the *sctest()* function in the *strucchange* package. The very small p value indicates that it is possible to strongly reject the no structural change null hypothesis

```
sctest(y~X-1, type = "supF" , from = t_L, to = t_H)
```

```
##
##  supF test
##
## data:  y ~ X - 1
## sup.F = 29.303, p-value = 1.864e-05
```

7.3.5 Robust Fstats

To conclude this section on single break testing it is important to recognize that the outcome of the *supF* test is sensitive to heteroskedasticity and autocorrelation in the regression residuals. These two features affect the covariance matrix of the coefficient estimates and therefore typically also affect the F statistic. In the standard linear model the variance-covariance matrix of the estimated coefficients is given by $\sigma^2(X'X)^{-1}$ where $\sigma^2 = RSS/(n-k)$ is the estimated variance of the residuals. This standard variance-covariance matrix is not efficient in the presence of heteroskedasticity and autocorrelation, and so an alternative variance-covariance matrix should be used.

Two popular estimators are the: (1) heteroskedasticity consistent covariance matrix estimator (*vcovHC*); and (2) the heteroskedasticity and autocorrelation consistent covariance matrix estimator (*vcovHAC*). Let's re-examine the distribution of *Fstats* and the *supF* test of the no-break null with each of this covariance matrix estimators. This involves using the *vcov* option in the *Fstats* function. The *vcovHC* estimator requires *vcov=sandwich* (the *vcovHC* estimator is commonly referred to as a sandwich estimator). The *vcovHAC* estimator requires assigning the following function: *vcov = function(x, …) vcovHAC(x, type = "HAC", …).*

```
Fstats_HC <- Fstats(y~X-1, from = t_L, to = t_H,vcov = sandwich  )

Fstats_HAC <- Fstats(y~X-1, from = t_L, to = t_H,vcov = function(x, ...) vcovHAC(x, type = "HAC",
```

Let's add this pair of robust *Fstats* values to the *data1* data frame and plot all three in order to highlight the differences. Figure 7.6 shows that The *vcovHC* and *vcovHAC* plots are similar but both are strikingly different than the non-robust distribution of *Fstats*. In particular, the original June 2018 optimal break date is now barely significant. Moreover, earlier dates which were far from significant iwth the non-robust estimator and now also marginally significant. Figure 7.6 reveals significant heteroskedasticity in the spark spread time series. As well, with only one lag included in the model it is possible that autorcorrelation remains in the data. The *vcovHC* estimator controls for heteroskedasticity whereas the *vcovHAC* estimator controls for both heteroskedasticity and autocorrelation.

```
supF_hc <- tibble(Fstat_HC = Fstats_HC$Fstats)
supF_hac <- tibble(Fstat_HAC = Fstats_HAC$Fstats)

data1 <- bind_cols(data1,supF_hc,supF_hac)

plot_Fstat <- melt(data1, id.vars = "month", variable.name = "series")

ggplot(plot_Fstat, aes(month, value)) +
```

```
  geom_line(aes(colour = series)) +
  labs(y = "Chi Square Test Statistic and Boundary", x="Break Date")
```

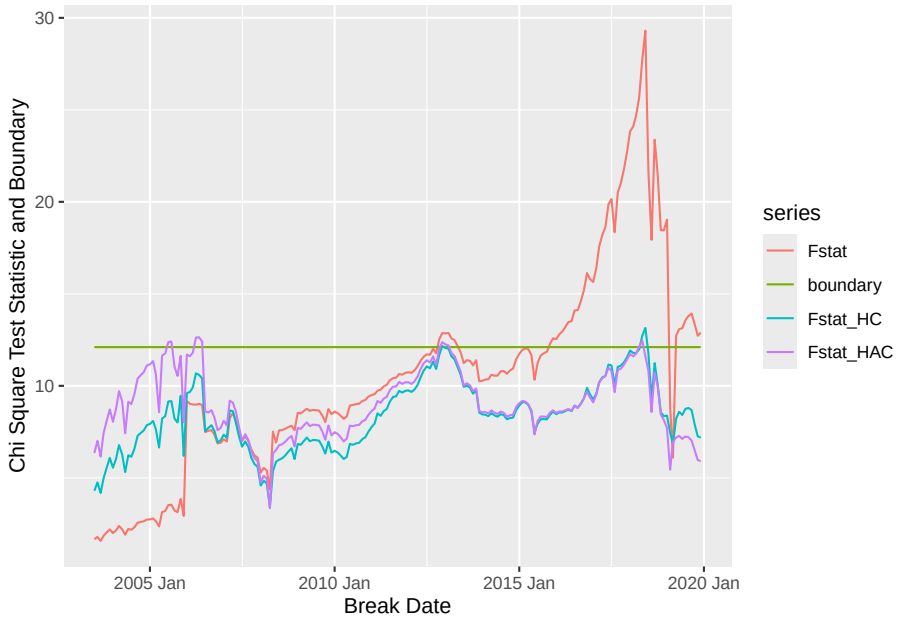

Figure 7.6: Robust Fstats and Critical Boundary

7.4 Multiple Breaks

Identifying and testing for multiple breaks is much more complicated than the single break case. For large data sets the number of unique combinations of break points can be very large, which means that grid search methods for identifying the optimal combination of breaks are highly inefficient. Fortunately, Bai and Perron [1998] derived a dynamic programming procedure to simplify the task of detecting and testing for multiple structural breaks.

As a way to manually illustrate the Bai and Perron [1998] programming procedure a toy data set with 18 observations and two obvious break points has been constructed. This manual optimization procedure can easily be scaled to more than two breaks and a large number of observations. Later we will use the *breakpoints()* function in the *strucchange* package to automate the identification of the optimal break points. However, the *strucchange* package does not have a function for individually testing the significance of each break when there are

multiple breaks. The procedure for constructing these individual p values is beyond the scope of this chapter.

Figure 7.5 reveals that the spark spread data series has only one break. A data series which is better suited for multiple break analysis is the natural gas component of the spark spread (see Figure 7.2). The price series itself is too volatility to work with and so we instead analyze the first difference of the natural gas price series. If the Bai and Perron [1998] procedure is restricted to one break, the results are consistent with the result of *supF* analysis. However, the optimal number of breaks according to the Bai and Perron [1998] procedure is two, and in this case there are some significant differences between the results from the Bai and Perron [1998] procedure and the results from the *supF* procedure.

7.4.1 Dynamic Programming Example

The simulated monthly data was constructed by first simulating normally distributed data for a simple trend model ($w_t = \alpha + \beta t + e_t$) and then adding in breaks in the intercept (α) term. The goal of the exercise is to use the Bai and Perron [1998] procedure to identify this pair of breaks. We begin by reading in the single time series from a saved *RDS* file and plotting it. Figure 7.7 shows that there are likely two breaks in the intercept of this time series, one between months 5 and 10, and a second between months 10 and 15.

```
w <- readRDS(here("data/ch6","break_simulate.RDS"))
month <- 1:18
plot(x=month, y=w, type = "l", lty = 1)
```

To estimate the trend model we require a constant and a trend stored in a matrix, which we will call V.

```
T <- 18
unit <- matrix(1,T,1)
trnd <- seq(1,T)
V <- as.matrix(cbind(unit,trnd))
```

The formal analysis begins by creating an upper triangular *RSS* matrix. Cell *i,j* in the upper right of this triangle shows the RSS from a regression with the sample data beginning in month i and finishing in month j. The following function is used to fill in the elements of this triangular matrix.

```
calcrss <- function(i, j){
  ws <- w[i:j]
  Vs <- V[i:j,]
  resid <-lm.fit(x = Vs, y=ws)$residuals
```

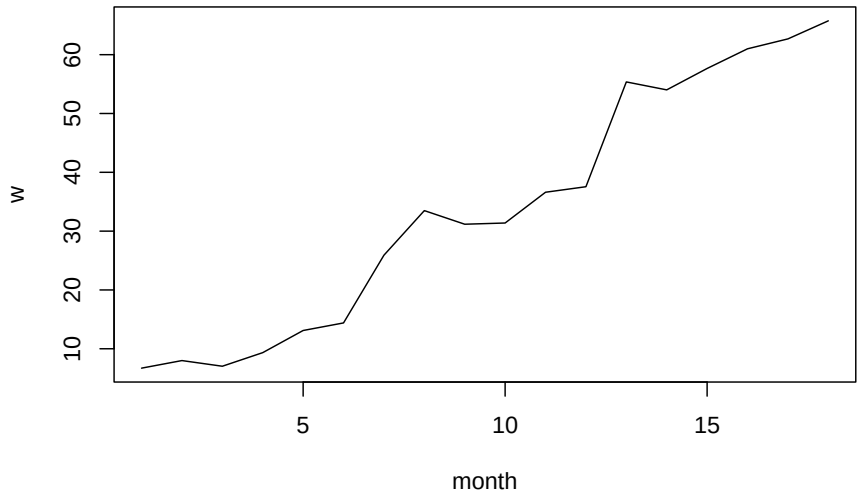

Figure 7.7: Simulated Trend Stationary Data with Two Breaks

```
  rss <- sum(resid^2)
  rss
  return(round(rss,1))
}
```

Let's test the function by calculating the values in cell (2,11) and cell (8,17) of the triangular *RSS* matrix. The first value is the RSS from a regression which runs from months 2 to 11 and the second is the RSS from a regression which runs from months 8 o 17.

```
calcrss(2,11)
```

```
## [1] 114.2
```

```
calcrss(8,17)
```

```
## [1] 168.5
```

When creating the triangular RSS matrix we require a minimum number of observations for each regression. Let's set this minimum at $h = 2$. The following code creates the RSS triangular matrix.

```
h <- 2
triang <- matrix(NA,nrow=T,ncol=T)
for(row in 1:(T-h+1)){
  for(tau in (row+h-1):T){
    triang[row,tau] <- calcrss(row,tau)
  }
}
```

The top row of this matrix is used below, and so let's extract it and place in a vector named *z*.

```
z <- t(triang[1,])
```

The next step is to use the completed RSS triangular matrix to create a second upper trianglular matrix, which we will call *RSSb*. Element *i,j* of this matrix shows the RSS if the regression uses data which begins in month i, runs through to month $T = 18$, and has one break at month $j > i$.

```
RSSb <- matrix(NA,nrow=T,ncol=T)
#row <- 4
for(row in 1:(T-h-1))
for(k in (h-1):(T-h-row)){
 #k<-6
  RSSb[row,row+k] <- triang[row,row+k] + triang[row+k+1,T]
}
```

For each row in the matrix *RSSb*, the minimum RSS across all the RSS values stored in that particular row is selected. For example, let's examine the RSS values in row/month 4 with possible breaks at months 11, 12 and 13. All of other RSS values in row 4 are higher. Therefore, the minimum of these three values, 118.0, is saved for further analysis.

```
RSSb[4,11:13]
```

```
## [1] 202.9 118.0 189.8
```

We will save the minimum RSS value for each row of *RSSb* in a vector named *RSSb_min*. The values in this vector are decreasing in value because rows with higher i values have few available months to include in the RSS minimizing regression.

```
bp1 <- matrix(NA,nrow=T,ncol=1)
RSSb_min <- matrix(NA,nrow=T,ncol=1)
for (row in 1:(T-h-1)){
  bp1[row] <- which.min(RSSb[row,])
  RSSb_min[row] <- RSSb[row,bp1[row]]
}
t(RSSb_min)
```

```
##       [,1]  [,2]  [,3] [,4]  [,5]  [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
## [1,] 151.6 131.8 118.1  118 113.3 102.6 31.6 23.7 12.1  10.7   7.6   1.1   0.5
##      [,14] [,15] [,16] [,17] [,18]
## [1,]     0     0    NA    NA    NA
```

We can now construct the dynampic programming value function. Row i of this function can be expressed as $z[i] + RSSb_{min}[i]$. The first term, $z[i]$, shows the RSS if the sample ranges from date 1 to date i. The second term, $RSSb_{min}[i]$, shows the minimum feasible RSS if the sample ranges from date $i + 1$ to date T, and the break in this second segment has been chosen optimally. For each row i the value function is stored in the *RSSc* vector.

```
RSSc <- matrix(NA,nrow=T,ncol=1)
for (t in h:T){
  RSSc[t] <- z[t] + RSSb_min[t]
}
```

Each row of *RSSc* corresponds to a different first-segment break date. Thus, the row in *RSSc* which has the lowest RSS is the optimal first-segment break date. After this date has been identified, we can move back and recover the optimal date in the second segment. This type of backward recursion is a standard dynamic programming method.

If we implement these last steps we see below that the first optimal break date is 7, and the second optimal break date is 12. These values are consistent with our earlier prediction that the first break will be between months 5 and 10, and the second break will be between months 10 and 15.

```
bp1_star <- which.min(RSSc)
bp2_star <- bp1[bp1_star]
bp1_star
```

```
## [1] 7
```

```
bp2_star
```

```
## [1] 12
```

7.4.2 Multiple Break Date Application

In this section we will implement the Bai and Perron [1998] dynamic programming procedure for identifying break points by calling the *breakpoints()* function from the *strucchange* package. The data we will use is the first difference in the monthly price of natural gas. This monthly natural gas price series, which ranges from June of 2001 to December of 2021, can be extracted from the *power* data frame, which was previously imported as *power.RDS*. The first difference of the natural gas price is coerced as a time series variable to facilitate the analysis below.

```
power <- readRDS(here("data/ch6","power.RDS"))

gas <- ts(power$gas_sa, start = c(2001, 6), frequency = 12)
D.gas <- diff(gas)
```

Figure 7.8 shows a plot of the differenced price of natural gas. As expected, much of the volatility from the non-differenced price series has been eliminated. If there are structural breaks in this data they are not nearly so obvious as in our earlier break date applications.

```
plot(D.gas, type = "l", lty = 1)
```

Before using the *breakpoints()* function to identify the optimal break points, let use the *Fstats()* and *boundary()* functions which were featured in the previous section to generate the *Fstats* series and boundary for this new data set. Both of these will be plotted after the optimal break points have been identified.

```
fs_gas <- Fstats(D.gas ~ 1, from = 20, to = 226)
fs_bound <- boundary(fs_gas,alpha=0.05)
Fstat_tib <- tibble(Fstat=fs_gas$Fstats)
Fstat_bound <- tibble(Boundary=fs_bound)

power3 <- as_tibble(power[,1]) %>%
  slice(21:(n()-20))
power3 <- bind_cols(power3,Fstat_tib,Fstat_bound)
```

Let's begin by restricting the *breakpoints* function to one break. In this case the dynamic programming procedure of Bai and Perron [1998] is relatively simple.

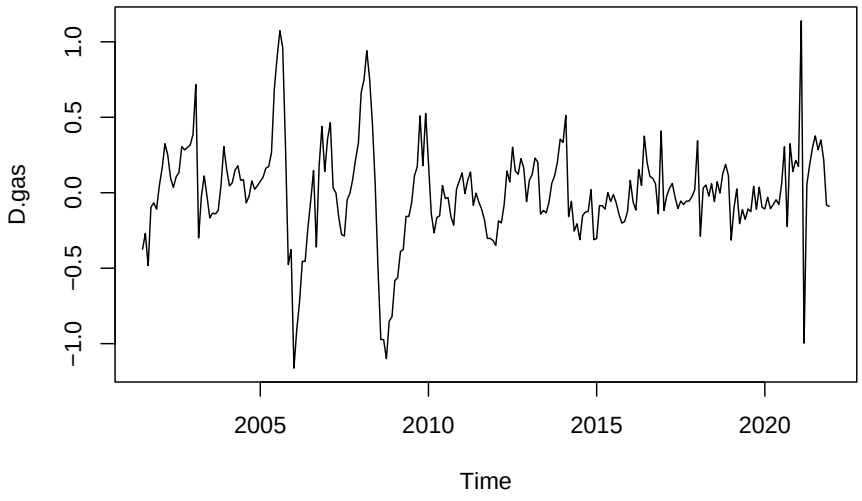

Figure 7.8: First Difference of Monthly Natural Gas Price

The results below show first that one break results in a lower RSS and a lower BIC as compared to no breaks, and second that that the optimal break date is October of 2005. There are no formal tests of statistical significance at this point.

```
bp.gas <- breakpoints(D.gas ~ 1, breaks=1)
summary(bp.gas)
```

```
##
##   Optimal (m+1)-segment partition:
##
## Call:
## breakpoints.formula(formula = D.gas ~ 1, breaks = 1)
##
## Breakpoints at observation number:
##
## m = 1   52
##
## Corresponding to breakdates:
##
## m = 1   2005(10)
##
```

```
## Fit:
##
## m   0      1
## RSS  26.05  24.86
## BIC 156.76 156.26
```

```
brk1 <- bp.gas$breakpoints[1] - 19
brk1_date <- as.Date(power3$month[brk1])
```

The vertical line in Figure 7.9 is the October, 2005 optimal break point. Notice that it lines up with the *supF* (i.e., maximum of the *Fstats*). This means that the two different methods for identifying the optimal break date when only one break date is allowed give consistent results. Moreover, the fact that the *supF* lies above the boundary implies that this first break point is statistically significant.

```
ggplot(power3, aes(month)) +
  geom_line(aes(y = Fstat)) +
  geom_line(aes(y = Boundary)) +
  geom_vline(xintercept = as.numeric(as.Date(brk1_date)), linetype=4) +
  ylab('Fstat')
```

```
## Don't know how to automatically pick scale for object of type <ts>. Defaulting
## to continuous.
```

Let's repeat the previous procedures except now we will allow the *breakpoints()* function to optimally choose the number of breaks, possibly in excess of one. The results give RSS and BIC statistics for 5 possible breakpoints. Notice that the BIC is minimized with two break points. The first is June of 2008 and the second is March of 2012.

```
bp.gas <- breakpoints(D.gas ~ 1)
summary(bp.gas)
```

```
##
##   Optimal (m+1)-segment partition:
##
## Call:
## breakpoints.formula(formula = D.gas ~ 1)
##
## Breakpoints at observation number:
##
## m = 1   52
```

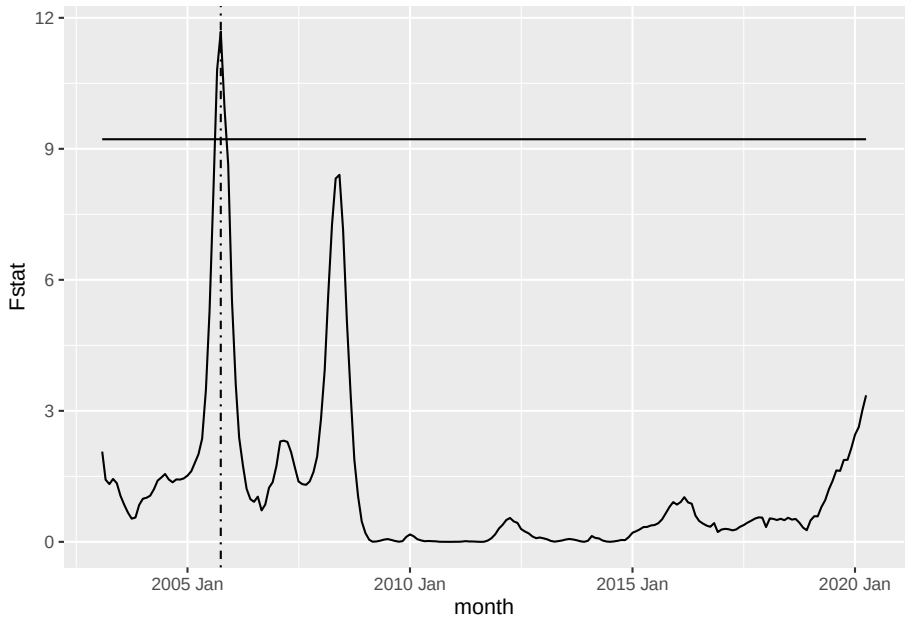

Figure 7.9: Fstats for Difference Natural Gas Price: One Break Point

```
## m = 2      84 129
## m = 3   43 84 129
## m = 4   52 96     152 197
## m = 5   43 84 129 174 210
##
## Corresponding to breakdates:
##
## m = 1   2005(10)
## m = 2           2008(6) 2012(3)
## m = 3   2005(1)  2008(6) 2012(3)
## m = 4   2005(10) 2009(6)         2014(2)  2017(11)
## m = 5   2005(1)  2008(6) 2012(3) 2015(12) 2018(12)
##
## Fit:
##
## m   0      1      2      3      4      5
## RSS  26.05  24.86  23.75  23.69  23.59  23.64
## BIC 156.76 156.26 156.10 166.45 176.47 187.99
```

```
brk1 <- bp.gas$breakpoints[1] - 19
brk2 <- bp.gas$breakpoints[2] - 19
brk1_date <- as.Date(power3$month[brk1])
```

```
brk2_date <- as.Date(power3$month[brk2])
```

As before we can superimpose these two optimal break points on the *Fstats* plot. Figure 7.10 reveals an unexpected result. While it is understandable that these two break points would emerge if they were forced, it is not clear why two points are chosen as optimal when the one break date which is significant with the *supF* test is no long deemed optimal.

```
ggplot(power3, aes(month)) +
  geom_line(aes(y = Fstat, colour = "var0")) +
  geom_line(aes(y = Boundary, colour = "var1")) +
  geom_vline(xintercept = as.numeric(as.Date(brk1_date)), linetype=4) +
  geom_vline(xintercept = as.numeric(as.Date(brk2_date)), linetype=4) +
  ylab('Fstat')
```

```
## Don't know how to automatically pick scale for object of type <ts>. Defaulting
## to continuous.
```

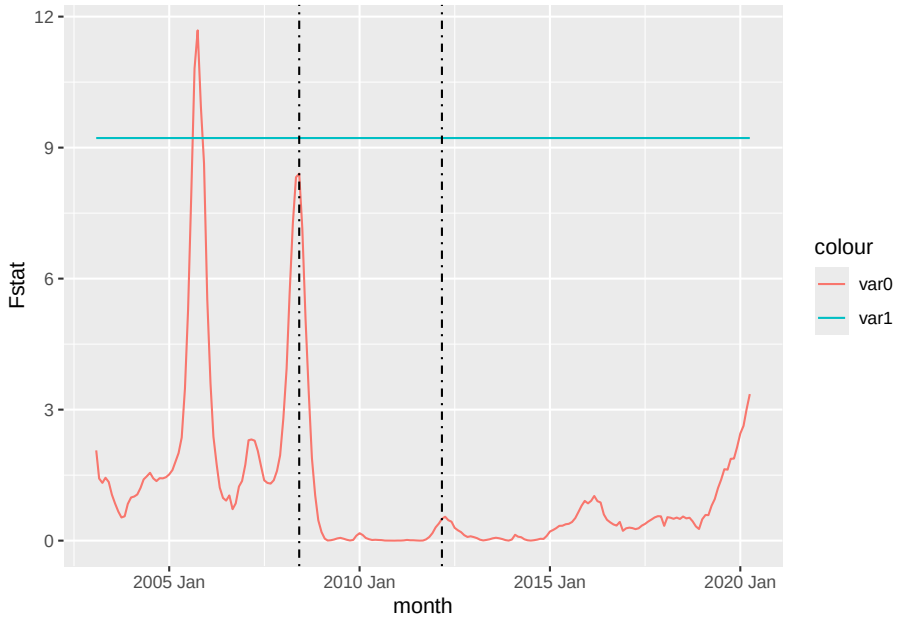

Figure 7.10: Fstats for Difference Natural Gas Price: Two Break Points

7.5 Unit Root and Structural Breaks

As was discussed in the Introduction, allowing for break points in time series has become an important component of testing for stationarity. There are many sophisticated methods which allow for one or more structural breaks while testing for a unit root. In this section we will use the spark spread data and the comparatively simple Zivot and Andrews [1992] procedure. Specifically, we will test for a unit root in the data while allowing for one structural break in the intercept parameter during the testing procedure. We will run the test with lags on the differenced data as a means to control for autocorrelation.

7.5.1 Testing Frameowrk

Using the notation from the previous section, let y_t denote the spark spread in month t. Under the null hypothesis, y_t evolves as a unit root with drift: $y_t = \mu + y_{t-1} + e_t$. Under the alternative hypothesis, y_t is a trend stationary stochastic process with a one time break in the intercept of the time series at an unknown time. The break date is chosen as the point in time for which the evidence in favour of the alternative hypothesis is at a maximum. This is equivalent to choosing the break date which corresponds to the largest absolute value of the unit root t statistic.

Define $DU(T_B)$ as a dummy variable which takes on a value of 1 for $t > T_B$ and 0 otherwise. The following equation is estimated for all possible break points:

$$y_t = \mu + \theta DU(T_B) + \beta t + \alpha y_{t-1} + \sum_{j=1}^{k} c_j \Delta y_{t-j} \tag{7.10}$$

For each feasible value of T_B we will test the unit root null, $\alpha = 1$, and collect the t statistic for that test. Within the set of collected t statistics, the break date which results in the largest absolute value of the t statistic (i.e., the one for which evidence of a unit root is weakest) is chosen as the optimal break point. Let λ denote the optimal break date. The test for the unit root requires substituting λ for T_B in equation (7.10), re-estimating the equation and then once again conduct the $\alpha = 1$ test of the unit root null. Zivot and Andrews [1992] provide critical values for this t test.

7.5.2 Manual Estimation

In this section we will manually implement the Zivot and Andrews [1992] testing procedure. The code used below is a close adaptation of the code used in the *ur.za()* function from the *urca* package. In the section to follow we will confirm our testing results using the *ur.za()* function. For this manual estimation we will

include one lag. In the next section when working with the *ur.za()* function, we will vary the number of lags which are included in the estimation. To facilitate the required looping we will once again conduct the analysis using matrices rather than data frames.

The first step is add the spark spread data to a y matrix and then add y and the lag of y to a *datmat* matrix.

```
#rm(list = ls())
#pacman::p_load(tidyverse, here, urca)

#power <- readRDS(here("data/ch6","power.RDS"))
y <- as.matrix(power$SprkSprd)
L.y <- lag(y)
n <- length(y)
datmat <- matrix(NA, n, 4)
datmat[,1] <- y
datmat[,2] <- L.y
```

We can now add a trend line and one lag of Δy to the *datmat* matrix

```
trend <- seq(1, n)
L.D.y <- c(NA, lag(diff(y)))
datmat[,3] <- trend
datmat[,4] <- L.D.y
colnames(datmat)[1:4] <- c("y", "L.y", "trend","L.D.y")
```

The next step is to create a function which adds the dummy variable to the *datmat* matrix, estimates equation (7.10), and returns the t statistic for the $\alpha = 1$ test. The function requires as input the break point z. This function will later be used to generate t statistics for all feasible break points.

```
roll <- function(z){
du <- c(rep(0, z), rep(1, (n-z)))
rollmat <- cbind(datmat, du)
lmfit <- lm(rollmat[-c(1:2),1]~rollmat[-c(1:2),-1])
coef <- summary(lmfit)$coefficients
tstat <- (coef[2,1]-1)/coef[2,2]
return(tstat)
}
```

The *sapply()* function in base R can be used to repeatedly evaluate a function with a parameter from a pre-defined set and store the results in a vector. In our case, we want to repeatedly evaluate the *roll* function which we just created with the full range of break points, which are stored as $idx = [1, 2, ..., n-1$. The vector of returned t statistics are stored in the *roll.stat* vector.

```
idx <- 1:(n-1)
roll.stat <- sapply(idx, roll)
head(roll.stat)
```

```
## [1] -8.470395 -8.470395 -8.496301 -8.312048 -8.371277 -8.396162
```

Keep in mind that the t statistic being calculated is not the usual one which requires dividing the coefficient by the standard error. The null is $\alpha = 1$, which means that the t statistic is calculated by subtracting one from the α coefficient before dividing by the standard error.

The next step is to find the date which gives the largest absolute t statistic since this is the point where the evidence against the null hypothesis is the greatest. All of the t statistics are negative and so it is sufficient to find the break date which corresponds to the minimum value of t. After this optimal break point has been identified a corresponding dummy variable series can be constructed and added to the main data matrix for the final regression.

The results below show that the optimal break is at position 240 in the data set. This position corresponds to May of 2021, which is much later than the June of 2018 break date which was identified as otimal in the previous section. The likely reason for this outcome is that in the Zivot-Andrews test there is no minimum and maximum value for the break points which are used in the analysis.

```
bpoint <- which.min(roll.stat)
bpoint
```

```
## [1] 240
```

```
du <- c(rep(0, bpoint), rep(1, (n-bpoint)))
testmat <- cbind(datmat, du)
```

The last step is to calculate the t statistic for the $\theta = 1$ null hypothesis and then report this value along with the set of critical t values.

```
lmtest <- lm(testmat[-c(1:2),1]~testmat[-c(1:2),-1])
coeftest <- summary(lmtest)$coefficients
teststat <- (coeftest[2,1]-1)/coeftest[2,2]
teststat
```

```
## [1] -9.978958
```

```
cval <- c(-5.34, -4.8, -4.58)
cval
```

```
## [1] -5.34 -4.80 -4.58
```

7.5.3 Zivot-Andrews Function

The final test statistic (-9.9789) is well below the set of critical values and so the unit root null hypothesis can easily be rejected for this specific case of one lag. Let's use the *ur.za()* function from the *urca* package to ensure that test statistic from the package matches this current value.

```
test_za <- ur.za(power$SprkSprd,model="intercept",lag=1)
summary(test_za)
```

```
##
## ################################
## # Zivot-Andrews Unit Root Test #
## ################################
##
##
## Call:
## lm(formula = testmat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -25.067  -4.527  -0.834   2.931  71.980
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.92534    1.30303  -1.478  0.14083
## y.l1         0.30466    0.06968   4.372 1.83e-05 ***
## trend        0.02868    0.00959   2.991  0.00307 **
## y.dl1        0.10964    0.06256   1.752  0.08097 .
## du          20.82401    4.47242   4.656 5.34e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.841 on 240 degrees of freedom
##   (2 observations deleted due to missingness)
## Multiple R-squared:  0.3772, Adjusted R-squared:  0.3668
## F-statistic: 36.34 on 4 and 240 DF,  p-value: < 2.2e-16
##
```

```
##
## Teststatistic: -9.979
## Critical values: 0.01= -5.34 0.05= -4.8 0.1= -4.58
##
## Potential break point at position: 240
```

The test statistics match. What remains is to conduct the analysis with additional lags. Presented below are same results as above except with two and three lags.

```
summary(ur.za(power$SprkSprd,model="intercept",lag=2))@teststat
```

```
## [1] -7.88806
```

```
summary(ur.za(power$SprkSprd,model="intercept",lag=3))@teststat
```

```
## [1] -7.6565
```

With the additional lags the test statistic is somewhat higher but still well below the critical values. Thus, the original conclusion of rejecting the unit root null remains.

7.6 Summary and Conclusions

This chapter began by examining structural breaks in the gasoline price premium paid by Vancouver drivers. In this case the cause of the break was the introduction of a new tax on gasoline for Vancouver drivers. With the break date known we separated the price premium into pre-break and post-break samples. We chose to ignore autocorrelation and thus focus only on the break in the mean premium. With this simplifying assumption we used a simple two-sample t test to show that the null hypothesis of no break in the mean price could be rejected.

The next step in our analysis was to show that we can obtain the same testing outcome if we conducted an analysis of variance (ANOVA) on the pre-break and post-break samples. In this case the ratio of the between group variation (i.e., variation in the pre-break and post-break means) and the within variation (i.e., average variance of the pre-break and post-break samples) was known to be distributed as an F statistic. This allows us to conduct an F test for the no-break null hypothesis and arrive at the same p value as the previous t test.

Most of the analysis was carried out assuming an autoregressive stochastic process with one lag. In this more general case a t test or ANOVA is no longer

appropriate and so we turned to the more general F test for linear restrictions on the estimated coefficients of a autoregressive model. When the break date is know, this method is the widely-used Chow test for a structural break. Using monthly data on the spark premium earned by producers of electricity and a "guess" regarding the date of the break (assumed to be accurate) the Chow test allowed us to reject the null hypothesis of no structural break. We showed that the Chow F statistic can be obtained either by collecting the residual sum of squares (RSS) separately for the pre-break and post-break samples, or by collecting the RSS from an estimated model with a dummy variable for the break.

The most important analysis in this chapter concerned the case of an unknown break date. Here the goal is to simultaneously identify the most likely structural break and test the no-break null hypothesis. With the original Quandt method researchers plotted the the Chow F statistic for every possible break in the data. The maximum value from this set of F statistics is the Quandt likelihood ratio (LR) test statistic, which is also known as the *supF* statistic. The break date associated with the *supF* is viewed as being the most likely break date in the sample.

We recognized that with large samples the likelihood ratio test statistic which is used to test the no-break null hypothesis converges to a Chi square distribution. In our model with two coefficients the Chi square test statistic has double the value of the Chow F statistic. Now the *supF* is the maximum from the set of Chi square test statistis – one for each possible break point. A comparison of the *supF* with the *boundary* identifies if the no-break null can be rejected. The *Fstats()* package gives the distribution of *Fstats* and the boundary, and the *sctest()* function gives the p value for the specific case where the *supF* (which identifies the most likely break point) is compared to the boundary.

We used a different approach when testing for multiple structural breaks. Rather than working with F statistics we use a dynamic programming procedure to find combinations of breaks which minimized the overall residual sum of squares (RSS). After using a simple data set to illustrate the method, we used the *breakpoints()* function from the *strucchange* package to identify multiple breaks in the first difference of the monthly price of gasoline. When the number of breaks was restricted to one, the results were consistent with the *supF* method. When no restrictions was imposed on the number of breaks, the *breakpoints* function indicated that two breaks were optimal. However, the timing of these two breaks were not consistent with the *supF* method.

We concluded our analysis by testing for a unit root in the presence of one structural break. The method involves calculating the t statistic which is used to test the unit root null hypothesis for all possible break points. The break point with the highest absolute t statistic, which maximizes the evidence against the unit root null, is identified as the most likely break point. This most likely break point is then incorporated into the unit root testing equation. Using the spark spread data there was ample evidence to reject the unit root null, and

thus conclude that the data is stationary.

Chapter 8

Seasonality

Food and energy prices often exhibit seasonality, which means that price patterns over the course of a year repeat across years. For example, the price of fresh vegetables is typically lowest during the summer months when local production is plentiful and highest in the winter months when local markets rely more heavily on imports. In deregulated markets for electricity, seasonal prices reflect both weekly peak demand (e.g. daytime on weekdays) and monthly differences in the demand for heating and cooling. For storable commodities with an annual harvest such as corn and soybeans, price tends to be lowest at harvest and then increases over the marketing year at a rate which reflects the marginal cost of storage.

In this chapter we are interested in how seasonality affects the estimated relationship between the price of a commodity and a set of exogenous determinants such as weather and the exchange rate. We will see that failing to account for seasonality may bias the estimated coefficient and depress the standard errors of the estimated coefficients. Seasonality implies a time dependent mean of the price series and is therefore a specific type of non-stationarity. However, unlike the case of a stochastic trend, taking the first difference of the price series will not eliminate the seasonality. The bulk of the analysis consists of examining how seasonal dummies and seasonal differencing can be used to control for seasonality.

Section 8.1 provides examples of seasonality in food and energy prices, and shows alternative ways for isolating the seasonal component. Section 8.2 describes why the estimated coefficient may be biased and why the associated t values are typically inflated if seasonality is not accounted for. In Section 8.3 the autoregressive integrated moving average (ARIMA) model is used to control for seasonality, first by using seasonal dummies and then by using seasonal differencing. Section 8.3 concludes by showing that a standard regression model can be enhanced by assuming the error term follows a seasonal ARIMA stochastic process. This particular specification is applied to a pair of case studies in

Section 8.4. The first case study examines energy consumption and the second case study examines the relationship between the U.S. price of slaughter hogs and the U.S. price of crude oil. Concluding comments are provided in Section 8.5.

8.1 Background

This section begins with a brief review of the literature on seasonality in food and energy prices. The bulk of this section is devoted to a descriptive analysis of seasonality in the price of fresh strawberries, grapes and gasoline.

8.1.1 Literature

As previously noted, commodity prices are generally analyzed from either a price forecasting/decomposition perspective or a time series econometrics perspective. Although seasonality plays an important role in both strands of research the role of seasonality in commodity price forecasting is rather limited. This is likely because forecasting models are generally built for storable commodities such as crude oil, lumber and soybeans, in which case seasonality is either not present or small in magnitude. An important exception is the inclusion of seasonality when the USDA forecasts the consumer price index for the major food categories such as fruits and vegetables [MacLachlan et al., 2022]. For non-food commodities, seasonality is important when forecasting the price of electricity. This is because electricity cannot be stored and as such its price is strongly influenced by changes in seasonal demand [Wagner et al., 2022].

In contrast, seasonality plays an important role when analyzing food and energy prices within a time series econometrics framework. One strand of literature examines seasonality of food prices as a measure of market integration and efficiency in food insecure countries [Gilbert et al., 2017]. In another strand of literature, the difference between the spot price and futures price of a commodity (i.e., the basis) is seasonal due to a risk premium and convenience yield which varies over the course of the marketing year [Sorensen, 2002, Asche et al., 2016]. Oglend [2013] examines seasonality in salmon prices as part of a more general analysis of salmon price volatility. Garcia and Leuthold [2004] review the agricultural futures market literature and cite several papers which examine seasonality in agricultural commodity markets.

In their study of natural gas markets, Mirantes et al. [2012] model seasonality as stochastic rather than deterministic. Moreno et al. [2019] model both long term swings and seasonality in natural gas markets. The market for gasoline has a high degree of seasonality due to the predictably high (low) demand for car travel during the summer (winter) months. Davis [2009] found that stronger environmental regulations is increasing the seasonality of gasoline prices. Lucia

and Schwartz [2002] examined seasonality in Nordic electricity markets (later in this chapter we will use a similar model). Escribano et al. [2011] find significant evidence of seasonality in wholesale electricity prices.

A seasonal effect when applied to commodity pricing is the size of the price deviation which regularly occurs within a calendar year. For example, the price of frozen turkeys in the U.S. typically declines by about $0.10 per pound when supply surges during the November-December holiday season. Størdal et al. [2022] estimate that the price for a Nordic electricity futures contract tends to be about 18 percent lower in February as compared to August.

A seasonally-adjusted price series is one which has the seasonal effects removed. U.S. statistical agencies such as the U.S. Census Bureau, the U.S. Bureau of Labour Statistics and the Federal Reserve of Economic Data (FRED) commonly report seasonally adjusted prices and price indexes. This includes various consumer and producer price indexes, the price of gasoline and the price of frozen turkeys. These agencies generally use a common seasonal adjustment methodology such as the *X-13ARIMA-SEATS*, which was developed by the United States Census Bureau.

8.1.2 Fruit Price Seasonality

Non-storable fresh fruits and vegetables generally have high pricing seasonality due to systematic changes in available supply and consumer demand over a calendar year. Let's take a look a pricing seasonality for U.S. fresh strawberries. To import this data we load the R packages for this chapter and then read in the pre-cleaned *fruit* RDS file.

```
pacman::p_load(tidyverse, here, lubridate, reshape2,fpp3, tsibble, tsibbledata, fable, gridExtra,
```

```
fruit <- readRDS(here("data/ch7","fruit.RDS"))
```

Figure 8.1 below shows the monthly U.S. price of fresh strawberries from January of 2018 to August of 2020. Notice that the high price of a little over $3 per pint occurs in January, and the low price of about $2 per pint occurs in June and July. With the exception of the summer of 2021 the monthly pattern of seasonality is very stable over this four year period.

```
ggplot(fruit[200:nrow(fruit),], aes(x = month, y = strawberry)) +
  geom_line() +
  guides(color = guide_legend(title = "Fruit")) +
  xlab("") + ylab("Dollars per Pint")
```

Figure 8.2 below shows the strong seasonality in the U.S. monthly price of fresh strawberries and grapes over the much longer 2000 to 2022 time period. The

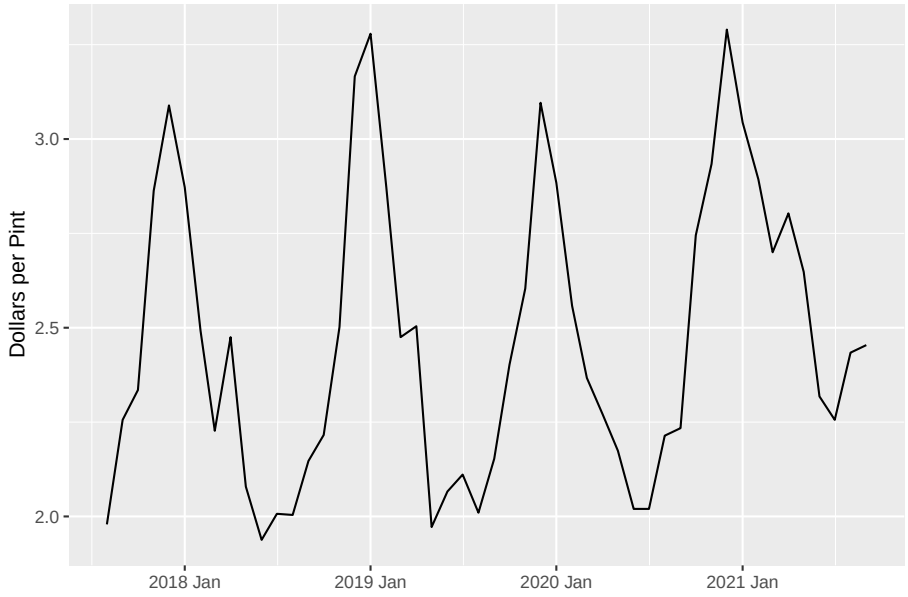

Figure 8.1: Retail Price of Fresh Strawberries

seasonality is not as regular as compared to Figure 8.1 but the overall seasonal patterns remain highly visible for both fruits. The occasional unusually large price spike is likely the result of poor growing conditions during that period and thus a shortage in supply.

```
fruit_long <- melt(fruit[,c("month","strawberry","grape")], id.vars = "month")
```

```
ggplot(fruit_long, aes(x = month, y = value, color = variable)) +
  geom_line() +
  guides(color = guide_legend(title = "Fruit")) +
  xlab("") + ylab("Strawberries ($/pint) & Grapes ($/lb)")
```

It is difficult to tell from Figure 8.2 whether the seasonal effects are largest for strawberries or grapes. One way to quantify seasonality is to calculate for each year the percent gap between the seasonal high price and the seasonal low price and then average this percentage over the sample period. The results below show that for the 2000 to 2020 sample period, the average percent gap is 60.0 percent for strawberries and 51.4 percent for grapes. It appears that seasonality is somewhat stronger in the fresh strawberry market than in the grape market.

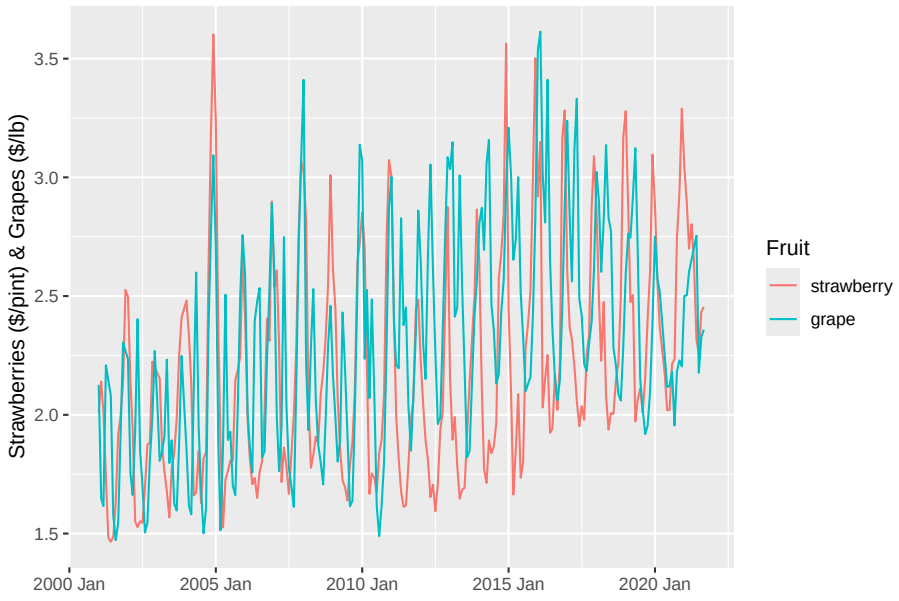

Figure 8.2: Retail Price of Fresh Strawberries and Grapes

```
grouped <- as_tibble(fruit) %>%
  mutate(year = year(month)) %>%
   group_by(year) %>%
  summarise(straw_mean = mean(strawberry),
                   straw_min = min(strawberry),
                   straw_max = max(strawberry),
                  grape_mean = mean(grape),
                  grape_min = min(grape),
                  grape_max = max(grape))
```

```
fruit_sum <- grouped %>%
  mutate(straw_gap = (straw_max-straw_min)/straw_mean,
         grape_gap = (grape_max-grape_min)/grape_mean) %>%
  dplyr::select(year,straw_gap,grape_gap)

fruit_compare <- fruit_sum %>%
  summarise(across(c("straw_gap","grape_gap"), mean))
fruit_compare
```

```
## # A tibble: 1 x 2
##   straw_gap grape_gap
```

```
##       <dbl>     <dbl>
## 1     0.600     0.514
```

When there is strong seasonality a statistical agency often reports the seasonally adjusted version of a data series. Most agencies are currently using a procedure known as X-13ARIMA-SEATS, which was developed by the U.S. Census Bureau. This version can beimplemented in R through use of the *season* package. Let's use the *season* package to construct seasonally adjusted prices for fresh strawberries and grapes.

To create this new time series we must convert the *fruit* tsibble object into a *ts* object. The *seas()* function creates a *seas* object and the *final()* function is used to extract the seasonaly adjusted values from *seas*. Regarding variable names, "Strawberries" is the seasonally-adjusted version of "strawberries", and "Grapes" is the seasonally adjusted version of "grapes".

```
fruit2 <- fruit %>%
  dplyr::select(month,strawberry,grape, ex_mex) %>%
  as_tsibble(index =  month)

fruit_ts <- as.ts(fruit2)

str_seas <- suppressWarnings(seas(fruit_ts[,1]))
Strawberry <- suppressWarnings(final(str_seas))
grape_seas <- suppressWarnings(seas(fruit_ts[,2]))
Grape <- final(grape_seas)

fruit2 <- fruit2 %>%
  mutate(Strawberries = Strawberry,
         Grapes = Grape)
```

Let's add the seasonally-adjusted prices to the revised data frame and then plot the data. With the seasonality removed, Figure 8.3 reveals much more information about short term volatility in the price of these two fruits. It is now clear that the price of grapes is significantly more volatile than the price of strawberries.

```
fruit_long2 <- melt(fruit2[,c("month","Grapes","Strawberries")], id.vars = "month")
```

```
ggplot(fruit_long2, aes(x = month, y = value, color = variable)) +
  geom_line() +
  guides(color = guide_legend(title = "Fruit")) +
  xlab("") + ylab("Strawberries ($/pint) & Grapes ($/lb)")
```

```
## Don't know how to automatically pick scale for object of type <ts>. Defaulting
## to continuous.
```

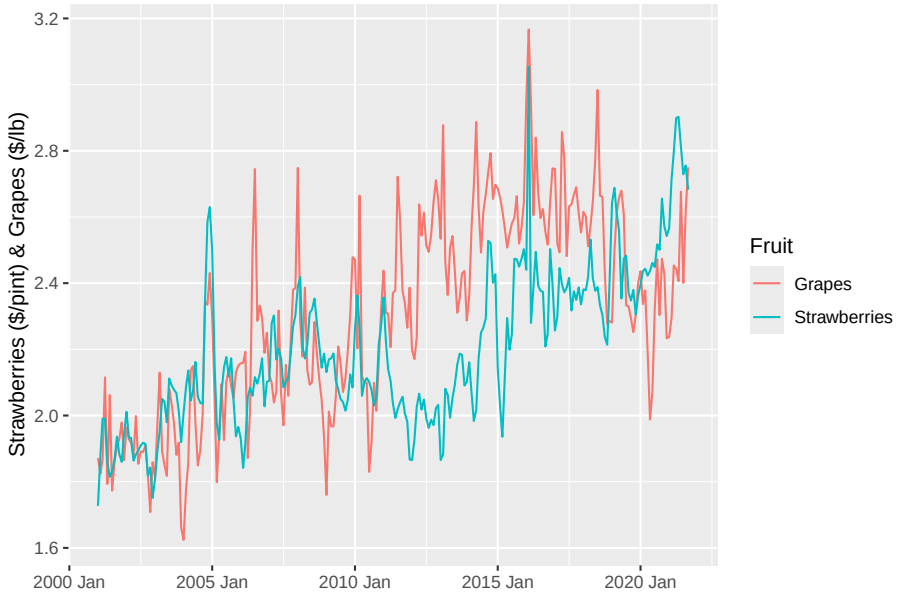

Figure 8.3: Seasonally Adjusted Price of Fresh Strawberries

8.1.3 Gasoline Price Seasonality

In the case of gasoline prices, the Federal Reserve of Economic Data (FRED) reports both a non seasonally adjusted and a seasonally adjusted price series. Let's read in these monthly price indexes from a pre-cleaned RDS file, and let "gasoline" refer to the non-adjusted price series and "Gasoline" refer to the adjusted price series.

```
gasoline <- readRDS(here("data/ch7","gasoline.RDS")) %>%
  mutate(season = 2*(gasoline-Gasoline)/(gasoline+Gasoline))
```

Figure 8.4 reveals strong growth in the non-seasonally adjust price of gasoline between year 2000 and the onset of the great recession in 2008/2009. The price made a strong recovery but then crashed again along with the global price of crude oil in 2015. A third strong price surge in 2021 coincides with the high inflationary "post" COVID time period.

```
ggplot(gasoline, aes(x = month, y = gasoline)) +
  geom_line() +
  xlab("") + ylab("Index: 1982-1984=100")
```

The high degree of price variation in Figure 8.4 makes it difficult to identify the

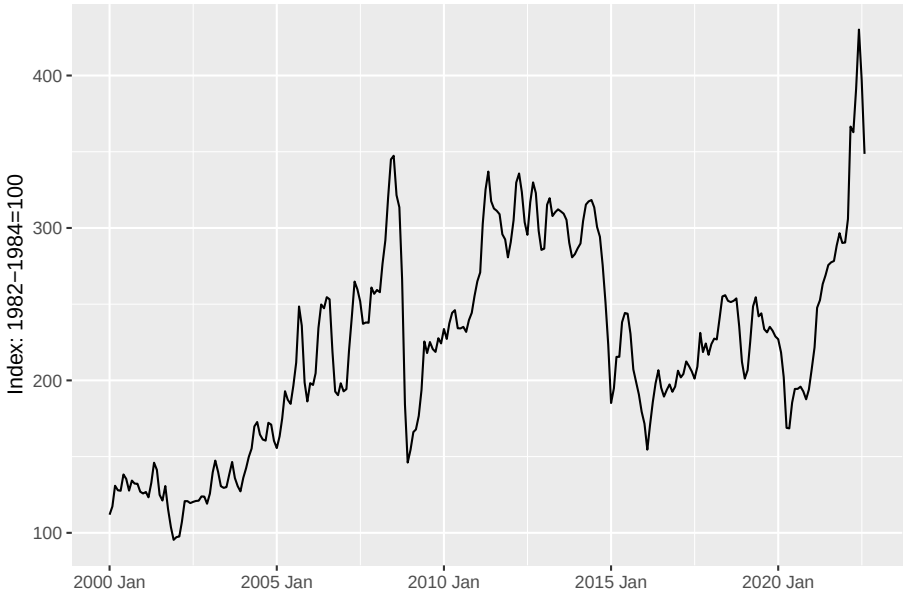

Figure 8.4: Index of U.S. Retail Gasoline Price

seasonality. Let's isolate the seasonal effects by calculating the percent difference between the seasonally adjusted and non-seasonally adjusted indexes. In Figure 8.5 the calculated seasonal adjustment reveals a strong seasonal pattern, with higher prices during the stronger demand summer season and lower prices during the weaker demand winter season. Gasoline prices tend to be approximately 10 percent above the yearly average during the summer and 10 percent below the yearly average during the winter. It is interesting to note that the pattern of seasonality was not impacted with the onset of the global pandemic in 2020.

```
ggplot(gasoline, aes(x = month, y = season)) +
  geom_line() +
  xlab("") + ylab("Percent Difference")
```

The *feasts* package in R is able to decompose a time series into trend, seasonality and residual components. Let's use this package to extract the seasonality in the price of gasoline. The extracted seasonality component is shown in Figure 8.6. It is similar but not identical to the seasonality in Figure 8.5, which is the percent difference between the seasonally adjusted and non-seasonally adjusted time series.

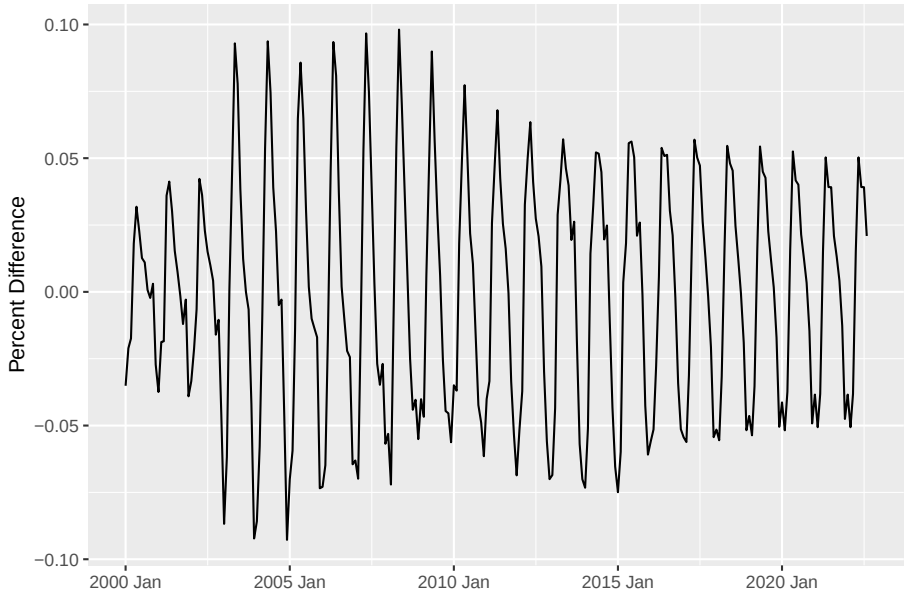

Figure 8.5: Percent Difference Between Seasonal and Non-Seasonal Price Index for Gasoline

```
gas_decompose <- gasoline %>% model(STL(gasoline)) %>% components()
```

```
ggplot(gas_decompose, aes(x = month, y = season_year)) +
  geom_line() +
  xlab("") + ylab("Cents per Gallon")
```

8.1.4 Corn Price Seasonality

Seasonality is much less pronounced in crops such as corn which can be stored across years. The seasonality which is present represents the cost of storing the corn over time. In particular, the price of corn tends to be lowest immediately after harvest and it then gradually rises over time at a rate which reflects the marginal cost of storage. With the arrival of the new harvest the following year, price deceases and the cycle continues.

The corn data is read in from a pre-cleaned RDS file. Figure 8.7 shows the monthly price of corn received by U.S farmers between 1980 and 2022. Notice that the timing of the price surges and rapid declines are similar to those in the gasoline market (see Figure 8.4).

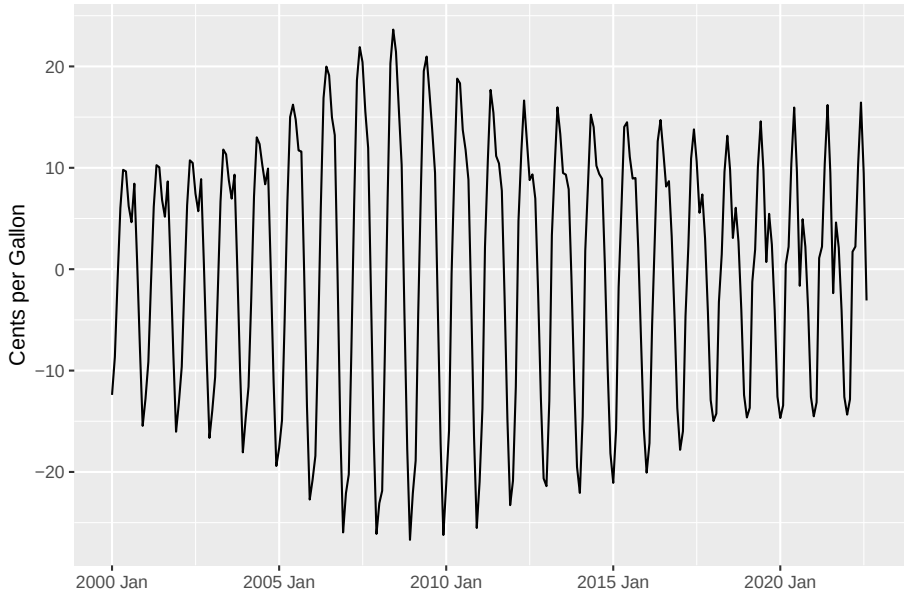

Figure 8.6: Seasonal Component of Gasoline Price Decomposition

```
corn <- readRDS(here("data/ch7","corn.RDS"))

ggplot(corn, aes(x = month, y = corn)) +
  geom_line() +
  xlab("Month") + ylab("Dollars per Bushel")
```

We can isolate the seasonality in the price of corn by grouping the price series by year and then subtracting each month's price from the annual average. These monthly deviations can then be averaged to provide a summary measure of monthly seasonality.

Figure 8.8 shows the monthly corn price deviations. As expected, prices tend to be lowest when the corn is harvested in the late fall, and highest when corn stocks are lowest in the summer. The price difference between fall and summer covers the cost of storing the grain between these two time periods.

```
mnth_mean <- as_tibble(corn) %>%
  group_by(month_ind) %>%
  dplyr::summarize(dev_mean = mean(deviate, na.rm = TRUE)) %>%
  mutate(mnth = as.character(month_ind))
```

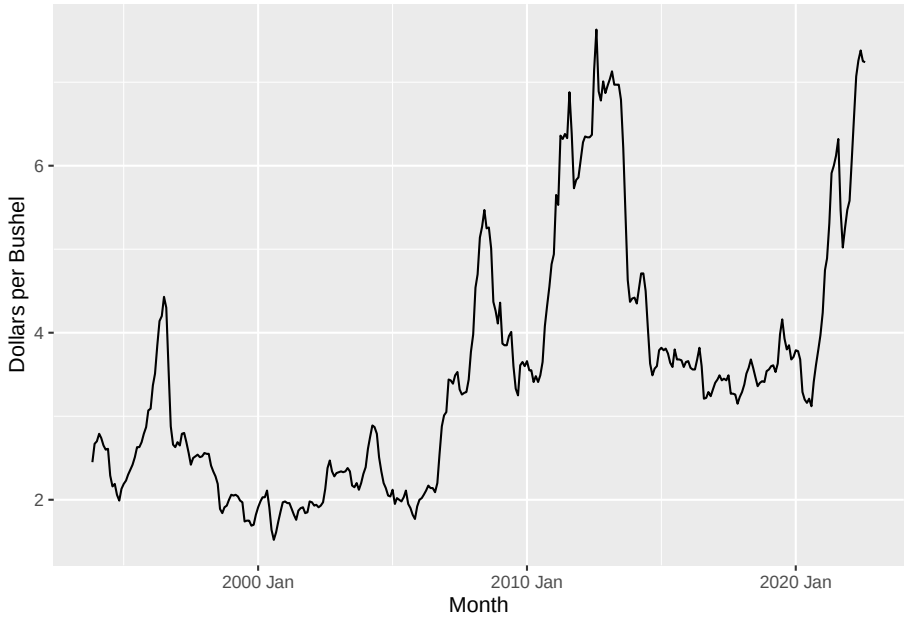

Figure 8.7: Corn Price Received by U.S. Farmers

```
ggplot(mnth_mean, aes(x = mnth, y = dev_mean)) +
 geom_bar(stat='identity',fill = "#FF6666") +
 xlab("") + ylab("Dollars per Bushel") +
 scale_x_discrete(limits=mnth_mean$mnth)
```

8.2 Seasonality Consequences

Suppose we are interested in how the U.S. dollar - Mexican Peso exchange rate affects the monthly price of strawberries. The exchange rate is the number of Pesos required to purchase one U.S. dollar and so a higher value implies a stronger U.S. dollar. A stronger dollar reduces the demand for U.S. strawberries in the global market and this results in a lower U.S. dollar price of strawberries. We can test this theory by regressing the U.S. price of strawberries on the U.S. dollar - Mexican Peso exchange rate.

Figure 8.2 reveals a trend in the price of strawberries and so to avoid a spurious regression due to non-stationarity of the data, we will regress the first difference in the price of strawberries on the first difference in the exchange rate. Let's also add the first difference of the price of grapes as an extra explanatory variable. We do this because the U.S. grape price reflects overall consumer demand for

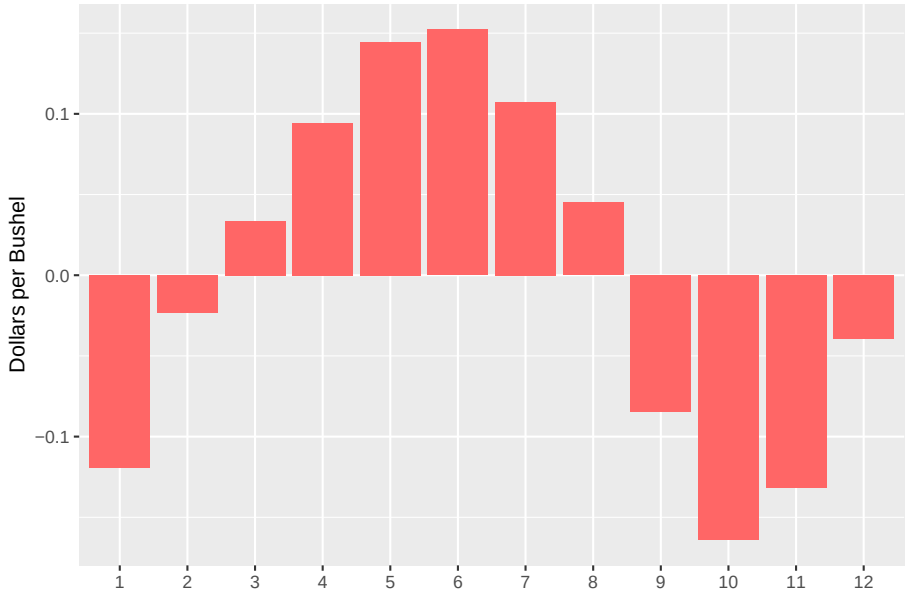

Figure 8.8: Monthly Percentage corn Price Deviations from Annual Average Price.

fruit (U.S. grape production is about six times as large as U.S. strawberry production). Finally, let's add as explanatory variables two lags of the differenced price of strawberries to reduce potential autocorrelation.

The *fruit* data frame which we imported in the previous section contains the U.S. dollar - Mexican Peso exchange rate. To proceed we create first difference variables and then run the regression.

```
fruit <- fruit %>% mutate(D.strawberry = difference(strawberry),
                          D.grape = difference(grape),
                          D.ex_mex = difference(ex_mex),
                          L.D.strawberry = lag(D.strawberry),
                          L2.D.strawberry = lag(D.strawberry,2))

fmod1 <- lm(D.strawberry~D.grape+D.ex_mex+L.D.strawberry+L2.D.strawberry,data=fruit)
summary(fmod1)
```

```
##
## Call:
## lm(formula = D.strawberry ~ D.grape + D.ex_mex + L.D.strawberry +
##     L2.D.strawberry, data = fruit)
```

```
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.30302 -0.16082  0.03529  0.16227  0.82773
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      0.0006139  0.0182404   0.034  0.97318
## D.grape          0.2210951  0.0535678   4.127 5.06e-05 ***
## D.ex_mex         0.0075194  0.0396839   0.189  0.84988
## L.D.strawberry   0.1987167  0.0635748   3.126  0.00199 **
## L2.D.strawberry -0.0335692  0.0622632  -0.539  0.59028
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2849 on 241 degrees of freedom
##   (3 observations deleted due to missingness)
## Multiple R-squared:  0.1243, Adjusted R-squared:  0.1098
## F-statistic: 8.556 on 4 and 241 DF,  p-value: 1.798e-06
```

We see that the differenced U.S. dollar - Mexican peso exchange rate ("D.ex_mex") has a positive sign rather than the anticipated negative sign. However, the estimated coefficient is not statistically significant and so the incorrect sign should not be a cause of concern. Notice that the estimated coefficient on the differenced price of grapes ("D.grape") is positive and highly significant. It seems that including grapes as a control variable is a good idea. Let's check the residuals to ensure there are no significant problems with autocorrelation.

We first create a *tsibble* consisting of the date and the regression residuals. We then use the *ACF* function together with the *autoplot()* function to generate the correlogram. Figure 8.9 below shows that substantial autocorrelation remains in the residuals, despite first differencing of the data and including lags as explanatory variables. Clearly the regression is mis-specified.

```
residuals <- fruit %>%
  slice(-c(1:3)) %>%
  dplyr::select(month) %>%
  mutate(resid = fmod1$residuals) %>%
  as_tsibble(index=month)
```

```
residuals  %>%
  ACF(resid) %>%
  autoplot() + labs(title = "Correlogram of Strawberry Price Regression Residuals")
```

Figure 8.9 is a classic case of seasonality in the residuals. The residuals are

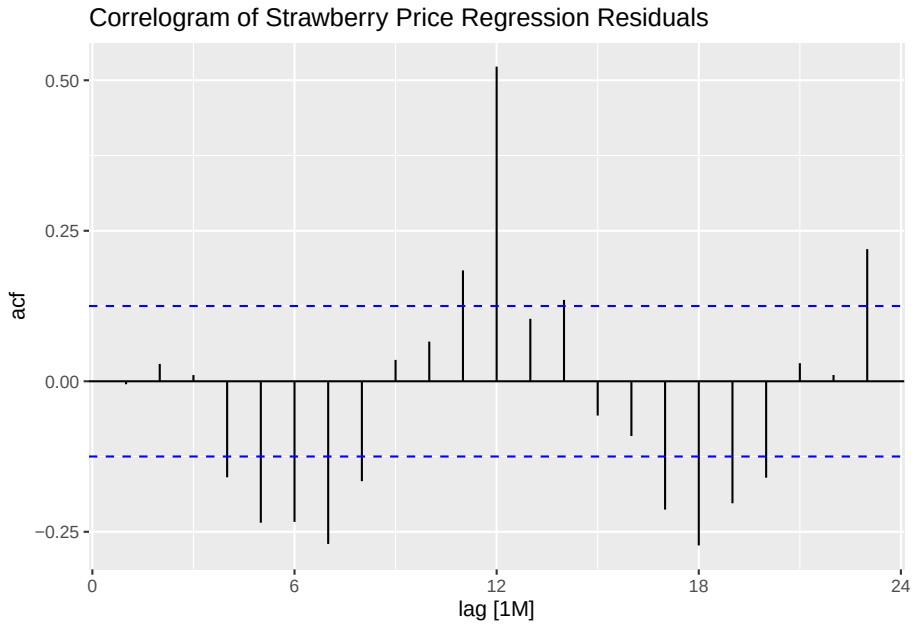

Figure 8.9: Correlogram of Strawberry Price Regression Residuals.

positively correlated with a 12 month lag and are negatively correlated with a six month lag. This is expected because the summer price in a particular year are positively correlated with the summer price of the previous year, and is negatively correlated with the most recent winter price.

Two types of problems emerge when the residuals of a regression are seasonal. The first problem is serious because failing to control for seasonality implies biased estimated coefficients due to a spurious regression. The second problem is less serious because in this case it is the standard errors of the estimated coefficients and not the actual estimates which are biased. In particular, the t values are likely to be inflated upward, which means that we may conclude a variable is statistically significant when in fact it is not. Both of these problems potentially explain the "incorrect/inconclusive" results concerning the exchange rate impact, which were presented above.

These two problems which arise from not controlling for seasonality are examined in turn.

8.2.1 Spurious Regression

Data which is seasonal is non-stationary because the mean of the series is time dependent. Similar to the case of stochastic trends, non-stationary which results

from seasonality typically results in a spurious regression. For the strawberry regression, the positive and highly statistically significant coefficient on the grape variable is due to a common seasonal pricing pattern for strawberries and grapes. To avoid a spurious outcome the seasonality should be removed before using the time series data in a regression.

In the previous section we constructed seasonally adjusted prices for strawberries and grapes. Let's add first differences and lags to the *fruit2* data frame and then rerun the previous regression using seasonally adjusted price differences rather unadjusted price differences.

```
fruit2 <- fruit2 %>% mutate(D.Strawberry = difference(Strawberry),
                D.Grape = difference(Grape),
                D.ex_mex = difference(ex_mex),
                L.D.Strawberry = lag(D.Strawberry),
                L2.D.Strawberry = lag(D.Strawberry,2))

fmod2 <- lm(D.Strawberry~D.Grape+D.ex_mex+L.D.Strawberry+L2.D.Strawberry,data=fruit2)
summary(fmod2)
```

```
##
## Call:
## lm(formula = D.Strawberry ~ D.Grape + D.ex_mex + L.D.Strawberry +
##     L2.D.Strawberry, data = fruit2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.69070 -0.05476  0.00500  0.04812  0.58185
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      0.003581   0.006898   0.519  0.60415
## D.Grape          0.130035   0.042486   3.061  0.00246 **
## D.ex_mex        -0.005921   0.014967  -0.396  0.69275
## L.D.Strawberry  -0.116024   0.062775  -1.848  0.06579 .
## L2.D.Strawberry -0.139931   0.062419  -2.242  0.02589 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1076 on 241 degrees of freedom
##   (3 observations deleted due to missingness)
## Multiple R-squared:  0.06172,    Adjusted R-squared:  0.04615
## F-statistic: 3.963 on 4 and 241 DF,  p-value: 0.00391
```

In the revised regression we see that the estimated coefficient on the grape variable is much smaller in magnitude and although still significant it's p value

is much larger as compared to the case where the price data was not seasonally adjusted. In this revised case the common seasonal pattern has been eliminated and so the regression is no longer spurious. Notice that the estimated coefficient on the exchange rate value now has the correct negative sign but it is still not statistically significant.

8.2.2 Inflated t Values

Suppose we are interested in how the U.S. - Mexico exchange rate impacts the price of corn received by U.S. farmers. We expect this exchange rate to be a significant explanatory variable because Mexico has a relatively high share of U.S. corn exports. Once again, theory tells us to expect a negative coefficient because a larger value for the exchange rate implies a stronger dollar and thus a lower export price due to reduced demand. Let's regress the differenced price of corn on the differenced exchange rate.

```
corn <- corn %>% mutate(D.corn=difference(corn),
                D.ex_mex=difference(ex_mex))
cmod1 <- lm(D.corn~D.ex_mex,data=corn)
summary(cmod1)
```

```
##
## Call:
## lm(formula = D.corn ~ D.ex_mex, data = corn)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.87008 -0.08250  0.00935  0.09387  0.78918
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.01778    0.01148   1.549  0.12242
## D.ex_mex    -0.07931    0.02712  -2.925  0.00367 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2119 on 343 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.02434,    Adjusted R-squared:  0.02149
## F-statistic: 8.555 on 1 and 343 DF,  p-value: 0.003675
```

These regression results show that the difference exchange rate variable (" D.ex_mex") has the anticipated negative sign and it is statistically significant. Given that we did not control for seasonality, can these regression results can

be trusted? There is no seasonality in the exchange rate and so we should not expect the type of spurious results which emerged in the previous section. But can we trust the estimated standard error and associated p value for the exchange rate variable?

To answer this question let's look at the correlogram of the regression residual. In Figure 8.10 the large positive spike at one lag indicates autocorrelation which has not been accounted for since no lags of the differenced corn price was included as an explanatory variable. As well, the small spike at a 12 month lag is indicative of weak seasonality. The negative spikes at the 6 month lag are for the most part not statistically significant. Overall we can conclude that these regression residuals have positive autocorrelation.

```
residuals2 <- corn %>%
  slice(-1) %>%
  dplyr::select(month) %>%
  mutate(resid = cmod1$residuals) %>%
  as_tsibble(index=month)
```

The Appendix of this chapter establishes the following important result. If the explanatory variable is autoregressive (i.e., current values depend on past values) and if there is positive autocorrelation, then the estimated standard errors are biased downward and the associated t values are inflated upwards. This describes the current situation and many other commodity price regression scenarios. Thus, based on the results established in the Appendix we can conclude that the t value of the estimated coefficient of the exchange rate variable is biased upward. This outcome is important because it may potentially reverse our earlier conclusion that the U.S. - Mexico exchange rate is a significant explanatory variable in the corn price regression.

```
residuals2  %>%
  ACF(resid) %>%
  autoplot() + labs(title = "Correlogram of Corn Price Regression Residuals")
```

Let's redo the previous regression but this time with the price of strawberries rather than the price of corn. The first step is to generate the regression residuals.

```
fmod3 <- lm(D.strawberry~D.ex_mex,data=fruit[-1,])
residuals3 <- fruit2 %>%
  slice(-1) %>%
  dplyr::select(month) %>%
  mutate(resid = fmod3$residuals) %>%
  as_tsibble(index=month)
```

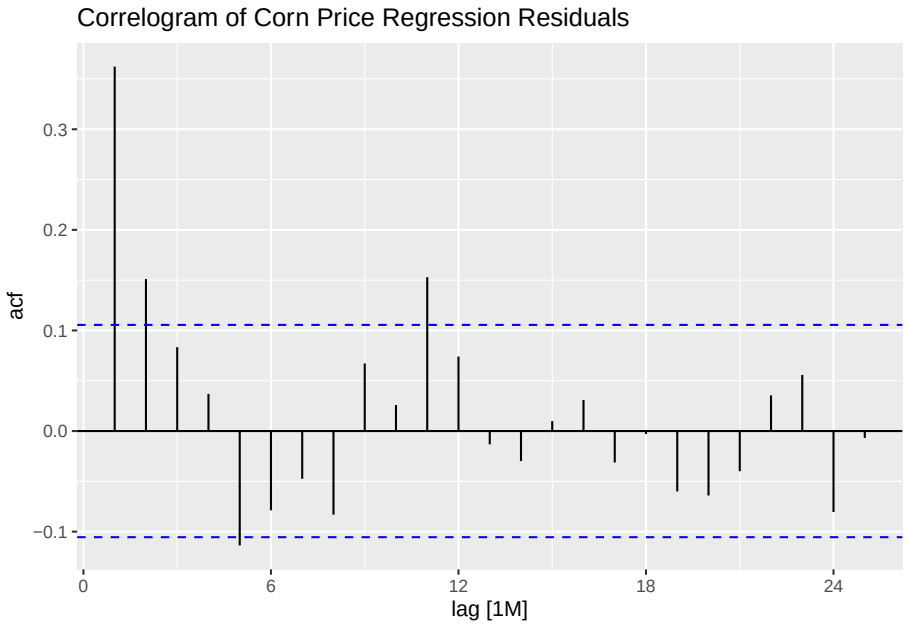

Figure 8.10: Correlogram of Corn Price Regression Residuals.

The correlogram in Figure 8.11 shows that the strong seasonality in the price of strawberries results in positive correlation at 12 month lags and negative correlation at 6 months lags. In this case the autocorrelation is no longer unambiguously positive. As a result we can no longer use the results from the regression to conclude that the t value on the estimated coefficient on the exchange rate variable is necessarily biased upward. This result is important because most introductory econometric textbooks automatically assume that not controlling for seasonality necessarily biases the t statistics upward.

```
residuals3  %>%
  ACF(resid) %>%
  autoplot() + labs(title = "Correlogram of Strawberry Price Regression Residuals")
```

8.3 ARIMA

Time series data which have trends, seasonality and various types of autocorrelation are often modeled using an autoregressive integrated moving average (ARIMA) framework. ARIMA models are most commonly used for forecasting (e.g., see Chapter 9 of Hyndman and Athanasopoulos [2021]) but they can

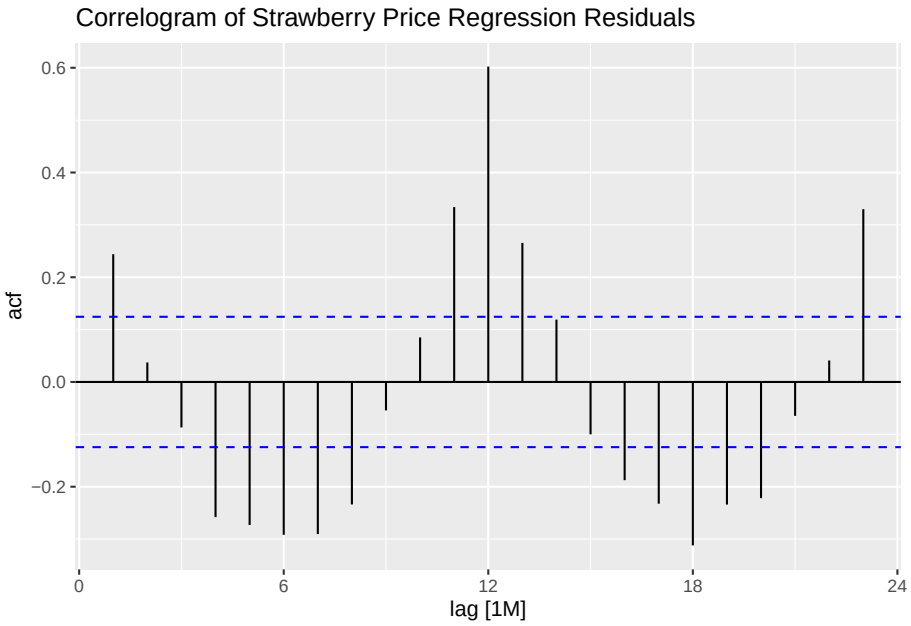

Figure 8.11: Correlogram of Strawberry Price Regression Residuals.

also be used for time series econometrics. There are several different types of ARIMA models:

- Standard ARIMA
- Seasonal ARIMA (SARIMA)
- ARIMA with exogenous regressors (ARIMAX)
- A seasonal ARIMA with exogenous regressors (SARIMAX)
- A linear model with an ARIMA or SARIMA error term.

An autoregressive moving average (ARMA) model is one which uses lags of the variable and lags of the error term as way to model the autocorrelation in the time series. An $ARMA(p,q)$ model includes as explanatory variables p lags of the variable in question and q lags of the regression error. Thus, an ARMA(2,1) can be expressed as $z_t = \alpha + \beta_1 z_{t-1} + \beta_2 z_{t-2} + \epsilon_t + \theta\epsilon_{t-1}$. When using an ARMA model the data must be stationary (i.e., integrated of order zero). An ARIMA model is the same as an ARMA model except due to the presence of a unit root the data must be differenced d times to make the data stationary before the ARMA model can be estimated. Thus, an ARIMA(1,1,1) model can be expressed as

$$z_t - z_{t-1} = \beta(z_{t-1} - z_{t-2}) + \epsilon_t - \epsilon_{t-1} + \theta(\epsilon_{t-1} - \epsilon_{t-2}). \tag{8.1}$$

8.3.1 Simulated Data and ARIMA(1,1,1)

In this section we introduce ARIMA models using simulated random walk with drift data. This data was previously created and can now be imported as *data_sim.RDS*. The data consists of a dependent variable, z_t, and an explanatory variable, x_t. A third variable, Z_t, is a seasonally adjusted version of z_t. All three variables were generated assuming an ARIMA(1,1,1) stochastic process.

Figure 8.12 is a plot of the three correlated time series, Z_t, z_t and x_t. Note that :

- Z_t alone implies a standard ARIMA.
- z_t alone implies a seasonal ARIMA
- A combination of Z_t and x_t implies an ARIMAX
- A combination of z_t and x_t implies a SARIMAX

```
data_sim <- readRDS(here("data/ch7","data_sim.RDS"))

long <- melt(as_tibble(data_sim), id.vars = "quarter")

ggplot(long, aes(x = quarter, y = value, color = variable)) +
  geom_line() + xlab("")
```

Our first task is to estimate a standard ARIMA model using the Z_t non-seasonal time series. It is fairly obvious from Figure 8.12 that the data must be differenced to make the data stationary. In this case (and in most other cases) using $d = 1$ order of differencing is sufficient to make Z_t stationary. Our next task is to choose values for the p and q parameters of the $ARIMA(p, d, q)$ model. This issue is dealt with below. For now assume $p = 1$ and $q = 1$ since we know the data was randomly simulated using an ARIMA(1,1,1) model. This assumption means we will estimate equation (8.1).

Only in the restricted case of $q = 0$ can a standard ARIMA model be estimated using regular least squares regression. This is because the MA component of the ARIMA model makes the regression non-linear (see HARVEY and PHILLIPS [1979] for details). Knowing this, let's use the *ARIMA()* function to estimate the standard ARIMA model for the Z_t variable with $p = d = q = 1$. When setting up the *ARIMA()* function it is necessary to use $Z \sim pdq(1, 1, 1) + PDQ(0, 0, 0)$. The last term, *PDQ(0,0,0)* is used to specify the parameters of the seasonal ARIMA. Even though we are not estimating a seasonal ARIMA at this point in the analysis it is necessary to use *PDQ(0,0,0)* when estimating a standard ARIMA model. This is because if *PDQ(0,0,0)* is omitted then the parameters of the seasonal ARIMA will automatically be selected rather than restricted to have zero values.

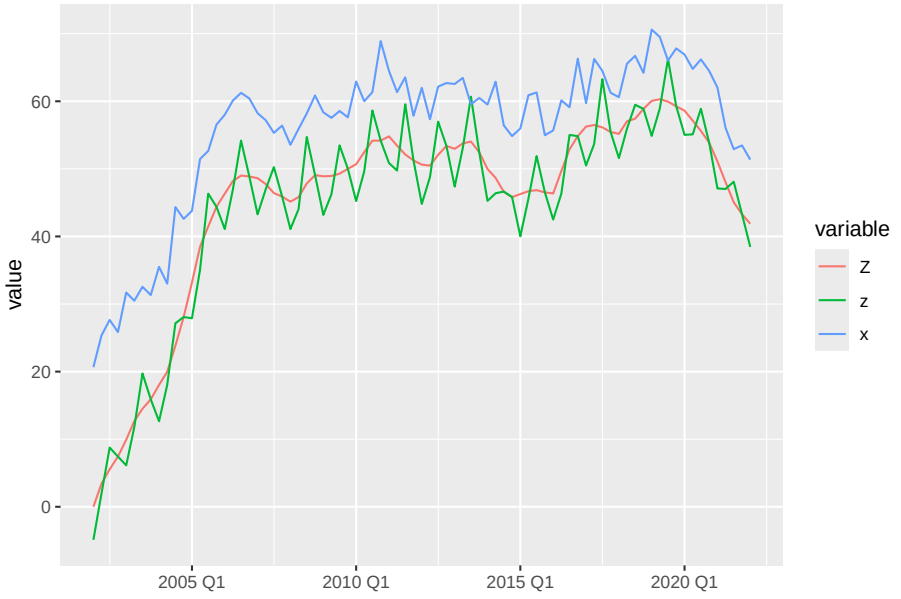

Figure 8.12: Simulated Correlated ARIMA Data

```
fit1 <- data_sim %>%
  model(ARIMA(Z~pdq(1,1,1)+PDQ(0,0,0)))
report(fit1)
```

```
## Series: Z
## Model: ARIMA(1,1,1)
##
## Coefficients:
##          ar1     ma1
##       0.8336  0.1078
## s.e.  0.0742  0.1377
##
## sigma^2 estimated as 0.9643:  log likelihood=-111.73
## AIC=229.47   AICc=229.78   BIC=236.61
```

The above regression results for equation (8.1) reveal that $\hat{\beta} = 0.8336$ and $\hat{\theta} = 0.1078$. Note that additional analysis is required in order to recover the intercept of the non-differenced ARIMA equation, α.

8.3.2 Seasonal Dummies vs Seasonal Differencing

In this section we will model the seasonality which is present in the z_t time series (see Figure 8.12). We know from Section 8.2 and Figure 8.9 in particular that the ARIMA procedure is not sufficient to eliminate the autocorrelation which can be attributed to the seasonality in z_t. There are two options for taking the additional step which will eliminate the seasonality and thus the residual autocorrelation. The first option is to use an ARIMAX model where the X_t consists of seasonal dummy variables. The second option is to use an SARIMA model, which means using seasonal differencing in addition to the standard first difference as a way to eliminate the seasonality. These two options are examined in turn.

Seasonal Dummies

The seasonality in the z_t time series is quarterly and so the constructed X will consist of dummies for Q1, Q2 and Q3. Omitting the Q4 dummy implies that the seasonality results must be interpreted relative to Q4. For example, if the estimated coefficient on the Q2 dummy is -0.5 then we can conclude that z tends be 0.5 units lower when in Q2 than when in Q4. Seasonal dummies create unique intercepts for each season, and these unique intercepts ensure the seasonal autocorrelation disappears from the residuals.

Let s_t^1 take on a value of 1 if the current quarter is Q1 and a value of 0 otherwise. The specification for s_t^2 and s_t^3 is similar. Let's create a new data frame by adding seasonal dummies to the original data frame and then difference the full set of variables.

```
data_sim2 <- data_sim %>%
  mutate(qtr = quarter(quarter)) %>%
  mutate(s1 = ifelse(qtr==1,1,0),
         s2 = ifelse(qtr==2,1,0),
         s3 = ifelse(qtr==3,1,0),
         D.z = difference(z),
         D.s1 = difference(s1),
         D.s2 = difference(s2),
         D.s3 = difference(s3)) %>%
  slice(-1)
```

A common mistake is to add s_t^1 through s_t^3 after the time series has been differenced rather than before. If we ignore the autoregressive (AR) and moving average (MA) components the incorrect specification is $z_t - z_{t-1} = \alpha + \gamma_1 s_t^1 + \gamma_2 s_t^2 + \gamma_3 s_t^3 + \epsilon_t - \epsilon_{t-1}$. The correct way to model seasonality with dummies is to difference the dummies along with the variable of interest. Thus, the correct model can be specified as follows.

$$z_t - z_{t-1} = \alpha + \gamma_1(s_t^1 - s_{t-1}^1) + \gamma_2(s_t^2 - s_{t-1}^2) + \gamma_3(s_t^3 - s_{t-1}^3) + \epsilon_t - \epsilon_{t-1} \quad (8.2)$$

There are two ways we can estimate this model using the *ARIMA()* function from the *fable* package. The first is to manually difference z_t and the set of seasonal dummies to ensure the stochastic trend has been eliminated and the data is stationary. To estimate the model this differenced data is used together with the function call:

$$D.z \sim pdq(1,0,1) + PDQ(0,0,0) + 0 + D.s1 + D.s2 + D.s3$$

The second way to estimate the model is to instruct the *ARIMA()* function to difference z. The function knows that if z is differenced then all of the exogenous variables which are included in the function call must also be differenced. In this case we use the non-differenced version of z and the following function call:

$$z \sim pdq(1,1,1) + PDQ(0,0,0) + s1 + s2 + s3$$

The two different methods for estimating the seasonal dummy model can now be implemented. As shown below, the estimated coefficients with the two different methods have similar but not identical values. This is to be expected because even with a standand linear model there are typically small differences in the estimated coefficients if the model is estimated in levels versus first differences.

```
dumarima1 <- data_sim2 %>%
  model(ARIMA(D.z~pdq(1,0,1)+PDQ(0,0,0)+0+D.s1+D.s2+D.s3)) %>%
  report()
```

```
## Series: D.z
## Model: LM w/ ARIMA(1,0,1) errors
##
## Coefficients:
##          ar1      ma1     D.s1     D.s2    D.s3
##       0.9103  -0.7087  -4.9819  -1.7933  4.5829
## s.e.  0.0907   0.1445   0.3978   0.4601  0.3968
##
## sigma^2 estimated as 5.497:  log likelihood=-179.3
## AIC=370.59   AICc=371.74   BIC=384.88
```

```
dumarima2 <- data_sim2 %>%
  model(ARIMA(z~ pdq(1,1,1)+PDQ(0,0,0)+s1+s2+s3)) %>%
  report()
```

```
## Series: z
## Model: LM w/ ARIMA(1,1,1) errors
##
```

```
## Coefficients:
##          ar1      ma1       s1       s2      s3
##       0.8884  -0.6826  -4.9645  -1.8545  4.5565
## s.e.  0.1018   0.1526   0.3975   0.4634  0.3969
##
## sigma^2 estimated as 5.503:  log likelihood=-177.03
## AIC=366.07   AICc=367.24   BIC=380.29
```

Let's add the first difference of the fitted values from the second variation of this model to the *data_sim2* for plotting later in this section.

```
fitted_dummy <- dumarima2 %>% fitted()

data_sim2 <- data_sim2 %>%
 inner_join(fitted_dummy, by="quarter") %>%
  mutate(D.fitted_dum = difference(.fitted)) %>%
  dplyr::select(quarter,z,D.z,D.fitted_dum)
```

Seasonal Differencing

As previously noted, the second option for controlling for seasonality is to use seasonal differencing. With quarterly data seasonal differencing implies $z_t - z_{t-4}$. If the seasonal effects are relatively constant over time, then $z_t - z_{t-4}$ will be free of seasonality. Seasonality may also have an autoregressive and moving average component. In this case we use P and Q to respectively to denote the number of lags in the seasonal differenced variable and the seasonal differenced error term. When a seasonal ARIMA is combined with a regular ARIMA in a multiplicative way (more details below) the notation (with quarterly data) is $ARIMA(p,d,q)(P,D,Q)_4$.

Consider, for example, an $ARIMA(0,0,0)(2,1,1)_4$ model, which can be expressed as

$$\begin{aligned} z_t - z_{t-4} - \rho_1(z_4 - z_{t-8}) - \rho_2(z_8 - z_{t-12}) \\ = \epsilon_t - \epsilon_{t-4} + \psi(\epsilon_{t-4} - \epsilon_{t-8}) \end{aligned} \tag{8.3}$$

It is easier to write these more complex ARIMA models using back shift notation, which includes $L^i z_t = z_{t-i}$ and $(1-L^i)z_t = y_t - z_{t-i}$. With this notation equation (8.3) can be rewritten as $(1-\rho_1 L^4 - \rho_2 L^8)(1-L^4)z_t = (1+\psi L^4)(1-L^4)\epsilon_t$. The multiplicative properties of the back shift operator means that the seasonal model which is given by equation (8.3) can be combined with a single differencing of z_t. Doing this will ensure the resulting variable is free of both the stochastic trend and seasonality. This enhanced $ARIMA(0,1,0)(2,1,1)_4$ model can be expressed as $(1-\rho_1 L^4 - \rho_2 L^8)(1-L^4)(1-L)z_t = (1+\psi L^4)(1-L^4)\epsilon_t$.

Let's use the z_t variable in the simulated data to estimate an $ARIMA(1,1,1)(0,1,0)_4$ model. The results of this model can be compared to the results of the previous

seasonal dummy model. The only difference between the two models is that one uses seasonal dummies to control for the seasonality and the other one uses seasonal differencing.

```
seasarima <- data_sim2 %>%
  model(ARIMA(z~pdq(1,1,1)+PDQ(0,1,0))) %>%
  report()
```

```
## Series: z
## Model: ARIMA(1,1,1)(0,1,0)[4]
##
## Coefficients:
##           ar1     ma1
##       -0.2633  0.1850
## s.e.   0.3920  0.3844
##
## sigma^2 estimated as 11.52:  log likelihood=-197.07
## AIC=400.15   AICc=400.49   BIC=407.1
```

```
fitted_season <- seasarima %>% fitted()
```

Let's once again add the difference in the fitted values from the estimated ARIMA model to the *data_stim2* data frame.

```
data_sim2 <- data_sim2 %>%
 inner_join(fitted_season, by="quarter") %>%
  mutate(D.fitted_seas = difference(.fitted)) %>%
  dplyr::select(quarter,D.z,D.fitted_dum,D.fitted_seas)
```

Comparison

We can now plot the differenced z_t data along with the differenced fitted values from the seasonal dummy model and the fitted values from the seasonal differencing model. Figure 8.12 shows that the seasonal dummy model fits the z_t time series significantly better than the seasonal ARIMA model. To understand this outcome it is useful to write the specific equation for the $ARIMA(0,1,0)(0,1,0)_4$ model as $z_t - z_{t-1} = \rho(z_{t-4} - z_{t-5}) + \epsilon_t$. This model has only one parameter to estimate whereas the seasonal dummy model has three parameters. The two additional parameters results in a better fit for the seasonal dummy model.

```
long2 <- melt(as_tibble(data_sim2), id.vars = "quarter")

ggplot(long2, aes(x = quarter, y = value, color = variable)) +
  geom_line() + xlab("")
```

```
## Warning: Removed 2 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

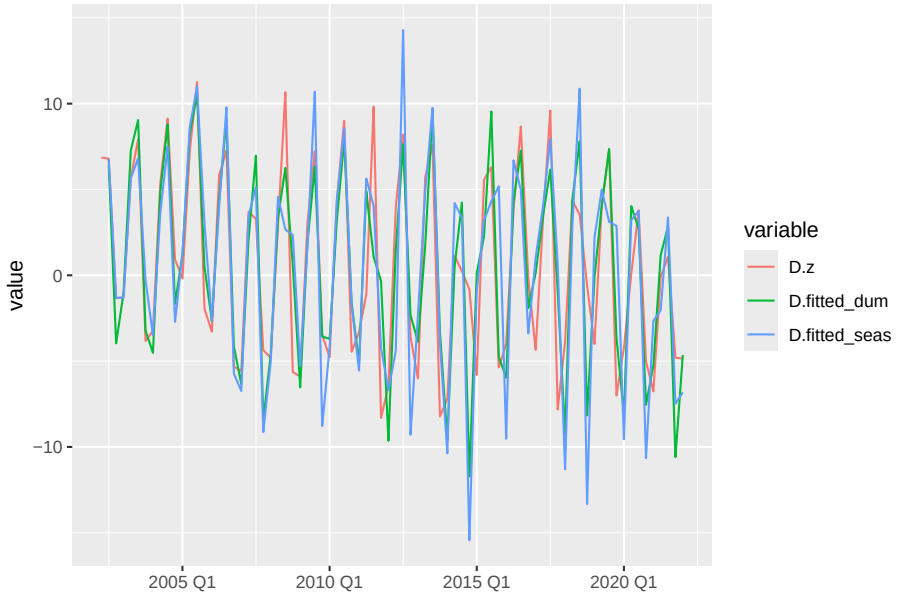

Figure 8.13: Fitted Values for Dummy and Difference Seasonality

8.3.3 Linear Model with ARIMA Errors

Using the simulated data, suppose we are interested in knowing how the x_t variable impacts the z_t variable. This requires us to add an exogenous variable to an ARIMA or an SARIMA. In the previous section we have already done this by adding exogenous seasonal dummies to an ARIMA model. If you look back to the results of that section you will notice that calling the *ARIMA()* function returns as its first line "Model: LM w/ ARIMA(1,0,1) errors". This tells us that by including a set of exogenous variables in the *ARIMA()* function, the function automatically assumes the user wants a linear model with ARIMA errors rather than an ARIMA with exogenous variables included (i.e., an ARIMAX).

This distinction is important. Recall that equation (8.1) is the expression for an ARIMA(1,1,1) model. Let's add the difference of the x_t exogenous variable to create an ARIMAX model.

$$z_t - z_{t-1} = \beta(z_{t-1} - z_{t-2}) + \gamma(x_t - x_{t-1}) + \epsilon_t - \epsilon_{t-1} + \theta(\epsilon_{t-1} - \epsilon_{t-2}). \quad (8.4)$$

It is possible to estimate equation (8.4) using a non-linear estimation procedure. However, it is not possible to instruct the *ARIMA()* function to estimate equation (8.4). As noted, the *ARIMA()* function will instead estimate a linear model with ARIMA(1,1,1) errors.

Before showing the expression for an linear model with ARIMA errors it is useful to point out why the *ARIMA()* function cannot be programmed to estimate equation (8.4). We are interested in knowing the total impact of a change in $x_t - x_{t-1}$ on the change in $z_t - z_{t-1}$. An estimate of the γ coefficient in equation (8.4) does not give this total change because it fails to account for the lag of the $z_t - z_{t-1}$ on the right side of equation (8.4). The linear model with ARIMA errors addresses this problem.

To keep things simple, let's write out the expression for a linear model with $ARIMA(1,0,0)$ errors and no intercept. The core pair of equations are $z_t = \phi x_t + \epsilon_t$ where $\epsilon_t = \rho\epsilon_{t-1} + e_t$. Now substitute $\epsilon_{t+1} = \rho\epsilon_t + e_{t+1}$ into $z_{t+1} = \phi x_{t+1} + \epsilon_{t+1}$ and then solve the resulting expression for ϵ_t to obtain:

$$\epsilon_t = \frac{z_{t+1} - \phi x_{t+1} - e_{t+1}}{\rho}$$

Now substitute this expression into $z_t = \phi x_t + \epsilon_t$ and rearrange to obtain:

$$z_t - \rho z_{t-1} = \phi(x_t - \rho x_{t-1}) + e_t \tag{8.5}$$

Equation (8.5) shows that the estimated ϕ coefficient is a measure of the full effect of a change in x_t on the change in z_t. Even though this equation does not have a MA component, it is still non-linear in its coefficients and so must be estimated with a non-linear procedure. The *ARIMA()* function provides this procedure when an exogenous variable is included. Keep Equation (8.5) in mind when using the *ARIMA()* function to estimate a seasonal dummy model since the coefficients on the seasonal dummies will be non-linear in the same way the coefficients on x_t in Equation (8.5) are non-linear.

Let's use the simulated data for the seasonal z_t and non-seasonal x_t to estimate a linear model with $ARIMA(1,1,1)(0,1,0)_4$ errors. The estimated model as shown below accounts for seasonality through seasonal differencing and accounts for the exogenous x_t variable in a way which ensures that the 0.2612 estimated coefficient on the x_t variable is a measure of the total impact of x_t on z_t.

```
modsim <- data_sim %>%
  model(ARIMA(z~pdq(1,1,1)+PDQ(0,1,0)+x)) %>%
  report()
```

```
## Series: z
## Model: LM w/ ARIMA(1,1,1)(0,1,0)[4] errors
##
```

```
## Coefficients:
##           ar1     ma1       x
##       -0.3806  0.0912  0.2612
## s.e.   0.2017  0.1998  0.1047
##
## sigma^2 estimated as 10.54:  log likelihood=-195.84
## AIC=399.69   AICc=400.25   BIC=409.01
```

8.4 Case Studies

In this section we use the regression with ARIMA errors model to estimate economic relationships in two separate case studies, both of which involve monthly seasonal data. The first case study involves regressing U.S. monthly energy consumption on a monthly temperature anomaly and a monthly price index for electricity. Of interest is how a temperature shock and/or a shock in the price of electricity impacts energy consumption. The second case study involves regressing the producer price for U.S. hogs on the U.S. price of crude oil. These variables are indirectly linked through the U.S. - Canada exchange rate.

8.4.1 Energy Consumption

Energy consumption in the U.S. is highest in the hot summer months and cold winter months. The price of electricity has a similar intertemporal pattern. Energy consumption and the price of electricity are jointly determined (i.e., endogenous) and so ideally this data should be analyzed using a vector autoregression (see Chapter 8). Nevertheless, for this chapter we will proceed with the assumption that electricity prices affect energy consumption but not vice versa.

We begin by reading in the data from the saved *RDS* file. As usual, we will plot the data before beginning the formal analysis. The top two panels of Figure 8.14 below reveal strong seasonality in energy consumption and the price of electricity. Deterministic or stochastic time trends may also be present but they are not obvious. At the very least the seasonality must be removed to ensure the data is stationary and thus spurious regression outcomes avoided prior to using the data in a regression model. Hyndman and Athanasopoulos [2021] recommend using a Box-Cox transformation if the variance of the time series is excessive. Excess variance is not present in the current data and so the Box-Cox adjustment will not be used.

```
energy <- readRDS(here("data/ch7","energy_temp.RDS"))

long_energy <- as_tibble(energy) %>%
  melt(id.vars = "month")
```

```
long_energy %>%
  ggplot(aes(x = month, y = value)) +
  geom_line() +
  facet_grid(vars(variable), scales = "free_y") +
  labs(title = "Energy Consumption, Temperature Anomaly and Electricity Price",
       y = "")
```

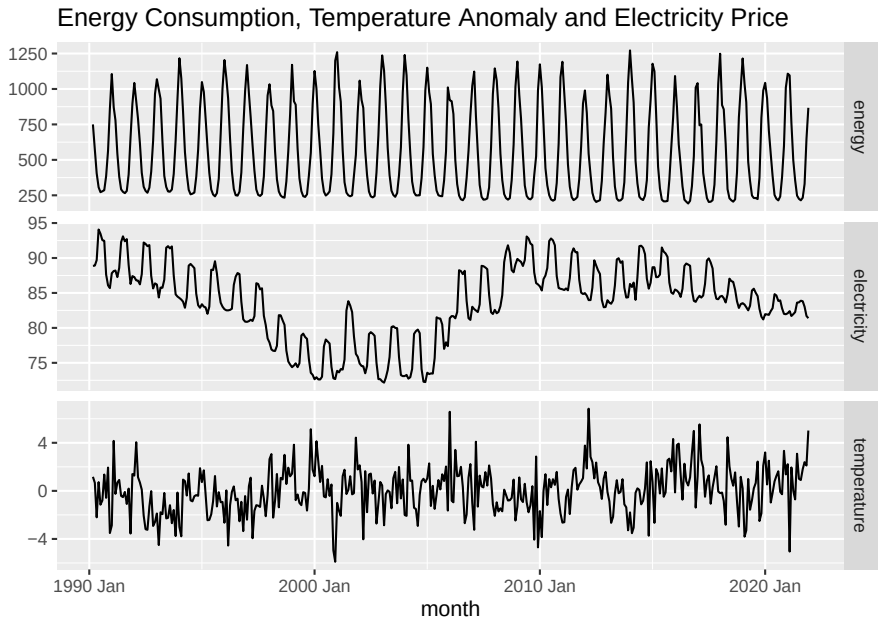

Figure 8.14: Variables in Energy Regression

Chapter 9 of Hyndman and Athanasopoulos [2021] describe best practices for choosing the parameters of the ARIMA and SARIMA models. If the data is non-stationary due to stochastic trends and is also seasonal then regular and seasonal differencing must be used to achieve stationarity. The next step is to choose the $\{p, q, P, Q\}$ parameters according to a pre-determined model selection criteria. Hyndman and Athanasopoulos [2021] recommend choosing the parameters which minimize the AIC. After estimating the model with the optimized level of differencing and parameters Hyndman and Athanasopoulos [2021] recommend checking to ensure that the regression residuals are free from autocorrelation.

For these case studies we will avoid using the lengthy Hyndman and Athanasopoulos [2021] procedures by exploiting the automatic parameter selection feature of the *ARIMA()* function. Specifically, if the *pdq()* and *PDQ* functions are omitted when the *ARIMA()* function is called, then the full differencing and

parameter selection procedure of Hyndman and Athanasopoulos [2021] is automatically implemented. If this automatic procedure was to be used with the previous simulation data the function call would be simply $ARIMA(z \sim x)$.

Seasonal Dummies vs Seasonal Differencing

Let's estimate two alternative linear models with ARIMA errors for the energy consumption case study. The first uses seasonal dummies and the second uses seasonal differencing to control for seasonality. In the first case *PDQ(0,0,0)* must be added to the *ARIMA()* function call to ensure that seasonal differencing is not automatically implemented.

To prepare the data seasonal dummies we can add an month indicator variable to the energy consumption data frame.

```
energy <- energy %>%
  mutate(m = month(month)) %>%
  as_tsibble(index=month)
```

The pair of ARIMA models can now be estimated. Both models are estimated as a *mable*, which is a feature of the *fable* package which allows multiple models to be estimated and tidy results presented. Notice from the output below that with the seasonal dummies, an $ARIMA(2,0,0)$ is optimal, and with the seasonal differencing an $ARIMA(1,0,0)(1,1,2)_{12}$ is optimal. In the previous section when seasonal differencing was used there was only one seasonal coefficient because the seasonal AR and MA terms were forced to zero. This gave an estimation advantage to the seasonal dummy model which had three coefficients to be estimated. In this current case the seasonal differencing model has four coefficients because it includes on seasonal AR term and two seasonal MA terms. These four coefficients are still less than the 11 coefficients in the seasonal dummy model. An explicit comparison of the two models is presented later.

```
fit <- energy %>%
  model(
    arimadummy = ARIMA(energy~PDQ(0,0,0)+temperature+electricity+factor(m)),
    arimaseason = ARIMA(energy~temperature+electricity))

fit %>% pivot_longer(everything(), names_to = "Model name",
                     values_to = "Orders")
```

```
## # A mable: 2 x 2
## # Key:     Model name [2]
##   `Model name`                                  Orders
##   <chr>                                        <model>
## 1 arimadummy              <LM w/ ARIMA(2,0,0) errors>
## 2 arimaseason  <LM w/ ARIMA(1,0,0)(1,1,2)[12] errors>
```

Of particular interest in the seasonal dummy and seasonal difference models are the signs and the sizes of the estimated coefficients for the temperature anomaly and the electricity price variables. In the results below the estimated coefficients for the seasonal dummies are not presented in order to save space. Notice that the estimated coefficients for temperature anomaly and electricity price are very close in value across the two models. For temperature anomaly the coefficients are -17.9 and 18.5, and for the electricity price the coefficients are -2.31 and -2.29.

Mean energy consumption over the sample period is 563.08 trillion btu and the mean value of the real electricity price index is 167.02. Using the -17.9 estimated coefficient on the temperature anomaly variable, we see that a one degree Fahrenheit increase above the long term average temperature, averaged over all months, decreases energy consumption by about 3.2 percent. As well, using the -2.31 estimated coefficient on the electricity price variable, we see that if the real price of electricity increases by 10 percent, energy consumption decreases by approximately $-2.31 * 0.1 * 167 =$ -38.41 billion btu. This is equivalent to approximately -38.41/563.08 = 0.0682 or 6.82 percent. Both of these estimates fall in a range which is reasonable to expect.

```
fit %>%
  coef() %>%
  slice(-c(5:15))
```

```
## # A tibble: 11 x 6
##    .model      term         estimate std.error statistic  p.value
##    <chr>       <chr>           <dbl>     <dbl>     <dbl>    <dbl>
##  1 arimadummy  ar1             0.408     0.0533     7.65  1.70e-13
##  2 arimadummy  ar2             0.0139    0.0515     0.269 7.88e- 1
##  3 arimadummy  temperature   -17.9       1.10     -16.3   8.35e-46
##  4 arimadummy  electricity    -2.31      0.727     -3.18  1.62e- 3
##  5 arimadummy  intercept    1321.       60.1       22.0   7.30e-70
##  6 arimaseason ar1             0.358     0.0519     6.90  2.33e-11
##  7 arimaseason sar1           -0.232     0.275     -0.847 3.98e- 1
##  8 arimaseason sma1           -0.518     0.266     -1.95  5.21e- 2
##  9 arimaseason sma2           -0.287     0.218     -1.32  1.89e- 1
## 10 arimaseason temperature   -18.5       1.10     -16.8   1.16e-47
## 11 arimaseason electricity    -2.29      0.761     -3.01  2.83e- 3
```

Model Assessment

Which of the two models (seasonal dummies or seasonal differencing) should be selected as the most preferred? Hyndman and Athanasopoulos [2021] recommend choosing the model specification with the lowest AIC value. The results to follow show that based on the minimum AIC criteria the seasonal differencing model is preferred over the seasonal dummy model.

```
fit %>% pivot_longer(everything(), names_to = "Model name",
                     values_to = "Orders")
```

```
## # A mable: 2 x 2
## # Key:     Model name [2]
##   `Model name`                                  Orders
##   <chr>                                        <model>
## 1 arimadummy              <LM w/ ARIMA(2,0,0) errors>
## 2 arimaseason  <LM w/ ARIMA(1,0,0)(1,1,2)[12] errors>
```

```
glance(fit)
```

```
## # A tibble: 2 x 8
##   .model      sigma2 log_lik   AIC  AICc   BIC ar_roots   ma_roots
##   <chr>        <dbl>   <dbl> <dbl> <dbl> <dbl> <list>     <list>
## 1 arimadummy   1678.  -1952. 3938. 3940. 4005. <cpl [2]>  <cpl [0]>
## 2 arimaseason  1677.  -1903. 3819. 3820. 3847. <cpl [13]> <cpl [24]>
```

Let's assess the status of autocorrelation in the residuals for the pair of models. We begin by comparing the ACF correlograms. Figure 8.15 reveals that there is evidence of autocorrelation in both models. However, the problem is more severe with the first seasonal dummy model.

```
resid <- fit %>% resid()
resid %>%
  ACF(.resid) %>%
  autoplot()
```

The correlograms in Figure 8.15 warrant a formal test for autocorrelation. Let's use the *Ljung-Box* test which was presented in Chapter 4. Recall that the null hypothesis for this test is independence of the observations in the each residual series. A p value less (greater) than 0.05 implies that independence is rejected (not rejected) and we conclude that the series is autocorrelated (not autocorrelated). With seasonal data it is recommended to use *l=24* for the lag parameters. The *dof* parameter should be set equal to the number of parameters being estimated in the ARIMA model. In the first model *ARIMA(2,0,0)* model there are *dof=2* parameters and in the second *ARIMA(1,0,0)(1,1,2)* there are *dof=4* parameters.

The results below show that the residuals from the seasonal dummy model are autocorrelated whereas the residuals from the seasonal differenced model are not autocorrelated. This result is an additional reason why the seasonal differencing model should be chosen over the seasonal dummy model.

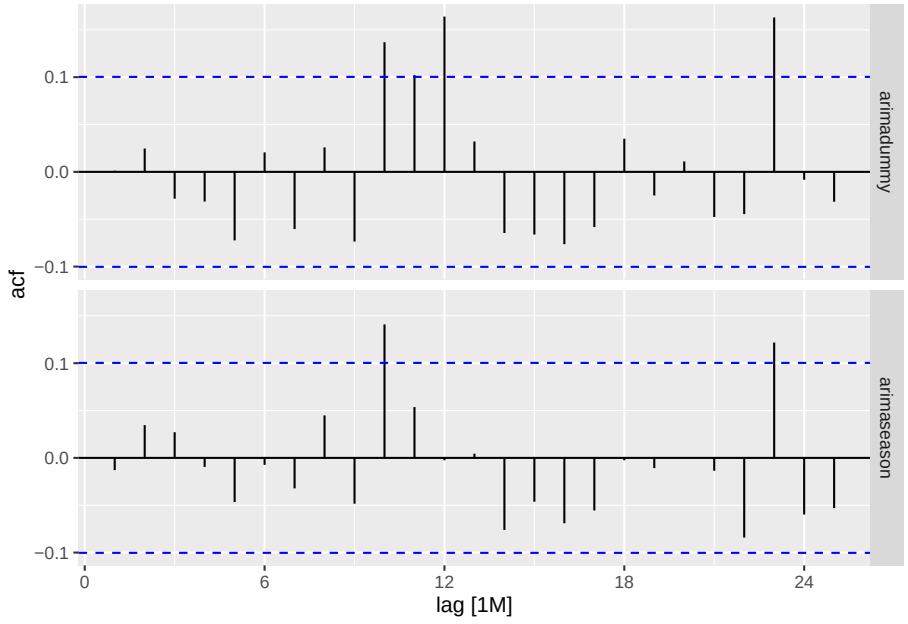

Figure 8.15: Correlogram of Residuals for Seasonal Dummy and Seasonal Difference Models

```
augment(fit) %>%
  features(.innov, ljung_box, lag=24, dof=2)
```

```
## # A tibble: 2 x 3
##   .model      lb_stat lb_pvalue
##   <chr>         <dbl>     <dbl>
## 1 arimadummy     50.1  0.000576
## 2 arimaseason    29.7  0.127
```

```
augment(fit) %>%
  features(.innov, ljung_box, lag=24, dof=4)
```

```
## # A tibble: 2 x 3
##   .model      lb_stat lb_pvalue
##   <chr>         <dbl>     <dbl>
## 1 arimadummy     50.1  0.000217
## 2 arimaseason    29.7  0.0757
```

8.4.2 Hogs and Oil

In this second case study we focus on the relationship between the price of West Texas Intermediate (WTI) crude oil and the price of U.S. slaughter hogs. The U.S. is a large importer of hogs from Canada and so the Canada-U.S. exchange rate is included as a control variable.

Before beginning the analysis it is worth noting that modeling the price of hogs as a linear model with ARIMA errors is supported by theory. Lucia and Schwartz [2002] model the price of electricity, P_t, as an Ornstein–Uhlenbeck mean-reverting stochastic process. Specifically, $P_t = f(t) + X_t$ where $f(t)$ is a deterministic function of time (e.g., trend and seasonality) and X_t is a stochastic state variable. Lucia and Schwartz [2002] assume that $X_t = \kappa X_t dt + \sigma dZ$ is a Brownian motion. For this current analysis it is appropriate to also model the price of hogs as a stationary mean reverting stochastic process rather than a random walk.

For their empirical analysis Lucia and Schwartz [2002] show that the above model can be estimated as $P_t = \alpha + \beta D_t + \sum_{i=2}^{12} \beta_i M_{it} + X_t$ where $X_t = \phi X_{t-1} + u_t$, D_t is a deterministic time trend and M_{it} are monthly dummy variables. This set up is equivalent to a linear model an $ARIMA(1,0,0)$ errors and a set of exogenous variables comprised on a time trend and seasonal dummies. We use a similar approach for modeling the price of hogs except we will allow the error term to have a more general ARIMA format and we will use both seasonal dummies and seasonal differencing.

The price of hogs is an index and so no additional transformation is required. The price of crude oil tends to be rather volatile (i.e., a non-constant variance) and so the log of oil price is used to stabilize this price series. Let's read in the data from the pre-cleaned RDS file and plot it. At first glance it does not appear from Figure 8.16 that the price index for hogs is strongly correlated with the price of oil and the CDN exchange rate. As will be shown below, despite the comparatively high level of volatility in the price of crude oil it still is positively correlated with the price of hogs.

```
hogs <-readRDS(here("data/ch7","hogs.RDS"))
data_long <- melt(hogs, id.vars = "month")
```

```
data_long %>%
  ggplot(aes(x = month, y = value)) +
  geom_line() +
  facet_grid(vars(variable), scales = "free_y") +
  labs(title = "Hog and Crude Oil Prices, CDN Exchange",
       y = "")
```

Figure 8.16 reveals seasonality in the price of hogs. Let's extract this seasonality using the *STL()* function. Figure 8.17 below shows sizeable seasonality which is

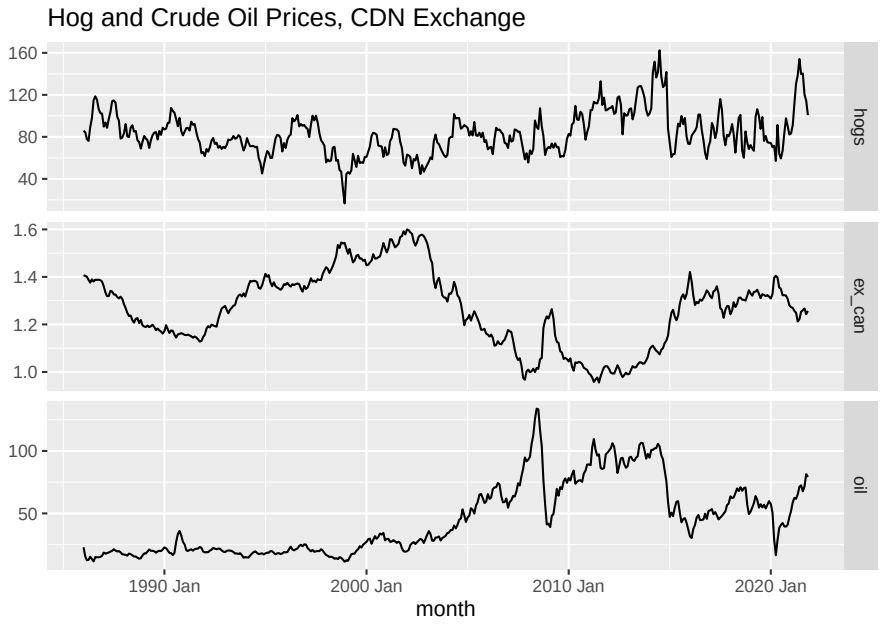

Figure 8.16: Plot of Hog and Oil Prices and CDN Exchange

not constant over time. As will be shown below, this variation in the seasonality over time is the likely cause of autocorrelation which remains in the optimized ARIMA model.

```
hogs_decompose <- hogs[,c("month","hogs")] %>% model(STL(hogs)) %>% components()
```

```
ggplot(hogs_decompose, aes(x = month, y = season_year)) +
  geom_line() +
  xlab("") + ylab("Index")
```

We will estimate four alternative linear models with ARIMA errors. The first three models allow for seasonal differencing and the fourth model uses seasonal dummies instead of seasonal differencing. In the three seasonal differencing models, the first allows the *ARIMA()* function to choose all ARIMA parameters optimally (i.e., to minize the AIC). The second model uses the first difference of the three variables and then allows the *ARIMA()* function to choose all ARIMA parameters optimally. The third model is the same as the second except it uses data which has been both first differenced and seasonally differenced. This last model can be viewed as starting the ARIMA estimation process with stationary data.

In order to estimate this model the differenced variables must be added to the original data set.

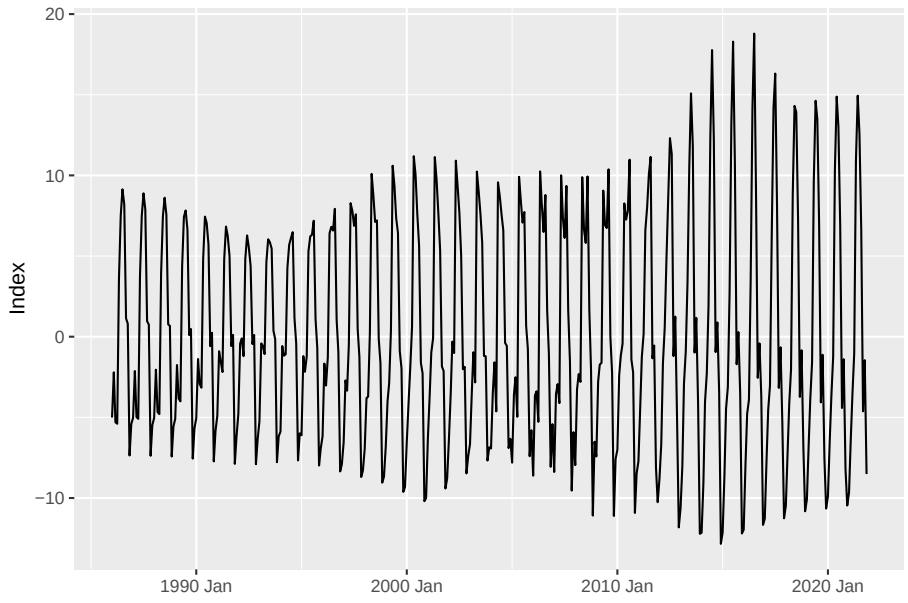

Figure 8.17: Seasonality Component of Hog Price Index

```
data <-  hogs %>%
  mutate(oil = log(oil),
         D.hogs = difference(hogs),
         D.ex_can=difference(ex_can),
         D.oil=difference(oil),
         D12.D.hogs = difference(D.hogs,12),
         D12.D.oil = difference(D.oil,12),
         D12.D.ex_can = difference(D.ex_can,12)) %>%
  slice(-1)

data <- data %>%
  mutate(m = month(month)) %>%
  dummy_cols(select_columns = c("m")) %>%
  as_tsibble(index=month) %>%
  dplyr::select(-m)
```

```
fit1 <- data %>%
  model(optimal  = ARIMA(hogs~oil+ex_can))

fit2 <- data %>%
  model(D.hogs = ARIMA(D.hogs~D.oil+D.ex_can))
```

```
fit3 <- data %>%
  model(D12.D.hogs = ARIMA(D12.D.hogs~0+D12.D.oil+D12.D.ex_can))

fit4 <- data %>%
  model(dummy = ARIMA(hogs~PDQ(0,0,0)+oil+ex_can+m_1+m_2+m_3+m_4+m_5+m_6+m_7+m_8+m_9+m_10+m_11))
```

The table below shows the optimally chosen p,q,P,D values for the four models. In the first model the *ARIMA(1,0,0)(2,0,0)* model is optimal. Rather surprisingly, first differencing and seasonal differencing is not used in this first model. It is for this reason why the second and third models uses data which was differenced prior to he estimation. In the last model which uses seasonal dummies, only one autoregressive term is used. This model is therefore equivalent to the model of Lucia and Schwartz [2002], which was discussed above.

```
order1 <-  fit1 %>% pivot_longer(everything(), names_to = "Model name",
                                 values_to = "Orders") %>% as_tibble()

order2 <-  fit2 %>% pivot_longer(everything(), names_to = "Model name",
                                 values_to = "Orders") %>% as_tibble()

order3 <-  fit3 %>% pivot_longer(everything(), names_to = "Model name",
                                 values_to = "Orders") %>% as_tibble()

order4 <-  fit4 %>% pivot_longer(everything(), names_to = "Model name",
                                 values_to = "Orders") %>% as_tibble()

dplyr::bind_rows(order1,order2,order3,order4)
```

```
## # A tibble: 4 x 2
##   `Model name`                                  Orders
##   <chr>                                        <model>
## 1 optimal      <LM w/ ARIMA(1,0,0)(2,0,0)[12] errors>
## 2 D.hogs       <LM w/ ARIMA(1,0,1)(0,0,2)[12] errors>
## 3 D12.D.hogs   <LM w/ ARIMA(0,0,0)(2,0,0)[12] errors>
## 4 dummy                  <LM w/ ARIMA(1,0,0) errors>
```

To save on space in the table to follow the regression results have been filtered to show only those pertaining to the price of crude oil and the CDN exchange rate. Of particular interest is the estimated coefficient for crude oil. In all four models the sign of this coefficient is positive, which implies positive co-movement. In the first two models the estimated coefficient for oil is statistically significant at the 95 percent level. In the dummy variable model the oil coefficient is statistically significant at the 90 percent level and for the third model where the data was fully differenced before entering the analysis the estimated coefficient

is not statistically significant. Overall, it is fair to claim that movements in the price of oil does impact the price of slaughter hogs.

```
coeff <- dplyr::bind_rows(fit1 %>% coef(),fit2 %>% coef(),fit3 %>% coef(),fit4 %>% coe

coeff %>%
  slice(c(4,5,11,12,15,16,18,19))
```

```
## # A tibble: 8 x 6
##   .model     term         estimate std.error statistic p.value
##   <chr>      <chr>           <dbl>     <dbl>     <dbl>   <dbl>
## 1 optimal    oil              10.4      4.19      2.48   0.0134
## 2 optimal    ex_can          -23.0     18.8      -1.23   0.220
## 3 D.hogs     D.oil            11.7      4.96      2.36   0.0189
## 4 D.hogs     D.ex_can        -21.6     18.9      -1.14   0.253
## 5 D12.D.hogs D12.D.oil         5.76     4.79      1.20   0.230
## 6 D12.D.hogs D12.D.ex_can    -21.9     22.9      -0.958  0.338
## 7 dummy      oil               7.26     3.71      1.96   0.0511
## 8 dummy      ex_can          -28.6     16.8      -1.70   0.0899
```

Recall that the price of oil is in log units. This means that if the estimated coefficient is divided by the value of the hog price index then the resulting value can be intepreted as an elasticity. The mean value of the index is 83, and so for the first model the hog - oil price elasticity is equal to 10.4/83, which is equal to 0.030. This means that a 10 percent increase in the price of oil causes the price of hogs to increase by 3 percent in the long run. This elasticity is slightly higher for the second model and somewhat lower for the last two models. Overall, this level of response of the hog price to the oil price is economically significant. Despite the U.S. being a larger importer of hogs from Canada, the CDN exchange rate is not a significant explanatory variable.

8.5 Conclusions

This chapter began with a brief review of the literature on modeling seasonality in commodity prices. This review was followed by a descriptive analysis of price seasonality in the U.S. fresh strawberry and grape markets. As part of this analysis We examined how the presence of seasonality affects the estimated relationship between the price of fresh strawberries and the U.S. - Mexico exchange rate. We noted that seasonality can create a spurious regression and thus biased coefficient estimates. We also noted that seasonality can depress the standard errors of the estimated coefficients, which in turn causes the associated p values to be inflated. For both of these reasons it is essential that seasonality be controlled when estimating commodity price regression models.

Much of the analysis in this chapter utilized a standard regression with ARIMA errors. This model specification is preferred over an ARIMAX model because the estimated coefficients in a regression model with ARIMA errors is generally more intuitive than the case of an ARIMA or seasonal ARIMA with exogenous regressors (i.e., an ARIMAX or an SARIMAX). When using the *fable* package it is important to keep in mind that if one or more exogenous variables are included in the *ARIMA* function then the default is to estimate a regression model with ARIMA errors rather than an ARIMA model with exogenous regressors.

This chapter concluded with two case studies. The first case study examined how monthly temperature anomalies and the monthly price of electricity impacts monthly energy consumption in the U.S. The regression model with ARIMA errors was estimated with both seasonal dummies and seasonal differencing. Both specifications gave similar results and so a more quantitative comparison was needed for model selection. The AIC criteria was used to select the model with seasonal differencing over the model with seasonal dummies. A formal test of autocorrelation revealed that the seasonal differencing did a reasonably good job of eliminating the autocorrelation.

The second case study used monthly data to examine how the price of U.S. crude oil affects the price of U.S. hogs with the Canada - U.S. exchange rate serving as a control. The exchange rate is dependent on the price of crude oil, and the price of hogs is dependent on the exchange rate. In this case study different approaches were used for automatically selecting the parameters of the ARIMA model. In all cases the estimated impact of crude oil on the exchange rate has the anticipated negative sign and are similar in magnitude.

8.6 Appendix

In this Appendix we use a standard linear regression to determine how autocorrelation in the error term impacts the standard error of the estimated coefficient. In most situations involving weekly or monthly commodity prices the explanatory variable is a first order and possibly higher autoregressive stochastic process; e.g., $x_t = rx_{t-1} + \omega_t$. As well, the error term in the regression model has positive first order and possibly higher autocorrelation; e.g., $\epsilon_t = \rho\epsilon_{t-1} + e_t$. We show below that with these two particular assumptions autocorrelation reduces the standard error of the estimated coefficient. This result that autocorrelation inflates the p values and can therefore result in incorrect conclusions about the statistical significance of the relationship between the two variables also holds more generally but it is beyond the scope of this chapter to prove these more advance results.

Let Y_t and X_t denote a pair of commodity prices. Of interest is an estimate of the coefficient β in the following regression: $Y_t = \alpha + \beta X_t + e_t$. Both prices are assumed to be integrated of order 1 and so the equation is estimated in first difference form:

$$y_t = \beta x_t + \epsilon_t \tag{8.6}$$

where $y_t = Y_t - Y_{t-1}$, $x_t = X_t - X_{t-1}$ and $\epsilon_t = e_t - e_{t-1}$. Of particular interest in this Appendix is how autocorrelation in ϵ_t impacts the standard error of β.

To keep things simple assume that there are n observations and $\epsilon_t = \rho\epsilon_{t-1} + v_t$ where $0 < \rho < 1$. As well, assume that $v_t \sim N(0, \sigma_u^2$ is normally distributed white noise. It follows from this last assumption that $\epsilon_t \sim N\left(0, \frac{\sigma_v^2}{1-\rho^2}\right)$.

An error term with first order autocorrelation has the following variance-covariance matrix:

$$V = \sigma_\epsilon^2 \begin{bmatrix} 1 & \rho & \rho^2 & \rho^3 & ... & \rho^{n-1} \\ \rho & 1 & \rho & \rho^2 & ... & \rho^{n-2} \\ \rho^2 & \rho & 1 & \rho & ... & \rho^{n-3} \\ ... & & & & & \\ ... & & & & & \\ \rho^{n-1} & \rho^{n-2} & \rho^{n-3} & \rho^{n-4} & ... & 1 \end{bmatrix} \tag{8.7}$$

We will derive results concerning the variance of β rather than the standard error of β because the variance and standard error are related by a square root transformation. Using $\beta = (x'x)^{-1}x'y$, an expression for the variance of β is given by

$$var(\beta) = (x'x)^{-1}x'var(y)x(x'x)^{-1} \tag{8.8}$$

Let's begin by analyzing $x'var(y)x$, which is the middle section of equation (8.8). We can substitute $\sigma_\epsilon^2 V$ for $var(y)$ because the variance-covariance of of y_t and ϵ_t are the same. We can therefore use equation (8.7) to write an expression for $var(y)x$ as

$$var(y)x = \sigma\epsilon^2 \begin{bmatrix} x_t + \rho x_{t-1} + \rho^2 x_{t-2} + ... + \rho^{n-1} x_{t-n+1} \\ \rho x_t + x_{t-1} + \rho x_{t-2} + ... + \rho^{n-2} x_{t-n+1} \\ \rho^2 x_t + \rho x_{t-1} + x_{t-2} + ... + \rho^{n-3} x_{t-n+1} \\ ... \\ ... \\ \rho^{n-1} x_t + \rho^{n-2} x_{t-1} + \rho^{n-3} x_{t-2} + ... + x_{t-n+1} \end{bmatrix} \tag{8.9}$$

Equation (8.9) can now be premultiplied by x' to obtain a complete expression for $x'var(y)x$. After gathering terms in the resulting expression, it can be shown that

$$x'var(y)x = x'x + 2\rho \sum_{i=0}^{n-1} x_{t-i}x_{t-1-i} + 2\rho^2 \sum_{i=0}^{n-2} x_{t-i}x_{t-2-i} + ... \tag{8.10}$$

We can now construct an expression for $var(\beta)$. Substituting equation (8.10) into equation (8.8) gives

$$var(\beta) = \sigma_\epsilon^2 (x'x)^{-1} \left[x'x + 2\rho \sum_{i=0}^{n-1} x_{t-i} x_{t-1-i} + 2\rho^2 \sum_{i=0}^{n-2} x_{t-i} x_{t-2-i} + ... \right] (x'x)^{-1} \tag{8.11}$$

Noting that $(x'x)^{-1} = \frac{1}{\sum_{i=0}^{n-1} x_{t-i}^2}$, equation (8.11) can be rewritten as

$$var(\beta) = \frac{\sigma_\epsilon^2}{\sum_{i=0}^{n-1} x_{t-i}^2} \left[1 + 2\rho \frac{\sum_{i=0}^{n-1} x_{t-i} x_{t-1-i}}{\sum_{i=0}^{n-1} x_{t-i}^2} + \frac{2\rho^2 \sum_{i=0}^{n-2} x_{t-i} x_{t-2-i}}{\sum_{i=0}^{n-1} x_{t-i}^2} + ... \right] \tag{8.12}$$

Within equation (8.12) the expression outside of the square brackets, $\frac{\sigma_\epsilon^2}{\sum_{i=0}^{n-1} x_{t-i}^2}$, is an expression for the variance of β in the absence of autocorrelation in the error term. This means that if the expression in the square brackets has a value less than one, then the variance of β is lower with the autocorrelation than without. The purpose of this Appendix is to derive the conditions for which this outcome emerges.

Analytical results can be easily derived if we assume that x_t follows a first order autoregressive process. Specifically, assume that $x_t = r x_{t-1} + \omega_t$. Given that we are assuming x_t is a random variable, it necessarily has a variance-covariance matrix which is given by

$$R = \sigma_x^2 \begin{bmatrix} 1 & r & r^2 & r^3 & ... & r^{n-1} \\ r & 1 & r & r^2 & ... & r^{n-2} \\ r^2 & r & 1 & r & ... & r^{n-3} \\ ... & & & & & \\ ... & & & & & \\ r & r^{n-2} & r^{n-3} & r^{n-4} & ... & 1 \end{bmatrix} \tag{8.13}$$

If we take the expectations of $var(\beta)$ over the x values and substitute in the corresponding values from the R matrix in equation (8.13) we obtain the following equation:

$$var(\beta) = \frac{\sigma_\epsilon^2}{\sigma_x^2} \left(1 + 2\rho r + 2\rho^2 r^2 + ... 2\rho^{n-1} r^{n-1} \right) \tag{8.14}$$

If n is large we can approximate this expression by evaluating it in the limit as $n \to \infty$. In this case equation (8.14) is an infinite geometric sum which reduces to

$$var(\beta) = \frac{\sigma_\epsilon^2}{\sigma_x^2} \frac{1 + \rho r}{1 - \rho r} \tag{8.15}$$

In equation (8.15) the variance of β (and thus also the standard error of β) is lower with autocorrelation if ρ and r both take on positive values. In most daily, weekly or monthly time series these two parameters are generally positive, which supports the claim that autocorrelation results in suboptimally small standard errors and suboptimally high p values. In the rare case where X is independent over time then equation (8.15) shows that autocorrelation in the error term has no impact on the standard error of β.

Chapter 9

Vector Autoregression

In this chapter and the chapters to follow we will use a systems approach when examining food and energy prices. This current chapter utilizes a vector autoregression (VAR) model to analyze the dynamic pricing relationships in a pair of energy commodities, which includes wood chips (used as a source of fuel) and heating oil. We begin our analysis by using a reduced form VAR to exploit the estimated system-wide pricing relationships when forecasting prices for these two commodities. We then turn to our main goal of policy analysis, in which case a structural VAR is required. We use an exclusion restriction to achieve identification of the structural VAR. Specifically, guided by theory, we assume that the price of heating oil "causes" the price of wood chips and not the other way around. This particular exclusion restriction is assessed using the concept of Granger causality. Lastly, we use the estimated structural VAR to conduct impulse response analysis. In this context impulse response means examining how a shock in the price of heating oil (perhaps the result of a permanent carbon tax) affects the pricing path of wood chips.

A VAR is a multivariate statistical model where each variable is a linear function of the lagged values of itself and the other variables in the system. Each variable in a n equation VAR is endogenous, which means that shocks are dynamically transmitted to and from each of the n variables. A VAR model can also include exogenous regressors such as a constant term, a time trend and variables which are fully exogenous (e.g., weather). A VAR model is well suited for the analysis of food and energy prices because of the typically high degree of both vertical and horizontal interdependence of these prices. For example, the price of corn, ethanol and crude oil should be modeled as a system rather than individually because corn is a key ingredient in the production of ethanol, and the prices of ethanol and crude oil are jointly determined due to a high degree of substitution. Other VAR applications in the analysis of food and energy prices include assessing dynamic linkages in global trade, identifying causes of food price inflation and examining the impact of biofuel policy on global food

security.

An important limitation of VAR analysis is that the data must be stationary or trend stationary. A VAR must satisfy long run stability conditions to ensure that a shock to a variable eventually dies out and is forgotten. Commodity prices are often non-stationary and so cannot be included in a VAR without modification. A common modification which usually satisfies the stationary requirement is to estimate a VAR with the data in first differences. Another commonly-used approach is to net inflation out of a commodity's price (i.e., use the real price instead of the nominal price) or to use the price premium or discount relative to a benchmark price rather than the price itself. In a later chapter you will see that if the data is nonstationary but is cointegrated then a vector error correction model (VECM) can be estimated. A VECM shares many of the same features as a VAR but this model is most commonly used for economic analysis rather than for forecasting.

The next section provides a conceptual overview of VAR analysis. Section 9.2 uses simulated data to illustrate the basic features of forecasting with a VAR. Following the usual procedure, we generate the VAR forecast manually (i.e., by programming matrices) before using the dedicated VAR forecasting packages in R. In Section 9.3 we use a two equation VAR for modelling the joint price premiums for wood chips and heating oil. This system also includes the monthly temperature anomaly and seasonal dummies as exogenous variables. Section 9.4 introduces the concept of Granger causality for VAR analysis. Granger causality is used to test if a particular variable in the VAR has predictive power. Section 9.5 uses both the simulated data and the wood chip/heating oil data to construct an impulse response function. This function traces the dynamic response of one of the VAR variables to a shock to a second variable within the systems. Concluding comments are provided in Section 9.6.

9.1 Conceptual Overview

In this section will we briefly review the VAR literature before examining a VAR at a conceptual level. Most importantly, we will distinguish between a reduced form VAR which is useful for forecasting and a structural VAR which is useful for policy analysis. There are several methods available for identifying the parameters of a structural VAR. In this chapter we use a simple exclusion restriction to achieve identification.

9.1.1 VAR Literature

Economics has a long tradition of modeling dynamic systems. For example, Jorgenson [1963] used a dynamic framework to examine production and investment. Similarly, Daly [1967] modeled the dynamics of demand in a multi-equation framework. Granger [1969] used a bivariate dynamic system to rule

out variable x causing variable y if the past values of x exert no impact on the future path of y. This Granger causality testing framework was the beginning of a reduced form VAR. Sims [1972] used the Granger method an thus an early VAR for assessing the directional impact between money and income.

The 1970s witnessed a large scale shift in macroeconomic modeling. The lack of microeconomic foundations in existing models gave rise to more structurally sound rational expectations (RE) models (e.g., Sargent and Wallace [1973]). In a RE model the explanatory variable becomes the latent expectation of the original exogenous variable, and its closed form solution is essentially a VAR [Qin, 2011]. The recognition that all variables within a VAR are endogenous gave rise to the Lucas critique, which is now a key feature of macroeconomic policy anaysis [Lucas, 1976].

The VAR model as we currently know it was fully worked out by Sims [1980]. His goal was to develop an integrated framework for data description, forecasting, structural inference and policy analysis [Stock and Watson, 2001]. If the goal is purely forecasting then a reduced form VAR is sufficient. In this case each variable is expressed as a linear function of the past values of itself, the other variables in the system and a serially uncorrelated error term. A structural VAR uses theory to impose exogenous restrictions which enable unique identification of the individual coefficients [Bernanke, 1986, Sims, 1986]. A structural VAR is widely used for policy analysis because it allows the system-wide impacts of a shock to a policy variable such as a central bank interest rate to be assessed.

In the food and resource economics literature, Orden and Fackler [1989] used a VAR to examine how agricultural commodity prices are impacted by monetary policy. Haldrup et al. [2010] examined electricity pricing using a VAR with regime switching. Kumar et al. [2012] used a VAR to examine the relationship between the stock price of clean energy firms and prices in oil and carbon markets. Gutierrez et al. [2015] constructed a A global VAR model for the analysis of wheat export prices. Kilian and Zhou [2020] used a VAR to analyze pricing dynamics in the global market for crude oil. Finally, Shahrazi et al. [2022] use a structural VAR to examine the impact of commodity prices on economy-wide inflation.

9.1.2 Structural VAR

A reduced form VAR is fairly intuitive since it consists of regressing each variable on its lagged values and the lagged values of the other variables in the system. A reduced form VAR is well suited for forecasting but it is not useful for policy analysis, which primarily consists of impulse response assessment. This is because the error term in a particular equation of a reduced form VAR is a composite of the individual shocks. Consequently, the impact of a shock to an individual variable cannot be assessed. An equivalent way of stating this problem is that the set of coefficients are over identified in a reduced form VAR.

A structural VAR solves this problem by using economic theory to impose exogenous restrictions on the VAR. A commonly used restriction in a two-equation VAR is to assume that a shock to variable z_t has a contemporaneous impact on variable x_t but not vice versa (i.e., an exclusion restriction). With this assumption the variance-covariance matrix of the estimated VAR can be made orthogonal, which in turn allows the coefficient which measures the contemporaneous impact of z_t on x_t to be uniquely identified. See Dungey and Fry [2009] for a description of other methods of achieving structural VAR identification.

To reduce the complexity of the analysis we will exclusively work with a two variable VAR. Let x_{1t} denote the date t heating oil price premium and x_{2t} denote the date t wood chip price premium. The heating oil price premium is equal to the price of heating oil minus the price of crude oil, and the wood chip price premium is equal to the price of wood chips minus the price of plywood. To simplify the discussion in this conceptual overview section, assume that the units of measure are adjusted such that the long run average value of the price premium is equal to zero in both markets. This assumption is relaxed in the empirical analysis.

A structural VAR with one lag can be written as follows:

$$\begin{aligned} x_{1t} &= \phi_{11}x_{1,t-1} + \phi_{12}x_{2,t-1} + b_{11}\epsilon_{1t} + b_{12}\epsilon_{2t} \\ x_{2t} &= \phi_{21}x_{1,t-1} + \phi_{22}x_{2,t-1} + b_{21}\epsilon_{1t} + b_{22}\epsilon_{2t} \end{aligned} \tag{9.1}$$

We can write this same equation in matrix format as

$$\begin{bmatrix} x_{1t} \\ x_{2t} \end{bmatrix} = \begin{bmatrix} \phi_{11} & \phi_{12} \\ \phi_{21} & \phi_{22} \end{bmatrix} \begin{bmatrix} x_{1,t-1} \\ x_{2,t-1} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \begin{bmatrix} \epsilon_t \\ \epsilon_{2t} \end{bmatrix} \tag{9.2}$$

Equations (9.1) and (9.2) are called structural VARs because there exists a structural interpretation. In this particular case we interpret ϵ_{1t} as a supply shock in the heating oil market and ϵ_{2t} as a supply shock in the wood chip market. Thus, b_{11} measures the same-period impact of a heating oil supply shock on the heating oil price premium, and b_{12} measures the same-period impact of a wood chip supply shock on the heating oil price premium. The interpretation of the b_{22} and b_{21} coefficients are analogous. We are particularly interested in the b_{21} coefficient because it is a measure of how a shock in the heating oil price premium impacts the wood chip price premium. We expect a negative sign for b_{21} because a higher supply of heating oil will reduce the demand for burning wood chips and this will reduce the price of wood chips.

We want the VAR to be stable which is equivalent to assuming that in the absence of shocks to ϵ_{1t} and ϵ_{2t} the values of x_{1t} and x_{2t} will converge to their long run mean values, which is zero. Stability is assured if x_{1t} and x_{2t} are covariance stationary (i.e., have a constant mean, variance and covariance). In practical applications if an ADF test allows us to reject a unit root for each x_{it} then we expect the VAR to be stable.

We can write equation (9.2) with more compact notation as

$$X_t = \Phi X_{t-1} + B\epsilon_t \tag{9.3}$$

where

$$\Phi = \begin{bmatrix} \phi_{11} & \phi_{12} \\ \phi_{21} & \phi_{22} \end{bmatrix} \qquad \text{and} \qquad B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

It is important to assume that the ϵ_{it} are white noise error terms, which means they are serially uncorrelated and independent of each other. The variance/covariance matrix for ϵ_t is given by

$$Var(\epsilon_t) = \Sigma_\epsilon = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

9.1.3 Reduced Form VAR

It is not possible to directly estimate the B matrix in equation (9.1) in order to obtain estimates of the individual impact coefficients such as b_{12}. The best we can do is estimate the reduced form VAR, which is

$$X_t = \Phi X_{t-1} + U_t \tag{9.4}$$

where:

$$U_t = \begin{bmatrix} u_{1t} \\ u_{2t} \end{bmatrix} \qquad \text{and} \qquad \begin{aligned} u_{1t} &= b_{11}\epsilon_{1t} + b_{12}\epsilon_{2t} \\ u_{2t} &= b_{21}\epsilon_{1t} + b_{22}\epsilon_{2t} \end{aligned}$$

Let the estimated variance/covariance matrix for U_t be given by

$$Var(U_t) = \Sigma_U = \begin{bmatrix} \sigma_{11}^2 & \sigma_{12}^2 \\ - & \sigma_{22}^2 \end{bmatrix}$$

The "-" in the bottom left corner of Σ_U indicates that Σ_U is a symmetric matrix.

The estimates within Σ_U do not have an economic interpretation. This is because it is not possible to assess the impact of a supply shock in one market while holding the supply in the other market constant. That is, an increase in u_{12} may be due to an increase in ϵ_{1t} or ϵ_{2t} or both.

9.1.4 Identification

Identification means we would like to recover estimates of the B matrix using our estimated Σ_U matrix. It can be shown that

$$\begin{bmatrix} \sigma_{11}^2 & \sigma_{12}^2 \\ - & \sigma_{22}^2 \end{bmatrix} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{21} \\ b_{12} & b_{22} \end{bmatrix} \tag{9.5}$$

This system of equations can be expanded as

$$\begin{aligned} \sigma_{11}^2 &= b_{11}^2 + b_{12}^2 \\ \sigma_{12}^2 &= b_{11}b_{21} + b_{12}b_{22} \\ \sigma_{12}^2 &= b_{11}b_{21} + b_{12}b_{22} \\ \sigma_{21}^2 &= b_{21}^2 + b_{22}^2 \end{aligned} \tag{9.6}$$

Due to the symmetry of Σ_U the middle two expressions in equation (9.6) are identical. This means we have four unknown b_{ij} values but only three equations. What is needed is an exogenous restriction which will allow the system of equations to be solved. There are various methods used in the literature to achieve identification. A popular method is to assume either $b_{12} = 0$ or $b_{21} = 0$. With this *zero contemporaneous restriction* method the above system can be solved since there will be three unknown b_{ij} values and three equations.

For the case at hand it may be reasonable to assume that $b_{12} = 0$, which means that a supply shock in the wood chip market has no direct impact on the price premium for heating oil. This assumption is likely to be reasonable because the volume of sales in the global wood chip market small relative to the volume of sales in the global heating oil market (about one tenth the size in 2020). With the assumption of $b_{12} = 0$ we can use the estimated Σ_U matrix to solve for b_{21} and thus recover how a supply shock in the heating oil market affects the price premium in the wood chip market in the same time period.

In the literature it is common to impose the "zero" restriction in the top right corner of the B matrix. WIth a k variable VAR the number of restrictions required to achieve exact identification is $K(K-1)/2$. Thus, for a three variable VAR three restrictions are required, which is equivalent to setting all values either above or below the principal diagonal of the B matrix equal to zero. The upper right corner of the B matrix is typically restricted and so it is important to order the variables in a way which are consistent with this restriction.

To conclude this section it is useful to note that the assumptions which are used to create the structural VAR (e.g., $b_{21} = 0$) is not required if the VAR is only to be used for forecasting. This is because the error terms are set to zero when forecasting, in which case the structural VAR and the reduced form VAR are identical. The structural VAR is used for impulse response analysis and variance decomposition. In these cases the reduced form VAR is estimated

first and the assumptions of the structural VAR are then imposed. This is made clear in the R programming where it can be seen that the *SVAR* function uses the estimated *VAR* object as its key input.

9.1.5 Forecasting

This section draws from https://faculty.washington.edu/ezivot/econ584/notes/varModels.pdf.

In this section we derive the VAR forecasting functions for the simple model constructed above (i.e., two variables and one lag). The functions consist of the point forecasts for future time periods (i.e., periods $T+1$,$T+2$, etc.) and the associated prediction intervals. A prediction interval is similar to a confidence interval. It shows the lower and upper range of values for an interval which is expected to contain the actual outcome with a predetermined level of probability.

The actual outcome of the x_1 and x_2 variables in period $T+1$, which is the first forecasted period, can be expressed as

$$\begin{aligned} \tilde{x}_{1,T+1} &= \hat{\phi}_{11} x_{1T} + \hat{\phi}_{12} x_{2T} + \tilde{u}_{1,T+1} \\ \tilde{x}_{2,T+1} &= \hat{\phi}_{21} x_{1T} + \hat{\phi}_{22} x_{2T} + \tilde{u}_{2,T+1} \end{aligned} \tag{9.7}$$

In this equation and the equations to follow a "$\sim$" denotes a random outcome and a "$\widehat{\ }$" denotes an estimated coefficient. If equation (9.7) is incremented by one time period we obtain an expression for the period $T+2$ outcomes of the x_1 and x_2 variables. Within this new equation substitute the two expressions in equation (9.7) for $x_{1,T+1}$ and $x_{2,T+1}$. After collecting terms the desired expressions for $\tilde{x}_{1,T+2}$ and $\tilde{x}_{2,T+2}$ can be expressed with matrix notation as

$$\begin{bmatrix} \tilde{x}_{1,T+2} \\ \tilde{x}_{2,T+2} \end{bmatrix} = \begin{bmatrix} \hat{\phi}_{11} & \hat{\phi}_{12} \\ \hat{\phi}_{21} & \hat{\phi}_{22} \end{bmatrix} \begin{bmatrix} \hat{\phi}_{11} & \hat{\phi}_{12} \\ \hat{\phi}_{21} & \hat{\phi}_{22} \end{bmatrix} \begin{bmatrix} x_{1,T} \\ x_{2,T} \end{bmatrix} + \begin{bmatrix} \hat{\phi}_{11} & \hat{\phi}_{12} \\ \hat{\phi}_{21} & \hat{\phi}_{22} \end{bmatrix} \begin{bmatrix} \tilde{u}_{1,T+1} \\ \tilde{u}_{2,T+1} \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{u}_{1,T+1} \\ \tilde{u}_{2,T+1} \end{bmatrix} \tag{9.8}$$

The following definitions will prove useful:

$$\Phi^0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad \hat{\Phi}^1 = \begin{bmatrix} \hat{\phi}_{11} & \hat{\phi}_{12} \\ \hat{\phi}_{21} & \hat{\phi}_{22} \end{bmatrix}, \quad \hat{\Phi}^2 = \begin{bmatrix} \hat{\phi}_{11} & \hat{\phi}_{12} \\ \hat{\phi}_{21} & \hat{\phi}_{22} \end{bmatrix} \begin{bmatrix} \hat{\phi}_{11} & \hat{\phi}_{12} \\ \hat{\phi}_{21} & \hat{\phi}_{22} \end{bmatrix}, \quad \text{etc.} \tag{9.9}$$

With these definitions, equation (9.8) can be rewritten with compact notation as

$$\tilde{X}_{T+2} = \hat{\Phi}^2 X_T + \sum_{j=0}^{1} \hat{\Phi}^j \tilde{U}_{T+2-j} \tag{9.10}$$

More generally, the actual value of X in period $T+n$ can be expressed as

$$\tilde{X}_{T+n} = \hat{\Phi}^n X_T + \sum_{j=0}^{n-1} \hat{\Phi}^j U_{T+n-j} \tag{9.11}$$

The forecasted value for X in period $T+n$ is the expected value of equation (9.11) conditioned on the information which is available in period T. The expected value of $\tilde{U}_{T+n-j}$ is zero for all j. Thus, the period $T+n$ forecasted value of X is $\hat{\Phi}^n X_T$.

The forecast error for period $T+n$ is the difference between the actual X_{T+n} outcome and the forecasted value for X_{T+n} as of date T. Using equation (9.11) together with expression for the forecasted value, $\hat{\Phi}^n X_T$, the following expression emerges for the forecast error:

$$\tilde{Z}_{T+n} = \sum_{j=0}^{n-1} \hat{\Phi}^j \tilde{U}_{T+n-j} \tag{9.12}$$

To construct a forecast prediction interval we need an expression for the standard deviation of the forecast error. This calculation is simplified by noting that although $\tilde{u}_{1t}$ and $\tilde{u}_{2t}$ are correlated in each time period, there is no cross period correlation. This means we can drop the t subscript and calculate an expression for the standard deviation of the forecast error using the generic expressions $\tilde{u}_1$ and $\tilde{u}_2$. This allows equation (9.12) to be written as

$$\tilde{Z}_{T+n} = \sum_{j=0}^{n-1} \hat{\Phi}^j \tilde{U}$$

.

To obtain an expression for the standard deviation of Z_{T+n}, which is the period n forecast error for x_1 and x_2, the following definitions are required:

$$\Sigma = \begin{bmatrix} var(x_1) & cov(x_1, x_2) \\ cov(x_1, x_2) & var(x_2) \end{bmatrix} \qquad \text{and} \qquad \Psi^n = \sum_{j=0}^{n-1} \hat{\Phi}^j \tag{9.13}$$

Deriving an expression for the standard deviation of Z_{T+n} is straight forward but it is somewhat tedious to write out the intermediate steps. Using the previous expression, the top left and bottom right elements of the following matrix are measures of the variance of the x_1 and x_2 forecast errors, respectively.

$$variance(Z_{T+n}) = \Psi^n \Sigma (\Psi^n)^\top \tag{9.14}$$

If the errors are normally distributed then the upper and lower bounds of the 95 percent prediction interval for the period $T+n$ forecast are approximately given

by the point forecast, $\Phi^n X_T$, plus and minus $1.96 * StdDev(Z_{T+n})$, respectively. Not surprisingly, the standard deviation of the forecast error and thus the width of the prediction interval is increasing with n (i.e., a longer forecasting time horizon). This outcome is evident in equation (9.13) where it can be seen that increasing the forecasting period from $T + n$ to $T + n + 1$ adds an additional Φ^{n+1} to the expression which is multiplying the variance/covariance matrix, Σ. However, the width of the confidence interval increases at a decreasing rate as n increases and will eventually level off because the individual elements of Φ^n are decreasing with higher n. The shape of the prediction interval is examined more closely in the empirical application below.

9.2 Forecasting With Simulated Data

In this section we use a two-variable, one-lag VAR and an artificially created data set to demonstrate the method of VAR forecasting. Doing this keeps the empirical analysis tightly linked to the conceptual material which was presented in the previous section. In the next section the VAR model we will use to evaluate price premiums will use multiple lags, an exogenous variable and a correction for seasonality.

9.2.1 Generating the Data

We will use the following structural VAR to generate the artificial data set:

$$\begin{bmatrix} x_{1t} \\ x_{2t} \end{bmatrix} = \begin{bmatrix} 0.9 & 0.025 \\ 0.2 & 0.8 \end{bmatrix} \begin{bmatrix} x_{1,t-1} \\ x_{2,t-1} \end{bmatrix} + \begin{bmatrix} 0.5 & 0 \\ 0.25 & 0.3 \end{bmatrix} \begin{bmatrix} \epsilon_t \\ \epsilon_{2t} \end{bmatrix} \tag{9.15}$$

Notice that this VAR is identified because the b_{12} element in the top right corner of the B variance/covariance matrix is restricted to zero. This means that shocks to ϵ_{2t} have no same-period impact on x_{1t} whereas shocks to ϵ_{1t} do have a same-period impact on x_{2t}.

We begin in the usual way by loading the required packages.

```
pacman::p_load(tidyverse, here, tsibble, vars, forecast)
```

We want to simulate values for x_{1t} and x_{2t} over 250 time periods, assuming that ϵ_{1t} and ϵ_{2t} are independent standard normal random variables. We begin by assigning the parameters from the Φ coefficient matrix in equation (9.15) to a matrix. After this assignment we should ensure that the VAR is stable by verifying that the eigenvalues of the Φ matrix are less than one in absolute value. The results below confirm this outcome.

```
Phi_sim <- matrix(c(0.9, .025,
                    0.2, 0.8), 2, byrow=TRUE)
eigen(Phi_sim)$values
```

```
## [1] 0.9366025 0.7633975
```

The next stop is to:

- Assign values to the variance/covariance matrix in equation (9.15).
- Assign $x_{1,0} = 8$ and $x_{2,0} = 5$ starting (date 0) values for the two variables.
- Define the $T = 250$ sample size parameter
- Seed the random number generator to ensure the results can be replicated.

```
B <- matrix(c(0.5, 0,
              0.25, 0.3), 2, byrow=TRUE)
initial <- c(8,5)
T <- 250
set.seed(35)
```

We can now build an empty shell matrix to hold the simulated data and then attach to this matrix the initial values for x_1 and x_2.

```
raw <- cbind(initial, matrix(0, 2, T ))
```

A loop procedure together with equation (9.15) is used to fill the shell with the simulated values for x_{1t} and x_{2t}.

```
for (i in 2:(T + 1)){
  raw[, i] <- Phi_sim %*% raw[, i - 1] +  B %*% rnorm(2, 0, 1)
}
```

To prepare this data for time series analysis a month index column is added and the resulting data set is converted into a tsibble object called X.

```
X <- data.frame(t(raw)) %>% mutate(month = 1:nrow(t(raw)))
rownames(X) <- NULL
X <- as_tsibble(X,index=month)
```

A plot of the simulated values for x_{1t} and x_{2t} is presented in Figure 2.1 below. The starting values of this pair of variables are well above their long run equilibrium values, which is zero. Consequently, over the first sixty periods the variables stochastically decline toward toward zero. For the remaining periods x_{1t} and x_{2t} stochastically fluctuate around zero. Both variables have a similar level of volatility and track each other fairly closely.

```
ggplot() +
  geom_line(data = X, aes(x = month, y = X1), color = "blue") +
  geom_line(data = X, aes(x = month, y = X2), color = "red") +
  labs(y= "x1 (blue) and x2 (red)", x = "Month")
```

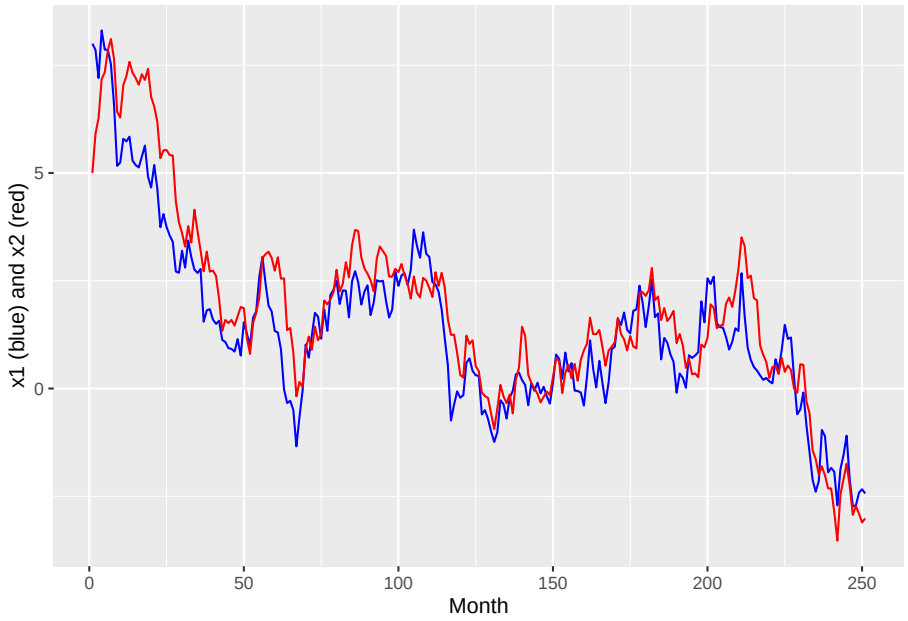

Figure 9.1: Simulated Values for x1 and x2

9.2.2 Estimating the VAR

Now that we have the data, which we know to be stationary and consistent with a one-lag VAR, we can use R's *vars* package to estimate the parameters of the reduced form VAR. Recall that a reduced form VAR is sufficient for forecasting. Later in this chapter we will estimate the parameters of the structural VAR and use the estimated model for conducting impulse response analysis.

The *VAR* function requires a time series object to evaluate (in our case the first two columns of the X matrix), the number of lags to include (in our case $p = 1$) and the options for including an intercept and/or deterministic time trend (in our case *type* = "none"). The full set of results can be generated using the *summary()* function. To reduce clutter only the coefficients and statistical testing variables are displayed.

```
var_est <- VAR(X[,1:2], p = 1, type = "none")
coef(var_est)
```

```
## $X1
##            Estimate Std. Error     t value     Pr(>|t|)
## X1.l1  0.9610077625 0.04124789 23.29834882 2.167960e-64
## X2.l1 -0.0004258584 0.03534349 -0.01204913 9.903961e-01
##
## $X2
##        Estimate Std. Error   t value     Pr(>|t|)
## X1.l1 0.1966216 0.03459673  5.683242 3.697713e-08
## X2.l1 0.8241393 0.02964441 27.800837 3.655554e-78
```

As expected, the estimated coefficients of the reduced form VAR which are shown above are quite close to the parameter values which we assigned to the Φ matrix when constructing this data set. We can also generate the estimated variance/covariance matrix of the residuals of the reduced form VAR (i.e., Σ). The Σ matrix is important because it is a key determinant of the prediction interval which we will generate when forecasting.

```
summary(var_est)$covres
```

```
##           X1        X2
## X1 0.2407767 0.1309224
## X2 0.1309224 0.1693616
```

9.2.3 Point Forecast

Using the forecasting function we derived in Section 9.2, $\Phi^n X_T$, we can forecast values n periods beyond date $T = 250$. This involves creating a matrix which contains the estimated coefficients, creating a vector which contains the period T values from the simulated data and then using a chain procedure to construct $\Phi^n X_T$. The following shows the forecasted values for x_1 and x_2 for periods $T+1$ through $T + 4$.

```
var_coef <- Bcoef(var_est)
X_T <- as.vector(unlist(c(X[nrow(X),1],X[nrow(X),2])))
Xf1 <- var_coef %*% X_T
Xf2 <- var_coef %*% Xf1
Xf3 <- var_coef %*% Xf2
Xf4 <- var_coef %*% Xf3
Xf <- t(cbind(Xf1,Xf2,Xf3,Xf4))
Xf
```

```
##             X1        X2
## [1,] -2.337844 -2.962272
## [2,] -2.245425 -2.900996
## [3,] -2.156636 -2.832324
## [4,] -2.071337 -2.758270
```

Of course it is easier to use the *predict* function within the *vars* package to generate these forecast values. This function requires as input the estimated VAR model, which is *var_est*, the number of periods to forecast (n.ahead = 4) and the level of confidence for the prediction interval (ci = 0.95). The first column of the output of the *predict* function is the set of forecasted values.

```
N <- 4
T_begin <- nrow(X)+1
T_end <- nrow(X)+N

X_for <- predict(var_est, n.ahead = N, ci = 0.95)

rbind(X_for$fcst$X1[,1:1],X_for$fcst$X2[,1:1])
```

```
##           [,1]      [,2]      [,3]      [,4]
## [1,] -2.337844 -2.245425 -2.156636 -2.071337
## [2,] -2.962272 -2.900996 -2.832324 -2.758270
```

The *plot* function in the *vars* package generates a plot of the predicted values along with the corresponding lower and upper limits of the prediction interval. Let's use this function to plot the forecasted values for 20 periods beyond date T.

```
N <- 20
T_end <- nrow(X)+N
X_for <- predict(var_est, n.ahead = N, ci = 0.95, dumvar = NULL)
plot(X_for)
```

The plots in Figure 9.2 do not provide a close up look at the forecasted values because the relatively long historical data dominates the plot. Unfortunately the *plot* function does not have the option to truncate the earlier portion of the historical data for the purpose of plotting. To address this shortcoming the desired plot is constructed manually.

The first step is to save in a matrix the forcasted values and prediction intervals.

```
X1_for <- X_for$fcst$X1[,1:3]
X2_for <- X_for$fcst$X2[,1:3]
```

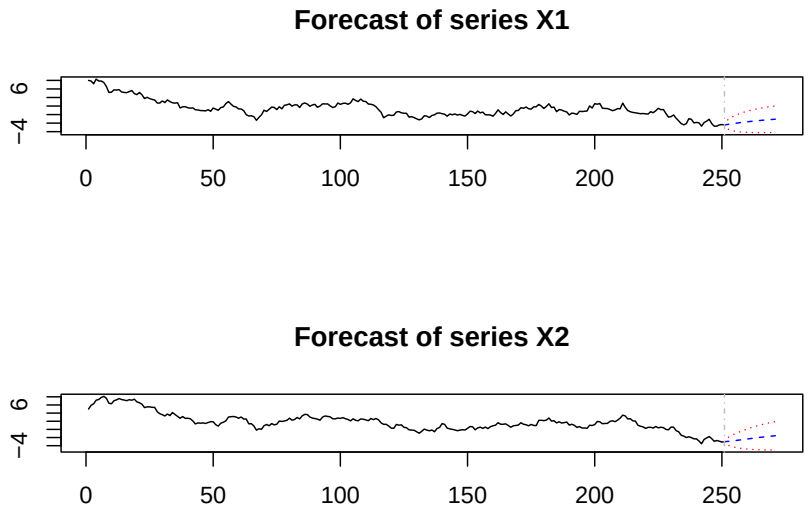

Figure 9.2: Simulated Values for x1 and x2

Now create a tsibble object named *X_plot* which is the same as X except it has additional rows to contain the forecasted values, and it has new columns to mesh with those in the matrix of forecasted values. The forecasted values are then incorporated into bottom set of rows.

```
X_plot <- X %>% add_row(month = c(T_begin:T_end)) %>%
  mutate(x1fcst = NA, x2fcst = NA, x1low = NA, x1up = NA, x2low = NA, x2up = NA)
X_plot$x1fcst[T_begin:T_end] <- X1_for[,1]
X_plot$x2fcst[T_begin:T_end] <- X2_for[,1]
X_plot$x1low[T_begin:T_end] <- X1_for[,2]
X_plot$x2low[T_begin:T_end] <- X2_for[,2]
X_plot$x1up[T_begin:T_end] <- X1_for[,3]
X_plot$x2up[T_begin:T_end] <- X2_for[,3]
```

For the purpose of plotting it is useful to create separate tsibble objects for the x_1 and x_2 variables. These objects contain the last 30 periods of the historic data and the 20 periods of the forecasted data for a total of 50 periods.

```
x1_plot <- X_plot %>%
  filter(row_number() >= (n() - 50)) %>%
  dplyr::select(month,X1,x1fcst,x1low,x1up)
```

```
x2_plot <- X_plot %>%
  filter(row_number() >= (n() - 50)) %>%
  dplyr::select(month,X2,x2fcst,x2low,x2up)
```

Plots of the x_1 and x_2 historic and forecasted values are shown in Figures 9.3 and 9.4, respectively. In each case we know from the AR1 feature of the model that the percent increase in the forecasted value toward the long run equilibrium value of zero is constant over time. This means that the forecasting schedule will asymptotically approach zero if the forecasting horizon becomes very long. If the model contained multiple lags rather than just one lag then the first few periods of the forecast would reflect the variation in the last few periods of the historic data.

```
ggplot() +
  geom_line(data = x1_plot, aes(x = month, y = X1), color = "black") +
  geom_line(data = x1_plot, aes(x = month, y = x1fcst), color = "green",linewidth=1.5) +
  geom_line(data = x1_plot, aes(x = month, y = x1low), color = "red",linewidth=1.5) +
  geom_line(data = x1_plot, aes(x = month, y = x1up), color = "blue",linewidth=1.5)
```

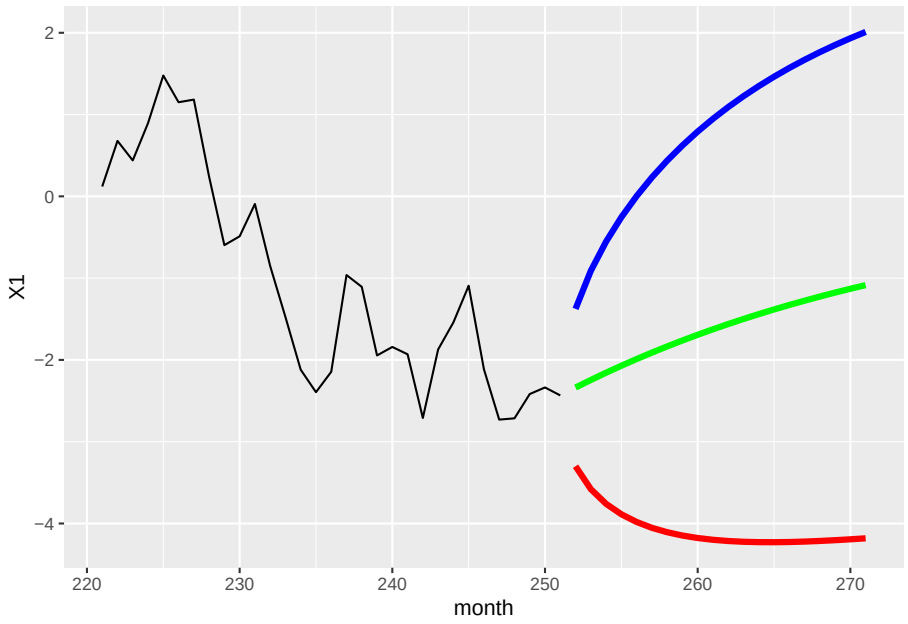

Figure 9.3: Forecast of x1 Variable

and

```
ggplot() +
  geom_line(data = x2_plot, aes(x = month, y = X2), color = "black") +
  geom_line(data = x2_plot, aes(x = month, y = x2fcst), color = "green",linewidth=1.5)
  geom_line(data = x2_plot, aes(x = month, y = x2low), color = "red",linewidth=1.5) +
  geom_line(data = x2_plot, aes(x = month, y = x2up), color = "blue",linewidth=1.5)
```

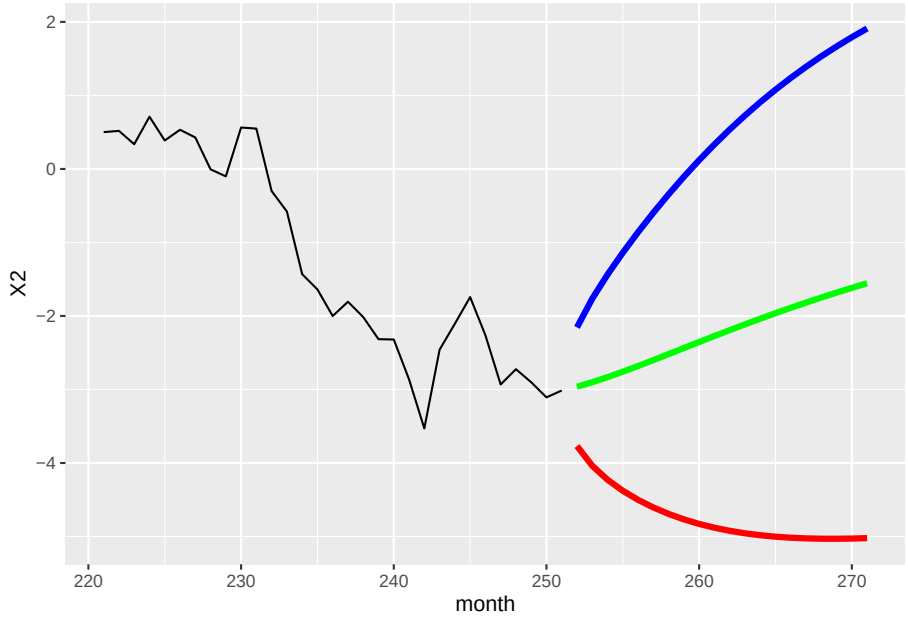

Figure 9.4: Forecast of x2 Variable

9.2.4 Prediction Intervals

The upper and lower coloured schedules in Figures 9.3 and 9.4 define the 95 percent prediction intervals. The width of the interval period 251, which is the beginning of the forecasting period, reflects the overall volatility of the error terms as measured by the elements of the Σ variance/covariance matrix. Figures 9.3 and 9.4 show that the width of this initial interval is about 2. This relatively wide interval implies a relatively low level of forecasting precision.

The vertical distance between the point forecast line and either the upper or lower prediction interval line is approximately equal to $1.96 * StdDev(Z_{t+n})$ where $StdDev(Z_{t+n})$ is the standard deviation of the period $T + n$ forecast error. The faster that this standard deviation rises with a longer forecasting horizon (i.e., a higher n) the steeper the upper and lower prediction interval schedules in Figures 9.3 and 9.4. Understanding the determinants of this slope

is important because the slope indicates how rapidly the forecast precision falls as the forecasting time horizon extends.

Recall from Section 9.1 that the variance of the period $T+n$ forecast error can be expressed as

$$variance(Z_{T+n}) = \Psi^n \Sigma (\Psi^n)^\top \qquad \text{where} \qquad \Psi^n = \sum_{j=0}^{n-1} \hat{\Phi}^j \tag{9.16}$$

Also recall that $\hat{\Phi}^j$ is the estimated coefficient matrix multiplied by itself j times. If x_{1t} and x_{1t} are highly autoregressive (i.e., close to evolving over time as correlated random walks) then the diagonal elements of $\hat{\Phi}$ will be close to 1, and $\hat{\Phi}^j$ will decline relatively slowly with higher j. In this case $StdDev(Z_{t+n})$ will rise relatively rapidly with higher n and the prediction interval lines will be relatively steep and linear. This makes sense because there is weaker attraction to the long run mean for highly autoregressive stochastic processes and so the ability to accurately forecast is lower. In contrast, if x_{1t} and x_{1t} have only a weak connection to past values then the diagonal elements of the $\hat{\Phi}^j$ matrix will be close to zero. In this case $\hat{\Phi}^j$ will decline relatively rapidly with higher j. The rapid decline implies that $StdDev(Z_{t+n})$ will rise relatively slowly with higher n, which in turn implies that the prediction interval lines will be relatively flat. The stronger attraction of x_{1t} and x_{1t} to their long run mean values imply that the ability to accurately forecast is relatively higher as n increases.

9.3 Forecasting Price Premiums

In this section we will forecast price premiums for wood chips and heating oil. The monthly wood chip price premium is the log of the difference between the average monthly wood chip and plywood price indexes. The monthly heating oil price premium is the log of the difference between the average monthly price of heating fuel and the price of crude oil, both measured in dollars per barrel. Logging the price difference implies that the price premiums can be approximately interpreted as a percentage difference rather than a straight difference. The data runs from January of 1986 to May of 2022.

Wood chips are used as a source of fuel and thus can be considered a substitute for heating oil. For this reason the pair of price premiums should be viewed as endogenous where current and past supply shocks in the heating oil market are expected to affect the current price premium in the wood chip market. This makes VAR modeling an appropriate method for forecasting future price premium outcomes for these two variables.

Forecasting does not require a structural VAR and so it is sufficient to estimate a reduced form VAR. An important difference between the analysis using real-world data in this section and the analysis using artificial data in the previous

section is that this current real-world analysis requires more than one lag and includes both the use of seasonal dummies to control for seasonality and a set of exogenous variables to control for shocks in demand for wood chips and heating oil.

In the Appendix of this chapter we use ADF testing to verify that the pair of price premium variables are stationary. It is also shown that the exogenous temperature data is also stationary. The outcome that all time series variables which are included in the model are stationary implies that the estimated VAR is stable and the forecasted values will asymptotically approach the long run equilibrium values of the two endogenous variables as the forecasting period becomes very long.

The formal analysis begins by reading in the pre-cleaned price premium data from a saved .RDS file. In addition to the pair of endogenous price premium variables for wood chips and heating fuel, the data includes exogenous monthly average temperature expressed as a deviation from the long term monthly average temperature (i.e., the "temperature anomaly"). Two lags of the temperture are also included. The "nas" which are associated with the lagged temperatures and which reside in the top two rows are deleted after the data has been read in.

```
wood <- readRDS(here("data/ch8","wood.RDS")) %>%
  slice(-c(1:2))
head(wood)
```

```
## # A tsibble: 6 x 7 [1D]
##   month      premE premW period  temp L.temp L2.temp
##   <date>     <dbl> <dbl>  <dbl> <dbl>  <dbl>   <dbl>
## 1 1986-03-01  3.72  2.69      3  2.61   0.2     2.33
## 2 1986-04-01  3.63  2.97      4  1.01   2.61    0.2
## 3 1986-05-01  3.52  2.87      5  0.03   1.01    2.61
## 4 1986-06-01  3.48  2.77      6  0.91   0.03    1.01
## 5 1986-07-01  3.22  2.77      7 -0.9    0.91    0.03
## 6 1986-08-01  3.32  2.75      8 -1.35  -0.9     0.91
```

Plots of the two endogenous price premium variables are as follows.

```
ggplot() +
  geom_line(data = wood, aes(x = month, y = premW), color = "blue") +
  geom_line(data = wood, aes(x = month, y = premE), color = "red") +
  labs(y= "Price Premiums", x = "Date") +
  ggtitle("Heating Oil (red) and Woodships (blue)")
```

Figure 9.5 reveals a modest upward slope in the pair of price premiums. A linear time trend will be included in the VAR model to control for this trend.

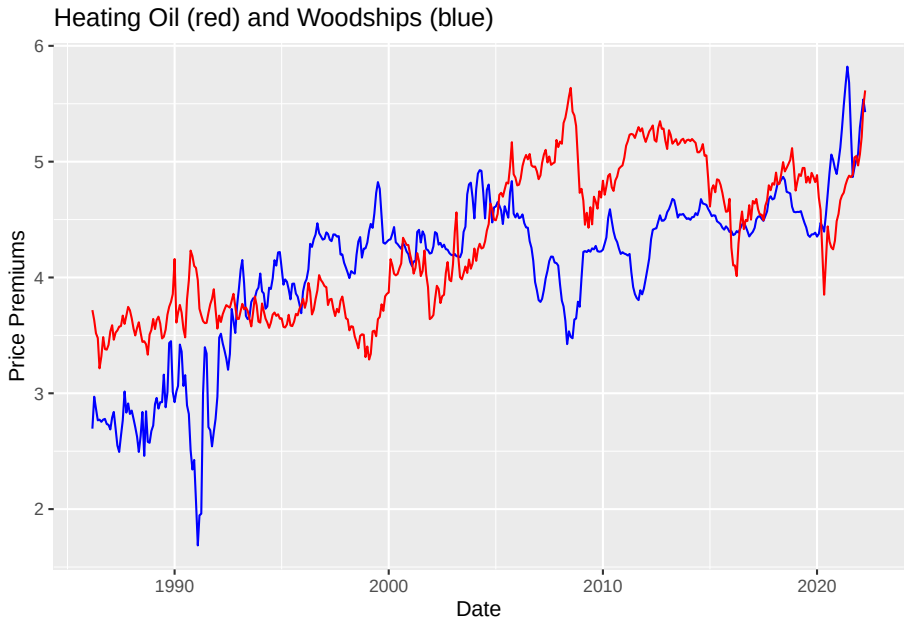

Figure 9.5: Price Premiums for Wood Chips and Heating Oil

A graph of the stationary temperature anomaly series is as follows.

```
ggplot() +
  geom_line(data = wood, aes(x = month, y = temp), color = "blue") +
  ggtitle("Average Monthly Temperature Anonamlies")
```

The price premium may have seasonality because the demand for wood chips as a fuel and heating oil is seasonal. One option is to use monthly seasonal dummies and the second option is to use quarterly seasonal dummies. There is a trade-off between having a monthly dummy model with a somewhat better fit and a quarterly dummy model with lower risk of over-fitting. It is useful to observe the differences in the forecasting procedure in each case and so both model specifications will be maintained.

Because the data is monthly there is no convenient option for including quarterly dummies in the VAR estimation procedure. Rather, the the quarterly dummies must be created manually and then combined with the other exogenous variables in the VAR. The following code creates the three dummy variables (the dummy for quarter 4 is omitted).

```
wood <- wood %>%
  mutate(qtr1 = ifelse(period == 1| period == 2| period == 3, 1,0),
```

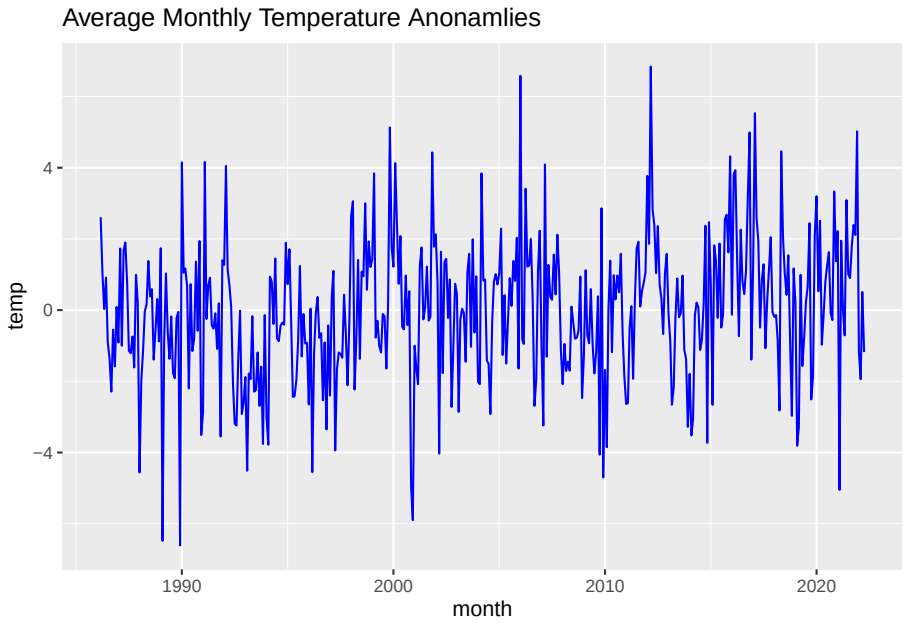

Figure 9.6: Average Monthly Temperature Anamoly

```
        qtr2 = ifelse(period == 4| period == 5| period == 6, 1,0),
        qtr3 = ifelse(period == 7| period == 8| period == 9, 1,0)
        )
head(wood)
```

```
## # A tsibble: 6 x 10 [1D]
##   month      premE premW period  temp L.temp L2.temp  qtr1  qtr2  qtr3
##   <date>     <dbl> <dbl>  <dbl> <dbl>  <dbl>   <dbl> <dbl> <dbl> <dbl>
## 1 1986-03-01  3.72  2.69      3  2.61   0.2     2.33     1     0     0
## 2 1986-04-01  3.63  2.97      4  1.01   2.61    0.2      0     1     0
## 3 1986-05-01  3.52  2.87      5  0.03   1.01    2.61     0     1     0
## 4 1986-06-01  3.48  2.77      6  0.91   0.03    1.01     0     1     0
## 5 1986-07-01  3.22  2.77      7 -0.9    0.91    0.03     0     0     1
## 6 1986-08-01  3.32  2.75      8 -1.35  -0.9     0.91     0     0     1
```

These three quarterly dummies must now be added to the temperature data and the combined data saved as a matrix. This is because quarterly dummies and temperature data are exogenous variables and as such they must enter the VAR estimation within a single matrix which contains all of the exogenous data. We also need a separate exogenous matrix which only contains the temperature data. This specification will be used when monthly seasonal dummies are in

place. In both cases the exogenous variables are created by extracting the corresponding columns from the *wood* data frame.

```
weather <- as.matrix(wood[,5:7])
head(weather)
```

```
##       temp L.temp L2.temp
## [1,]  2.61   0.20    2.33
## [2,]  1.01   2.61    0.20
## [3,]  0.03   1.01    2.61
## [4,]  0.91   0.03    1.01
## [5,] -0.90   0.91    0.03
## [6,] -1.35  -0.90    0.91
```

```
weather_season <- as.matrix(wood[,5:10])
head(weather_season)
```

```
##       temp L.temp L2.temp qtr1 qtr2 qtr3
## [1,]  2.61   0.20    2.33    1    0    0
## [2,]  1.01   2.61    0.20    0    1    0
## [3,]  0.03   1.01    2.61    0    1    0
## [4,]  0.91   0.03    1.01    0    1    0
## [5,] -0.90   0.91    0.03    0    0    1
## [6,] -1.35  -0.90    0.91    0    0    1
```

We can now estimate the VAR, first with monthly seasonal dummies and then with quarterly dummies. In the first case the *season = 12* option is used within the *var()* function. In the second case the *exog = weather_season* option is used within the *var()* function. Before proceeding with this estimation we must use the *VARselect()* function to determine the optimal number of lags to include in the VAR. This selection procedure is used first for the case of monthly seasonal dummies and then for the case of quarterly seasonal dummies. The "both" option is used to ensure that both an intercept and linear time trend are included with the set of exogenous VAR variables.

```
VARselect(wood[,2:3],lag.max=10,type="both", exog=weather, season=12)
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      2      2      1      2
##
## $criteria
##                  1            2            3            4            5
```

```
## AIC(n) -8.4788401418 -8.5131984495 -8.5116333210 -8.506599538 -8.5070507595
## HQ(n)  -8.3429886715 -8.3622523714 -8.3455926350 -8.325464244 -8.3108208579
## SC(n)  -8.1349948484 -8.1311481235 -8.0913779624 -8.048139147 -8.0103853357
## FPE(n)  0.0002078408  0.0002008287  0.0002011526  0.000202179  0.0002021011
##                    6             7             8             9            10
## AIC(n) -8.5020213205 -8.4887860676 -8.4844648144 -8.4748816193 -8.4611775041
## HQ(n)  -8.2906968111 -8.2623669504 -8.2429510894 -8.2182732865 -8.1894745635
## SC(n)  -7.9671508641 -7.9157105787 -7.8731842929 -7.8253960652 -7.7734869174
## FPE(n)  0.0002031357  0.0002058604  0.0002067729  0.0002087879  0.0002116963
```

```
VARselect(wood[,2:3],lag.max=10,type="both",exog=weather_season)
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      2      2      1      2
##
## $criteria
##                    1             2             3             4             5
## AIC(n) -8.4991945856 -8.5304548034 -8.5264831284 -8.5240286022 -8.5238428173
## HQ(n)  -8.4237215465 -8.4398871565 -8.4208208737 -8.4032717397 -8.3879913470
## SC(n)  -8.3081694226 -8.3012246078 -8.2590479002 -8.2183883414 -8.1799975239
## FPE(n)  0.0002036359  0.0001973712  0.0001981601  0.0001986518  0.0001986948
##                    6             7             8             9            10
## AIC(n) -8.5199070986 -8.5072269006 -8.5013282472 -8.4941346490 -8.4799700579
## HQ(n)  -8.3689610204 -8.3411862146 -8.3201929535 -8.2979047474 -8.2686455485
## SC(n)  -8.1378567726 -8.0869715420 -8.0428678561 -7.9974692252 -7.9450996015
## FPE(n)  0.0001994859  0.0002020409  0.0002032475  0.0002047284  0.0002076649
```

We typically use either the *AIC* or *BIC* as the information criteria for selecting the optimal number of lags. Note that *BIC* is the same as the Schwarz Criteria (SC). In the above results we see that for the monthly seasonal dummies (first case) two lags is optimal according to the *AIC* and one lag is optimal according to the *BIC*. For the quarterly dummies (second case) four lags is optimal according to the *AIC* and one lag is optimal according to the *BIC*. Which information criteria should be used?

In section 12.3 of the "Forecasting: Principles and Practice (3rd ed)" by Rob J Hyndman and George Athanasopoulos we are told: "Care should be taken when using the AICc as it tends to choose large numbers of lags; instead, for VAR models, we often use the BIC instead. Following this advice, one lag will be used for the VAR estimation of the wood chip - heating oil market.

We can now estimate the VAR first with monthly seasonal dummies and then with quarterly seasonal dummies. In the first case, the exogenous matrix is set to *weather* and the *season=12* is used as an option. In the second case the

temperature and quarterly dummies are combined and so only the exogenous matrix is set to *weather_season* and the *season* option is not used.

```
var12 <- VAR(wood[,2:3], p = 1, type = "both", exog=weather, season = 12)
coef(var12)
```

```
## $premE
##               Estimate   Std. Error    t value      Pr(>|t|)
## premE.l1  0.9625150999 0.0149119598 64.5465193 1.599225e-218
## premW.l1  0.0163123164 0.0124564284  1.3095501  1.910726e-01
## const     0.0740002898 0.0735578444  1.0060149  3.149946e-01
## trend     0.0001163531 0.0001027576  1.1323065  2.581594e-01
## sd1       0.0052620870 0.0252098314  0.2087315  8.347602e-01
## sd2       0.0050152780 0.0250325878  0.2003500  8.413050e-01
## sd3      -0.0043359380 0.0251897589 -0.1721310  8.634185e-01
## sd4      -0.0202469534 0.0252197220 -0.8028222  4.225370e-01
## sd5      -0.0106593780 0.0251830012 -0.4232767  6.723126e-01
## sd6       0.0266174505 0.0251965311  1.0563935  2.914031e-01
## sd7       0.0438324888 0.0251755136  1.7410763  8.241129e-02
## sd8       0.0239380121 0.0252112998  0.9494954  3.429213e-01
## sd9      -0.0026552538 0.0252183077 -0.1052907  9.161960e-01
## sd10     -0.0530232159 0.0251790602 -2.1058457  3.581790e-02
## sd11     -0.0042013072 0.0251917404 -0.1667732  8.676297e-01
## temp     -0.0010005584 0.0027247749 -0.3672077  7.136512e-01
## L.temp   -0.0083318749 0.0027276099 -3.0546431  2.398562e-03
## L2.temp   0.0056026068 0.0027250311  2.0559790  4.040950e-02
##
## $premW
##               Estimate   Std. Error    t value      Pr(>|t|)
## premE.l1 -0.0484950920 0.0180688562 -2.6839049  7.567982e-03
## premW.l1  0.9435197367 0.0150934831 62.5117296 2.753707e-213
## const     0.3529792516 0.0891302107  3.9602650  8.809456e-05
## trend     0.0004256805 0.0001245116  3.4188026  6.911876e-04
## sd1       0.0080639995 0.0305468111  0.2639883  7.919200e-01
## sd2      -0.0213822615 0.0303320447 -0.7049397  4.812432e-01
## sd3      -0.0229396498 0.0305224892 -0.7515655  4.527384e-01
## sd4      -0.0226051984 0.0305587956 -0.7397281  4.598833e-01
## sd5      -0.0179541984 0.0305143010 -0.5883864  5.565931e-01
## sd6      -0.0624485700 0.0305306952 -2.0454356  4.144199e-02
## sd7       0.0056223752 0.0305052282  0.1843086  8.538614e-01
## sd8      -0.0714666506 0.0305485905 -2.3394418  1.978604e-02
## sd9      -0.0539522119 0.0305570819 -1.7656206  7.819482e-02
## sd10     -0.0444849299 0.0305095256 -1.4580669  1.455784e-01
## sd11     -0.0260464953 0.0305248903 -0.8532871  3.939923e-01
## temp      0.0018265077 0.0033016160  0.5532163  5.804132e-01
```

```
## L.temp    0.0026773889 0.0033050512  0.8100900  4.183527e-01
## L2.temp  -0.0025362883 0.0033019266 -0.7681238  4.428505e-01
```

```
var4 <- VAR(wood[,2:3], p = 1, type = "both", exogen = weather_season)
coef(var4)
```

```
## $premE
##               Estimate   Std. Error    t value      Pr(>|t|)
## premE.l1  0.9627172798 0.0150456636 63.9863621 1.187888e-219
## premW.l1  0.0173985957 0.0125596543  1.3852766  1.666979e-01
## const     0.0582649427 0.0753074407  0.7736944  4.395437e-01
## trend     0.0001119621 0.0001036629  1.0800598  2.807310e-01
## temp     -0.0008199895 0.0027437425 -0.2988580  7.651951e-01
## L.temp   -0.0086462169 0.0027460681 -3.1485807  1.756906e-03
## L2.temp   0.0054655684 0.0027431374  1.9924515  4.696377e-02
## qtr1      0.0109517204 0.0147017925  0.7449242  4.567312e-01
## qtr2      0.0041672227 0.0146392256  0.2846614  7.760429e-01
## qtr3      0.0305282137 0.0146931262  2.0777208  3.833774e-02
##
## $premW
##               Estimate   Std. Error    t value      Pr(>|t|)
## premE.l1 -0.0481054943 0.0180520527 -2.6648213  7.997581e-03
## premW.l1  0.9429156202 0.0150692949 62.5719802 6.159964e-216
## const     0.3240918035 0.0903551961  3.5868640  3.737853e-04
## trend     0.0004278204 0.0001243766  3.4397187  6.401427e-04
## temp      0.0017734239 0.0032919906  0.5387087  5.903713e-01
## L.temp    0.0027372732 0.0032947809  0.8307906  4.065606e-01
## L2.temp  -0.0027484918 0.0032912647 -0.8350868  4.041402e-01
## qtr1      0.0506585054 0.0176394701  2.8718836  4.285498e-03
## qtr2      0.0343739677 0.0175644012  1.9570247  5.100191e-02
## qtr3      0.0317702763 0.0176290721  1.8021525  7.223319e-02
```

The above regression results for the quarterly seasonality model can be summarized as follows. For the heating oil price premium equation (i.e., the first one) in both the 12 month and quarterly seasonality models the lag of the heating oil price premium is significant but the lag of the wood chip price premium is not. As well, the first two lags of the temperature anomaly variable is significant but the temperature variable itself is not. The time trend variable is not significant and only a small number of the seasonality dummies are significant. In contrast, for the wood chip price premium equation, the lags of both endogenous variables are significant as is the time trend. However, none of the temperature variables are significant and once again only a small number of the seasonality variables are significant. The most important result is that wood chip price premium depends on the lag of the heating oil price premium but not vice versa.

To generate a forecast we need a matrix which contains the forecasts of the exogenous variables. When working with the monthly seasonal dummies we need forecasts of temperature only because the forecasting procedure will automatically include the forecasts of the monthly seasonal dummies. When working with the quarterly seasonal dummies we need forecasts of both the temperature and the quarterly dummies. Let's forecast for five months ahead and assume a value of zero for each forecasted temperature anomaly. Forecasting the quarterly dummies involves continuing the sequence of seasonal dummies for the futures months.

The last month in the data is April of 2022, which means that May is the first month of the forecast. With this information in hand, the pair of exogenous forecasts can be created as follows:

```
temp <- c(0,0,0,0,0)
L.temp <- c(0,0,0,0,0)
L2.temp <- c(0,0,0,0,0)
weather_for <- as.matrix(cbind(temp,L.temp,L2.temp))
```

```
temp <- c(0,0,0,0,0)
L.temp <- c(0,0,0,0,0)
L2.temp <- c(0,0,0,0,0)
qtr1 <- c(0,0,0,0,0)
qtr2 <- c(1,1,0,0,0)
qtr3 <- c(0,0,1,1,1)
weather_season_for <- as.matrix(cbind(temp,L.temp,L2.temp,qtr1,qtr2,qtr3))
```

The forecasts and associated prediction intervals with the 12 month seasonal dummy model can be generated as follows:

```
for12 <- predict(var12, n.ahead = 5, ci = 0.95, dumvar = weather_for)
for12$fcst$premW[,1:3]
```

```
##          fcst    lower    upper
## [1,] 5.391405 5.137996 5.644814
## [2,] 5.358152 5.009527 5.706777
## [3,] 5.332764 4.917059 5.748468
## [4,] 5.265171 4.797448 5.732895
## [5,] 5.268515 4.758566 5.778464
```

```
for12$fcst$premE[,1:3]
```

```
##          fcst    lower    upper
## [1,] 5.612565 5.403430 5.821699
```

```
## [2,] 5.593812 5.303544 5.884081
## [3,] 5.584925 5.235984 5.933865
## [4,] 5.613349 5.217823 6.008875
## [5,] 5.656937 5.222790 6.091084
```

The forecasts and associated prediction intervals with the quarterly dummy model can be generated as follows.

```
for4 <- predict(var4, n.ahead = 5, ci = 0.95, dumvar = weather_season_for)
for4$fcst$premW[,1:3]
```

```
##          fcst    lower    upper
## [1,] 5.392205 5.139000 5.645409
## [2,] 5.359458 5.011196 5.707719
## [3,] 5.326603 4.911431 5.741776
## [4,] 5.294998 4.827985 5.762010
## [5,] 5.264631 4.755579 5.773683
```

```
for4$fcst$premE[,1:3]
```

```
##          fcst    lower    upper
## [1,] 5.611249 5.400213 5.822285
## [2,] 5.607111 5.314182 5.900039
## [3,] 5.629030 5.276867 5.981193
## [4,] 5.649672 5.250466 6.048879
## [5,] 5.669107 5.230892 6.107322
```

Note that when setting the options for the *predict()* function (see above) we used *dumvar=weather_for* or *dumvar = weather_season_for* Use of the placeholder *dumvar* rather than *exog* (which is the option in the *var()* function) is rather unusual because the exogenous variables are typically a mix of continuous variables such as weather and dummy variables such as those which measure seasonality.

9.4 Granger Causality

In our analysis of heating oil and wood chips it was noted that the volume of sales in the global wood chip market is small relative to the volume of sales in the global heating oil market (about one tenth the size in 2020). For this reason it is reasonable to assume that within a given period, a shock to the price premium in the heating oil market will affect the price premium in the wood chip market but not vice versa. As was noted in Section 9.1.2, making this assumption allows the parameters of the structural VAR to be identified.

Is there a way that we can test whether heating oil is the more important of the two markets? A common approach is to test for *Granger causality*, which is equivalent to testing the information content of heating oil relative to that of wood chips. Specifically, we believe that current and past values of the price premium for heating oil are statistically significant predictors of future values of the price premium for wood chips. That is, we believe the heating oil price premium Granger causes (i.e., predicts) the wood chips price premium. In contrast, we believe that the wood chip price premium does not help to predict future values of the heating oil price premium. It is important to keep in mind that Granger causality is very different than the type of causality which we have in mind when using instrumental variables and other variables to eliminate omitted variable bias and other forms of endogeneity bias in a regular regression equation. Indeed, even if we find that price premiums in the heating oil market Granger cause price premiums in the wood chip market, the estimated relationship is unlikely to be truly causal because there are no controls for important omitted variables and other aspects of endogeneity.

Testing for Granger causality is straight forward after a reduced form VAR has been estimated. The reduced form VAR for wood chips and heating oil can be written with extended notation as follows:

$$\begin{aligned} x_{1t} = & \phi_{11}^1 x_{1,t-1} + \phi_{11}^2 x_{1,t-2} + \phi_{11}^3 x_{1,t-3} \\ & + \phi_{12}^1 x_{2,t-1} + \phi_{12}^2 x_{2,t-2} + \phi_{12}^3 x_{2,t-3} \\ & + \beta_{10} + \beta_{11} D_1 + \beta_{12} D_2 + \beta_{13} D_3 + \beta_{14} t + \gamma_{11} T_t + \gamma_{12} T_{t-1} + \gamma_{13} T_{t-3} + u_{1t} \end{aligned} \tag{9.17}$$

and

$$\begin{aligned} x_{2t} = & \phi_{21}^1 x_{1,t-1} + \phi_{21}^2 x_{1,t-2} + \phi_{21}^3 x_{1,t-3} \\ & + \phi_{22}^1 x_{2,t-1} + \phi_{22}^2 x_{2,t-2} + \phi_{22}^3 x_{2,t-3} \\ & + \beta_{20} + \beta_{21} D_1 + \beta_{22} D_2 + \beta_{23} D_3 + \beta_{24} t + \gamma_{21} T_t + \gamma_{22} T_{t-1} + \gamma_{23} T_{t-3} + u_{2t} \end{aligned} \tag{9.18}$$

Within this pair of equations, x_1 represents the price premium for heating oil, x_2 represents the price premium for wood chips, D_1 through D_3 are quarterly dummies, T is the temperature anomaly and t is the time trend.

The null hypothesis that the heating fuel price premium does not Granger cause the wood chip price premium, which we expect to reject, requires testing the following restriction in equation (9.18): $\phi_{21}^1 = \phi_{21}^2 = \phi_{21}^3 = 0$. The opposite null, which we expect not to reject, requires testing $\phi_{12}^1 = \phi_{12}^2 = \phi_{12}^3 = 0$ in equation (9.17). We can test this pair of restrictions both manually and using the *causality* function in the *vars* package. We will use the quarterly seasonality model in both cases. However, To make the Granger test more relevant we will

estimate the VAR with three lags rather than just one lag (a Granger test with one lag is a simple t test for the significance of the lagged variable).

There are two approaches to manually conducting the F test. The first manual method is described in most undergraduate statistics textbooks. In this standard F test the residual sum of squares (RSS) for the unrestricted and restricted estimated regression models are used to calculate the F statistic. The second manual method, which is similar to that used in the *causality* function requires specifying a restriction matrix and then combining this matrix with the variance-covariance matrix of the unrestricted model to obtain the F statistic. The second manual approach is presented next.

Three matrices are required to generate the F statistic. The first is the matrix of estimated coefficients, which we will call Π. The second is a restriction matrix, R, which is used to assign the $\phi_{21}^1 = \phi_{21}^2 = \phi_{21}^3 = 0$ or $\phi_{12}^1 = \phi_{12}^2 = \phi_{12}^3 = 0$ restriction in equations (9.17) and (9.18). The third is the variance-covariance matrix of Π, which we call Σ. Calculating Σ is complicated because it must accommodate both equations (i.e., *premE* and *premW*). To simplify the analysis we will import and then use the *toMlm()* function which is used within to the *causality()* function. This function returns the Π and Σ matrices after we supply it with the estimated VAR model. Prior to implementing this procedure we must re-estimating the model with three lags rather than one lag.

```
var5 <- VAR(wood[,2:3], p = 3, type = "both", exogen = weather_season)

source(here("Data/ch8","toMlm.R"))
xMlm<-toMlm(var5)
PI <- coef(xMlm)
sigma.pi  <- vcov(xMlm)
PI.vec <- as.vector(PI)
```

In the previous chunk of code, *PI.vec* is a column vector consisting of the two columns of coefficient estimates from the Π matrix. The estimates for the *premE* equation are in the top half of *PI.vec* and the estimates for the *premW* equation are in the bottom half. Note that *PI.vec* has 28 columns because each equation has 14 variables: three lags for *premE*, three lags for *premW*, three temperature variables, three seasonal dummy variables, one time trend and one constant.

We must now specify the pair of restriction matrices: *Re*, for the test of the null hypothesis that the wood chip price premium does not Granger cause the heating oil price premium; and *Rw*, for the test of the null hypothesis that the heating oil price premium does not Granger cause the wood chip price premium. In the first case we are testing $\phi_{12}^1 = \phi_{12}^2 = \phi_{12}^3 = 0$ in equation (9.17) and in the second case we are testing $\phi_{21}^1 = \phi_{21}^2 = \phi_{21}^3 = 0$ in equation (9.18). To express this pair of restrictions in matrix format let $R_e = c(Z_1, Z_0)$ and $R_w = c(Z_0, Z_1)$ be 3x28 matrices where

$$Z_1 =$$

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{9.19}$$

and

$$Z_0 =$$

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{9.20}$$

The pair of restrictions can now be expressed as $R_e\Pi = 0$ and $R_w\Pi = 0$. We can create these restriction matrices as follows:

```
r1e <- c(0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
r2e <- c(0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
r3e <- c(0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
Re <- rbind(r1e,r2e,r3e)

r1w <- c(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0)
r2w <- c(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0)
r3w <- c(0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0)
Rw <- rbind(r1w,r2w,r3w)
```

We now have the components which are required to formally test the linear restrictions which correspond to the Granger null hypotheses. The matrix formula for the F statistic for a test of a linear restriction in a regression model is explained in the following teaching note: http://users.wfu.edu/cottrell/ecn215/linrestrict.pdf. The formula is

$$F = \left(R\hat{\beta}\right)' \left[R\Sigma R'\right]^{-1} \left(R\hat{\beta}\right) / N \tag{9.21}$$

Within equation (9.21) note that R is the restriction matrix, $\hat{\beta}$ is the matrix of estimated coefficients, Σ is the variance-covariance matrix for $\hat{\beta}$ and N is the number of restrictions (in our case 3).

Using equation (9.21) we can calculate the F statistic for the two sets of restrictions as follows:

```
N <- 3
Fe <- t(Re %*% PI.vec) %*% solve(Re %*% sigma.pi %*% t(Re)) %*% Re %*% PI.vec / N
Fe
```

```
##           [,1]
## [1,] 0.5604654
```

```
Fw <- t(Rw %*% PI.vec) %*% solve(Rw %*% sigma.pi %*% t(Rw)) %*% Rw %*% PI.vec / N
Fw
```

```
##          [,1]
## [1,] 3.40334
```

Before we deriving the p values which are associated with this pair of F statistics, let's confirm these F values using the *causality()* function. Following the same order as above, we will first test the null hypothesis that the price premium for wood chips Granger causes the price premium for heating oil. We will then test the opposite null hypothesis.

```
caus_w <- causality(var5,cause="premW")
caus_w$Granger
```

```
##
##  Granger causality H0: premW do not Granger-cause premE
##
## data:  VAR object var5
## F-Test = 0.56047, df1 = 3, df2 = 834, p-value = 0.6412
```

```
caus_e <- causality(var5,cause="premE")
caus_e$Granger
```

```
##
##  Granger causality H0: premE do not Granger-cause premW
##
## data:  VAR object var5
## F-Test = 3.4033, df1 = 3, df2 = 834, p-value = 0.0173
```

The F statistics are 0.560 for the wood chip Granger causes heating oil null hypothesis, and 3.4033 for the heating oil Granger causes wood chips null hypothesis. As expected, these values are the same as those which we derived using the manual matrix multiplication method.

The respective p values for the pair of tests are 0.6412 and 0.0173. This means we are unable to reject the null hypothesis that wood chip price premiums Granger cause heating oil price premiums but we are able to reject the converse. This uni-directional outcome helps to justify the exclusion restriction which is being used to identify the structural VAR.

9.5 Impulse Response

Impulse response is normally included in the results section of a VAR analysis. This section begins by theoretically deriving the impulse response function (IRF), first for the reduced form VAR and then for the structural VAR. This section concludes by presenting the calculated IRFs for the VAR which was estimated using artificially generated values for x_1 and x_2 and for the VAR which was estimated using price premiums for heating oil and wood chips.

Recall from Section 9.1.4 that the reduced form VAR with one lag can be expressed as

$$X_t = \Phi X_{t-1} + U_t \tag{9.22}$$

where:

$$X_t = \begin{bmatrix} x_{1t} \\ x_{2t} \end{bmatrix} \quad \text{and} \quad U_t = \begin{bmatrix} u_{1t} \\ u_{2t} \end{bmatrix} \quad \text{and} \quad \begin{array}{l} u_{1t} = b_{11}\epsilon_{1t} + b_{12}\epsilon_{2t} \\ u_{2t} = b_{21}\epsilon_{1t} + b_{22}\epsilon_{2t} \end{array}$$

The impulse response function (IRF) is a dynamic measure of the partial equilibrium impact on X_t from a shock to one of the error terms in period t. The reduced form IRF examines the impact on the future trajectories of x_{1t} and x_{2t} from a shock to one of the aggregate error terms (i.e., u_{1t} or u_{2t}). The reduced form IRF is usually not very informative because shocking the aggregate error term, $u_{1t} = b_{11}\epsilon_{1t} + b_{12}\epsilon_{2t}$, is not consistent with standard policy analysis. The structural IRF also measures the impact on x_{1t} and x_{2t} but in this case the shock is to one of the individual error terms (i.e., ϵ_{1t} or ϵ_{2t}). The structural IRF is normally of specific interest to researchers (e.g., how does a shock in the price premium of heating oil affect the price premium for wood chips over the next 12 months)?

The reduced form IRF can be derived by first expressing the VAR in its infinite moving average (MA) representation. Specifically, using recursive substitution similar to how the period n VAR forecast was generated in section 9.1.5, the MA representation of the reduced form VAR with one lag can be expressed as

$$X_t = U_t + \Phi U_{t-1} + \Phi^2 U_{t-2} + ... + \Phi^j U_{t-j} + ... \tag{9.23}$$

Within equation (9.23) the expression for Φ^i is given by is given by equation (9.9). Now it is obvious that Φ^j is a measure of the impact that a shock to U_t has on X_t after j periods. More generally, $\Phi^j(m, n)$ measures the response of m^{th} variable to the n^{th} reduced form error after j periods. If the stability conditions for the VAR are satisfied, then the reduced form IRF function will begin with a value of zero (since it is not capable of measuring same-period impacts), increase in absolute value and then gradually decline to zero as j

becomes sufficiently large. This standard pattern for the reduced form IRF is apparent in the simulation results, which are presented below.

We can now construct the formula for the structural VAR given the identification restriction that a shock to x_{1t} impacts x_{2t} in the same period but not vice versa. Specifically, we can use Cholesky decomposition to find a lower triangular matrix, P, where $PP' = \Sigma$ and Σ is the variance-covariance matrix for the reduced form VAR. Now define

$$\tilde{U}_t = P^{-1}U_t \tag{9.24}$$

The variance-covariance of this transformed error matrix, $\tilde{U}_t$, is orthogonal, which means that all of the off-diagonal elements of this matrix are equal to zero. We can now multiply equation (9.23) through by PP^{-1} to obtain

$$PP^{-1}X_t = PP^{-1}U_t + PP^{-1}\Phi U_{t-1} + PP^{-1}\Phi^2 U_{t-2} + ... + PP^{-1}\Phi^j U_{t-j} + ... \tag{9.25}$$

If we substitute in $\tilde{U}_t = P^{-1}U_t$ and note that PP^{-1} is the identity matrix, a revised expression for equation (9.25) can be expressed as

$$X_t = U_t + \Phi P\tilde{U}_{t-1} + \Phi^2 P\tilde{U}_{t-2} + ...\Phi^j P\tilde{U}_{t-j} + ... \tag{9.26}$$

Within equation (9.26) the off-diagonal elements of the variance-covariance matrix for $\tilde{U}_{t-j}$ are equal to zero. This means that shocking the first and second elements in $\tilde{U}_t$ is equivalent to shocking ϵ_{1t} and ϵ_{1t}, respectively. Thus, given the restriction that shocks to x_{1t} affect x_{2t} but not vice versa, we can interpret $\Phi^j(m,n)P$ as the response of m^{th} variable to the n^{th} structural error after j periods. In other words, $\Phi^j P$ is the j-period structural IRF.

9.5.1 IRF With Simulated VAR Data

In Section 9.2 artificial data was created for a two-variable VAR with one lag. The data was created with a lower triangular variance-covariance matrix to ensure that the coefficients of the structural VAR are identified. In this section this artifical data is used to first generate a reduced form IRF, and then a structural IRF.

To generage the reduced form IRF manually it is necessary to create a matrix which contains estimates of the reduced form coefficients (i.e., the Φ matrix). These estimates can be extracted from the *var_est* object, which was created in section 9.2.

```
Phi <- rbind(t(coef(var_est)$X1[,1]),t(coef(var_est)$X2[,1]))
Phi
```

```
##           X1.l1         X2.l1
## [1,] 0.9610078 -0.0004258584
## [2,] 0.1966216  0.8241393070
```

With Φ constructed, equation(9.9) can be used to manually generate n values for the reduced form IRF. Let's use $n = 5$.

```
IRF_red <- diag(2)
for (i in 1:5){
  IRF_red <-IRF_red %*% Phi
  print(IRF_red)
}
```

```
##           X1.l1         X2.l1
## [1,] 0.9610078 -0.0004258584
## [2,] 0.1966216  0.8241393070
##           X1.l1         X2.l1
## [1,] 0.9234522 -0.0007602199
## [2,] 0.3509985  0.6791218644
##           X1.l1        X2.l1
## [1,] 0.8872952 -0.001019787
## [2,] 0.4708423  0.559541547
##           X1.l1        X2.l1
## [1,] 0.8524971 -0.001218309
## [2,] 0.5625010  0.460939671
##           X1.l1        X2.l1
## [1,] 0.8190168 -0.001367099
## [2,] 0.6311986  0.379638955
```

Note that within these results the bottom left values in each set are of particular interest because they measure how a one unit increase in x_1 affects the value of x_2 in future periods. Because we are working with a reduced form VAR there is no same-period impact. Thus, the first value (0.196) is a measure of the impact one period later. The second value (0.351) is a measure two periods later, etc.

These reduced form IRF results can be verified by using the *irf* function in the *vars* package. This function requires specifying which variable is shocked (i.e., impulsed) and which variable is having the impact measured. To ensure the IRF is reduced form, we must set *ortho = FALSE*. Let's plot the IRF values from this function for $n = 30$ periods.

```
irf_pac <- irf(var_est, impulse = "X1", response = "X2",
               n.ahead = 30, ortho = FALSE, boot = FALSE)
```

```
plot(irf_pac)
```

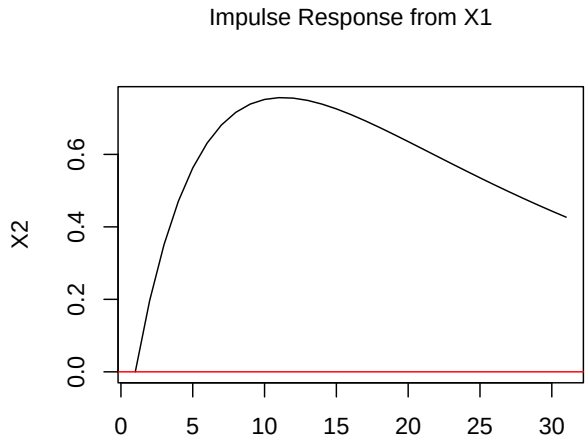

Figure 9.7: Reduced Form IRF: X1 impact on X2

The IRF values for the first five periods in Figure 9.7 are the same as those which were generated manually. Figure 9.7 shows more generally that the IRF peaks after about 12 periods and declines steadily for subsequent time periods. As was discussed above, this particular shape is expected.

To construct the IRF for the structural VAR it is first necessary to use Cholesky decomposition of the reduced form variance-covariance matrix to find values for the P matrix.

```
Sigma <- summary(var_est)$covres
P <- t(chol(Sigma))
```

With the P matrix in hand, we can use equation (9.26) to manually generate values for the first five periods of the structural IRF:

```
IRF_str <- diag(2)
for (i in 1:5){
  IRF_str <-IRF_str %*% Phi
  print(IRF_str %*% P)
}
```

```
##              X1           X2
## [1,] 0.4714433 -0.000133432
## [2,] 0.3163712  0.258223361
##              X1            X2
## [1,] 0.4529259 -0.0002381958
## [2,] 0.3534299  0.2127857861
##              X1            X2
## [1,] 0.4351148 -0.0003195247
## [2,] 0.3803305  0.1753182959
##              X1            X2
## [1,] 0.4179868 -0.0003817265
## [2,] 0.3989983  0.1444238734
##              X1            X2
## [1,] 0.4015186 -0.0004283462
## [2,] 0.4110154  0.1189503353
```

Let's once again verify these values by generating the structural IRF with the *irf* function. To ensure that the IRF is structural it is necessary to use the *ortho=TRUE* option.

```
irf_str <- irf(var_est, impulse = "X1", response = "X2",
               n.ahead = 30, ortho = TRUE, boot = FALSE)
```

```
plot(irf_str)
```

Similar to the reduced form comparison, the structural IRF values in Figure 9.8 match the corresponding IRF values which were generated manually. Keep in mind the manual structural IRF method assumes a one unit shock and the *irf* function structural IRF method assumes a one standard deviation shock. In this particular case the standard deviation of the shock is equal to one, and so the values match without any adjustment for a non-unit standard deviation of the shock.

An importance difference between the reduced form IRF shown in Figure 9.7 and the structural IRF shown in Figure 9.8 is that the first value is zero in the reduced form case and a positive value in the structural case. Figure 9.9 below shows that for the case where x_2 is shocked and x_1 is impacted, the first period impact is zero for the structural VAR. This outcome is expected because the

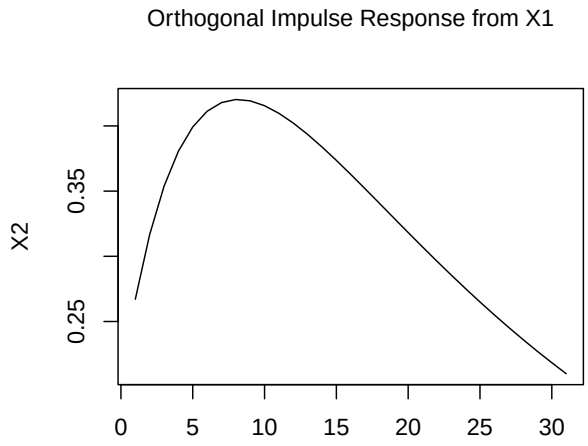

Figure 9.8: Structural IRF: X1 impact on X2

impact was forced to equal zero as part of the coefficient identification strategy. More generally, Figure 9.9 shows that the IRF in this reverse case is close to zero for all time periods.

```
irf_str2 <- irf(var_est, impulse = "X2", response = "X1",
            n.ahead = 30, ortho = TRUE, boot = FALSE)

plot(irf_str2)
```

9.5.2 IRF With Heating Oil and Wood Chip VAR

We conclude this section be calculating the structural IRF for the wood chip - heating oil price premium VAR, which was estimated in section 9.3. Given that the heating oil market is much larger than the wood chip market, it is acceptable to assume that shocks in the heating oil market have same-period impacts in the wood chip market but not vice versa. It is for this reason that the price premium for heating oil is the first variable and the price premium for wood chips is the second variable in the VAR regression (remember, the variables should be entered in order of decreasing exogeneity).

The *irf* function with *ortho=TRUE* can be used to generate a graph of the

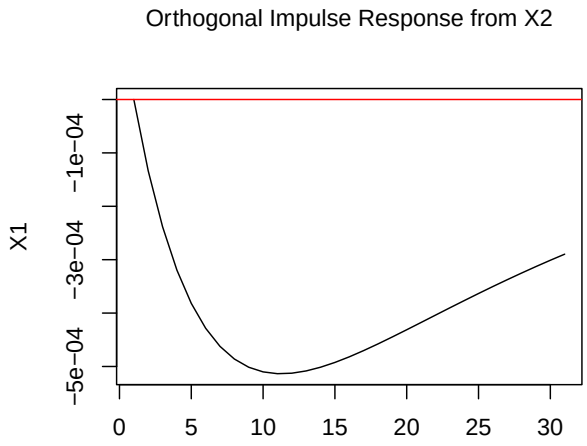

Figure 9.9: Structural IRF: X2 impact on X1

structural IRF for wood chips and heating oil. For this case we will also graph the bootstrapped error bands for the impulse response coefficients.

```
irf_heat <- irf(var4, impulse = "premE", response = "premW",
            n.ahead = 30, ortho = TRUE, boot = TRUE, runs = 500, ci = 0.9)
```

```
plot(irf_heat)
```

Figure 9.10 reveals that the structural IRF begins at approximately zero, dips below zero for about 5 months and then rises to a positive value. After about 12 months the shock to the heating oil price premium very gradually fades away. The error bounds are quite large and so overall the impact of a shock in the heating oil market on current and future price premiums in the wood chip market is relatively small. If we were to generate a graph of the reduced form IRF we would see that the shape is similar to that of the structural IRF but the vertical axis scaling is quite different. This difference reflects the fact that the shock is one unit in the reduced form case and one standard deviation in the structural case.

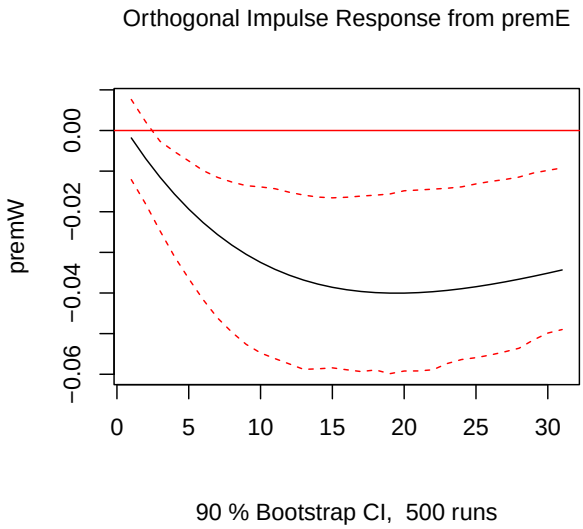

Figure 9.10: Reduced Form IRF: Heat Oil impact on Wood Chips

9.6 Conclusions

This chapter began our analysis of food and energy prices from a systems perspective. The vector autoregression (VAR) model which was featured in this chapter is used extensively in time series econometrics. Indeed, entering "vector autoregression" into Google Scholar returned close to 6000 articles which were posted during the 2019 - 2022 four year period. For the analysis of food and energy prices there are strong advantages to modeling these prices as a system rather than individually. This is particularly true when the prices are interconnected both vertically and horizontally. In this chapter we restricted our attention to a two-equation VAR. In the time series econometrics literature a VAR with three of four equations is typical.

For the particular scenario considered in this chapter it was reasonable to assume that shocks to price premium of heating oil impacts the price premium of wood chips but not vice versa. This exclusion restriction is reasonable because it aligns with our knowledge of the comparative size of these two markets and also with the results of Granger causality testing. There are other options for achieving identification. A popular second option is to impose restrictions on the long run impact of a shock within one of the two sectors [Blanchard and Quah, 1989]. This is relevant for scenarios where the VAR is estimated in first differences since the variables in levels are non-stationary. The identifying restriction is

that one of the shocks can permanently impact both variables when measured in levels whereas the other shock can only have a temporary effect (i.e., no long run permanent impact).

As noted in the Introduction, an important limitation of VAR analysis is that all included variables must be stationary or trend stationary. This seldom holds for the case of food and energy prices. The standard procedure to deal with this limitation is to estimate the model in first differences rather than levels. While this procedure is effective at achieving stationarity, the process of differencing does commonly result in lost information. In this paper stationarity was achieved by using relative prices rather than actual prices. This method also has limitations because theory typically has less to say about the relationship between pricing premiums and discounts versus the actual prices.

The next two chapters are devoted to the concept of cointegration. We will see that if the prices in the VAR are non-stationary but are cointegrated then the system of equations can be estimated as a vector error correction model (VECM) rather than as a VAR. This is important because an estimated VECM contains valuable information which is lost if a VAR is estimated with differenced data. When estimating a VECM there is no distinction between a reduced form and structural version of the model. Nevertheless, the estimated coefficients of a VECM have important economic interpretations similar to those within the structural VAR.

9.7 Appendix

This Appendix is used to establish that the heating oil and wood chip price premium variables are stationary. The testing procedure comes from Chapter 5. We begin by generating the set of ADF test statistics for the heating oil price premium.

```
adf_heat <- list(
  trend = ur.df(wood$premE, type = "trend", selectlags = "AIC"),
  drift = ur.df(wood$premE, type = "drift", selectlags = "AIC")
)

heat_trend <- summary(adf_heat$trend)@teststat
heat_drift <- summary(adf_heat$drift)@teststat
adf1 <- cbind(heat_trend,heat_drift)
```

Now repeat this process for the wood chip price premium variable.

```
adf_chips <- list(
  trend = ur.df(wood$premW, type = "trend", selectlags = "AIC"),
  drift = ur.df(wood$premW, type = "drift", selectlags = "AIC")
```

```
)

chips_trend <- summary(adf_chips$trend)@teststat
chips_drift <- summary(adf_chips$drift)@teststat
adf2 <- cbind(chips_trend,chips_drift)
```

The final step is to gather the ADF test statistics and display these along with the ADF critical values.

```
adf_wood <- rbind(adf1,adf2)
rownames(adf_wood)  <- c("Heating Oil", "Wood Chips")
adf_wood
```

```
##                  tau3     phi2     phi3      tau2     phi1
## Heating Oil -3.268659 3.858992 5.441901 -1.540425 1.527109
## Wood Chips  -3.826365 5.065453 7.330477 -2.278598 2.858586
```

```
rbind(adf_heat$trend@cval,adf_heat$drift@cval)
```

```
##       1pct  5pct 10pct
## tau3 -3.98 -3.42 -3.13
## phi2  6.15  4.71  4.05
## phi3  8.34  6.30  5.36
## tau2 -3.44 -2.87 -2.57
## phi1  6.47  4.61  3.79
```

The above results show that for the case of heating oil, the null hypothesis of a unit root and/or no time trend in the price premium can be rejected at the 10 percent level since the ϕ_3 test statistic (5.785) exceeds the ϕ_3 critical value (5.36). Similarly, the null hypothesis of no time trend can be rejected since the τ_3 test statistic (-3.352) is less than the τ_3 critical value (-3.13). We can therefore conclude with $\alpha = 0.10$ that the heating oil price premium does not contain a unit root.

For the case of wood chips, the null hypothesis of a unit root and/or no time trend in the price premium can be rejected at the 5 percent level since the ϕ_3 test statistic (8.000) exceeds the ϕ_3 critical value (6.30). Similarly, the null hypothesis of no time trend can be rejected since the τ_3 test statistic (-3.998) is less than the τ_3 critical value (-3.98). We can therefore conclude with $\alpha = 0.05$ that the wood chip price premium does not contain a unit root.

Chapter 10

Cointegration

In the previous chapter on vector autoregression (VAR) we recognized that sets of related commodity prices should be modeled as a system rather than individually. This is particularly important when modeling pricing relationships in vertical markets such as corn and ethanol, or when commodities are strong substitutes such as soybeans, canola and palm oil. However, the VAR requirement that prices included in the system are stationary is an important limitation. Most commodity prices are $I(1)$ (i.e., integrated of order one) and so due to their non-stationarity must enter the VAR in first differences. Analyzing prices in first differences is useful for short run analysis but doing so comes at the expense of losing information about long run economic relationships.

Fortunately, the concept of cointegration allows us to potentially bypass this limitation of VAR analysis. In statistics the order of integration is a measure of the minimum number of differencing which is required to make a data series stationary and cointegration is a related measure which is relevant for systems of time series variables. If commodities such as soybeans, canola and palm oil have prices which are all $I(1)$ and these prices prove to be cointegrated then even though they are non-stationary they can be modelled as a system. A system of cointegrated non-stationary prices should be be analyzed in a vector error correction model (VECM) rather than a VAR. This is the topic of the next chapter.

According to Engle and Granger [1987],

If each element of a vector of time series x, first achieves stationarity after differencing, but a linear combination $a'x$, is already stationary, the time series x, are said to be co-integrated with co-integrating vector a.

The co-integrating vector is particularly important for commodity analysis because it reflects the long-run equilibrium pricing relationship. For the case of energy and food prices the long-run relationship is typically implied by arbitrage

and the law-of-one price (LOP). In a cointegrated system with n equations we should expect $0 < r < n$ cointegrating vectors. For example, in a system consisting of the prices of natural gas, electricity and nitrogen fertilizer, we expect two cointegrating vectors, one resulting from the LOP which connects natural gas and electricity prices, and a second which connects natural gas and nitrogen fertilizer prices. The price of natural gas is likely the common trend (i.e., has the primary unit root) and as such is the cause of the non-stationarity in electricity and nitrogen fertilizer prices.

Testing for commodity price cointegration is relatively straight forward in the case of two commodities because in this case there is either no cointegrating vector (which implies no cointegration) or one cointegrating vector. In two commodity systems the Engle-Granger procedure can be used to test for the presence of this single cointegrating vector in the pair of commodity prices. Testing for cointegration is more complicated when there are more than two prices in the system because in this case there may be more than one cointegrating vector. As we will see below, the number of cointegrating vectors is related to the system eigenvalues. The Johansen test for cointegration exploits the relationship between statistically significant eigenvalues and statistically significant cointegrating vectors.

Testing for cointegration can only proceed if the prices are both non-stationary and integrated of the same order. Establishing this outcomes requires two ADF unit root tests for each set of prices. The first test requires prices in levels in order to verify that each price series has a unit root. The second test requires prices in first differences in order to verify that each price in levels is $I(1)$. This means that much of the overall analysis of cointegration is devoted to unit root testing of the individual price series.

When working through the rest of this chapter it is useful to keep the discussion and literature review from Chapter 1 in mind. There it was emphasized that cointegration testing is an important tool in policy analysis. Recall that there is a sizeable literature which examines the food for fuel debate by testing whether or not food prices and energy prices are cointegrated. Similarly, policy makers worry that large-scale holdings of commodity futures by hedge funds serve to drive up the price of food. This issue has been examined by testing for cointegration between the futures price of a food commodity and a broad stock market index such as the S&P 500. Cointegration is also commonly used to test for market efficiency such as the spatial law-of-one price, intertemporal pricing relationships and spot-futures price arbitrage.

In the next section we will briefly examine the theory of cointegration. The remainder of the chapter is then devoted to empirical testing for cointegration using a vegetable oil data set and a biofuel data set. In Section 10.2 the data sets are introduced and the results from one of four unit root tests are presented (the remaining three tests are in the Appendix). In Section 10.3 the Engle-Granger cointegration testing procedure is explained and then put to work using a pair of prices from the biofuel data set. In Section 10.4 the Johansen cointegration

testing procedure is explained and then put to work on both data sets. A summary and concluding comments are provided in Section 10.5

10.1 Cointegration Theory

In this section we will examine the theory of cointegration, beginning with a simple two-equation system with one cointegrating vector and ending with a more complex four-equation system with two cointegrating vectors.

10.1.1 Two Equation System

Let's begin with a simple system consisting of the price of wheat and the wholesale price of flour, both of which have a unit root. Wheat is the primary input in the production of flour and so we expect a long run relationship between the prices of these two commodities. For this reason it is natural to assume cointegration for this pair of prices.

To keep things simple, suppose the price of wheat follows a pure random walk, $P_t^w = P_{t-1}^w + e_t^w$. The wholesale price of flour is assumed to be a linear function of the price of wheat, $P_t^f = a + bP_t^w + e_t^f$. To further simplify, assume the intercept, a, is zero and so we have $P_t^f = bP_t^w + e_t^f$. It should be clear that P_t^w and P_t^f are both $I(1)$ stochastic processes. Our system of equations can now be written as follows.

$$\begin{aligned} P_t^f &= bP_t^w + e_t^f \\ P_t^w &= P_{t-1}^w + e_t^w \end{aligned} \tag{10.1}$$

which ensure $\beta_1 P_t^f + \beta_2 P_t^w$ is stationary. One solution is $\beta = (1, -b)$ where b can be interpreted as the amount of wheat it takes to produce one unit of flour (i.e., the LOP implies $P^F = bP^w$ in the long run). The linear combination $\beta_1 P_t^f + \beta_2 P_t^w$ gives $bP_t^w - bP_{t-1}^w + e_t^f - be_t^w$, which reduces to e_t^f after substituting in equation (10.1). This linear combination is clearly stationary.

The cointegrating vector $\beta = (1, -b)$ is not unique because for all values of ϕ the cointegrating vector $\beta = (\phi, -\phi b)$ also works to achieve stationarity. Typically the first element of the cointegrating vector is set equal to 1 in order to obtain a unique set of values.

In this simple case it was fairly obvious that $\beta = (1, -b)$ is a cointegrating vector. A more formal procedure requires writing the pair of equations in matrix format as follows.

$$\begin{bmatrix} 1 & -b \\ 0 & 1 \end{bmatrix} \begin{bmatrix} P_t^f \\ P_t^w \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} P_{t-1}^f \\ P_{t-1}^w \end{bmatrix} + \begin{bmatrix} e_t^f \\ e_t^w \end{bmatrix} \tag{10.2}$$

This system of equations can be solved to give

$$\begin{bmatrix} P_t^f \\ P_t^w \end{bmatrix} = \begin{bmatrix} 0 & b \\ 0 & 1 \end{bmatrix} \begin{bmatrix} P_{t-1}^f \\ P_{t-1}^w \end{bmatrix} + \begin{bmatrix} e_t^f + be_t^w \\ e_t^w \end{bmatrix} \tag{10.3}$$

Defining $\Delta P_t^f = P_t^f - P_{t-1}^f$ and $\Delta P_t^w = P_t^w - P_{t-1}^w$, equation (10.3) can be expressed as first differences.

$$\begin{bmatrix} \Delta P_t^f \\ \Delta P_t^w \end{bmatrix} = \begin{bmatrix} -1 & b \\ 0 & 0 \end{bmatrix} \begin{bmatrix} P_{t-1}^f \\ P_{t-1}^w \end{bmatrix} + \begin{bmatrix} e_t^f + be_t^w \\ e_t^w \end{bmatrix} \tag{10.4}$$

Equation (10.4) is a simple version of the vector error correction model (VECM), which is featured in the next chapter. The "VECM" name of this model stems from the fact that deviations in P_t from its underlying cointegration vector are stationary and so there is an ongoing "error correction". Later in this chapter when the Johansen cointegration testing procedure is featured, this simple VECM is enhanced by allowing for intercepts, deterministic trends and lags.

In equation (10.4) the leading coefficient matrix

$$\Pi = \begin{bmatrix} -1 & b \\ 0 & 0 \end{bmatrix}$$

is important because its rank tells us the number of unique cointegration vectors in the system. As well, Π can be decomposed to give us the specific elements of these vectors (more on this below). In this particular case the rank of Π, which is the number of linearly independent column vectors (or row vectors), is equal to one. In this example with two non-stationary prices we shouldn't expect a rank of two because full rank implies that all of the variables in the system are individually stationary. We also shouldn't expect a rank of zero since that would imply that all variables are non-stationary and there are no cointegrating vectors.

We can extract the single cointegrating vector from Π by decomposing it. Specifically, let $\Pi = \alpha\beta'$ where α and β both have one column and two rows. The β vector is the cointegrating vector which we are wishing to extract. Solving this system gives $\alpha_1\beta_1 = -1$, $\alpha_1\beta_2 = b$, $\alpha_2\beta_1 = 0$ and $\alpha_2\beta_2 = 0$. The solution is $\beta_1 = 1$, $\beta_2 = -b$, $\alpha_1 = -1$ and $\alpha_2 = 0$. The $\beta_1 = 1$ and $\beta_2 = -b$ solution is consistent with the cointegrating vector which was derived above. The α_1 and α_2 are speed of adjustment parameters, which are discussed in greater detail in the next chapter.

10.1.2 Additional Considerations

The previous model was constructed assuming a pure random walk for wheat. The $(1, -b)$ cointegrating vector remains unchanged if the price of wheat instead

follows a random walk with drift (i.e., $P_w^t = a + P_{t-1}^w + e_t^w$). Adding an intercept term to the flour pricing equation and adding a deterministic trend to either pricing equation complicates the analysis somewhat and adds little intuition in this theoretical section. Consequently, these features are omitted. In the empirical analysis the econometric model allows for intercepts, trends and drifts.

Suppose a third "other" commodity, P_t^o is added to the system. Assume either $P_t^0 = \gamma P_t^W + e_t^0$ or $P_t^0 = \gamma P_t^f + e_t^o$. In other words, this new commodity is not in itself a common trending variable but it is non stationary by virtue of being proportional to either the price of wheat or the price of flour. In both cases it is straight forward to show that a second cointegrating vector emerges which reflects the long run direct or indirect relationship between P_t^o and P_t^w .

Suppose instead the price of the third "other" commodity was also a random walk. Specifically, $P_t^o = P_{t-1}^o + e_t^o$. If there is no long term relationship between P_t^o and either P_t^f or P_t^w then what is the consequence of adding P_t^0 as the third variable in the system of equations? It can be shown that there will be just one cointegrating vector in the three equation system, and the last element of the vector will be zero.

Prior to testing for the presence of cointegration and estimating the elements of the cointegrating vectors it is always a good idea to formulate a theoretical model which will guide the empirical analysis. For example, suppose you are interested in testing if the price of soybeans and the price of soybean oil are cointegrated. A 60 pound bushel of soybeans will produce about 11 pounds of soybean oil, which means that based on the law-of-one price we should expect $P^{beans} = \frac{11}{60} P^{oil}$ in the long run. Of particular interest in the empirical analysis is how close the value from the estimated cointegration vector is to this proposed $P^{beans} = \frac{11}{60} P^{oil}$ pricing relationship.

10.1.3 Blended Prices

Consider a three equation system consisting of the prices of ethanol (P_t^e), gasoline (P_t^g) and corn (P_t^c). There is no long run equilibrium relationship between the price of gasoline and the price of corn. Consequently, these two prices are modeled as individual random walks with an upward drift. The price of ethanol depends on both the price of gasoline, which is a key substitute for ethanol, and the price of corn, which is a key input for producing ethanol. Thus, the price of ethanol is a blended price. The three equation system can therefore be expressed as

$$\begin{aligned} P_t^e &= b_1 P_t^g + b_2 P_t^c + e_t^e \\ P_t^g &= P_{t-1}^g + e_t^g \\ P_t^c &= P_{t-1}^c + e_t^c \end{aligned} \tag{10.5}$$

This three-equation system has only one cointegrating vector because the price of gasoline and corn are independent within the system. It is important to keep in mind that individual pairs of prices are not necessarily cointegrated in a system which is integrated as a whole. In this current case the price of gasoline and the price of corn are not cointegrated if they were to be modeled as a two-commodity system .

The single cointegrating vector for this three-equation system is given by $\beta = (1, -b_1, -b_2)$. This can be confirmed by substituting the expressions from (10.6) into $P_t^e - b_1 P_t^g - b_2 P_t^c$ and noting that the resulting expression is stationary since it reduces to e_t^e.

Let's add the price of distillers' dried grains (DDG) to the system of equations and assume that it is also a blend price. DDG is an ethanol industry by-product which is used for livestock feed. The price of DDG depends on both the price of ethanol and the price of corn. Specifically, $P_t^d = \delta_1 P_t^e + \delta_2 P_t^c + e_t^d$. Our system of equations can be written as

$$\begin{aligned} P_t^e &= b_1 P_t^g + b_2 P_t^c + e_t^e \\ P_t^g &= P_{t-1}^g + e_t^g \\ P_t^c &= P_{t-1}^c + e_t^c \\ P_t^d &= \delta_1 P_t^e + \delta_2 P_t^c + e_t^d \end{aligned} \tag{10.6}$$

It is likely that this system has two cointegrating vectors but the elements of this second vector are not immediately obvious. For example, should this new vector only account for direct effects (in which case $\beta_2 = 0$ since P_t^g is excluded) or both direct and indirect effects (in which case $\beta_2 \neq 0$).

Let's use the matrix decomposition method to derive the specific values for the second cointegration vector, which we expect to be present. The steps are similar to those used in the previous section. Begin by rewriting equation (10.6) as

$$\begin{bmatrix} 1 & -b_1 & -b_2 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\delta_1 & 0 & -\delta_2 & 1 \end{bmatrix} \begin{bmatrix} P_t^e \\ P_t^g \\ P_t^c \\ P_t^d \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_{t-1}^e \\ P_{t-1}^g \\ P_{t-1}^c \\ P_{t-1}^d \end{bmatrix} + \begin{bmatrix} e_t^e \\ e_t^g \\ e_t^c \\ e_t^d \end{bmatrix} \tag{10.7}$$

The next step is to pre-multiply equation (10.7) by the inverse of the first matrix and then convert the system into first differences. The system can now be written as

$$\begin{bmatrix} \Delta P_t^e \\ \Delta P_t^g \\ \Delta P_t^c \\ \Delta P_t^d \end{bmatrix} = \begin{bmatrix} -1 & b_1 & b_2 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & b_1\delta_1 & b_2\delta_1 + \delta_2 & -1 \end{bmatrix} \begin{bmatrix} P_{t-1}^e \\ P_{t-1}^g \\ P_{t-1}^c \\ P_{t-1}^d \end{bmatrix} + \begin{bmatrix} v_t^e \\ v_t^g \\ v_t^c \\ v_t^d \end{bmatrix} \tag{10.8}$$

In equation (10.9) the v_t^j error terms are linear functions of the original e_t^i error terms. The specific expressions are not shown because they are not important for this long run analysis. It should be clear that the coefficient matrix in equation (10.9), which is typically labeled Π, has a rank of two since two of the four rows only have zeros. From this information we know the system has two cointegrating vectors.

We don't need to solve for the first cointegrating vector because it remains unchanged other than adding a zero at the end. Specifically, $\beta^1 = (1, -b_1, -b_2, 0)$. The reason it doesn't change is because the long run relationship between the prices of ethanol, gasoline and corn are not affected by the addition of the price of DDG to the system. Let's guess that the second cointegrating vector reflects the direct linkages only, which means $\beta_2 = 0$ (i.e., P_t^g is excluded). In this case the proposed solution is $\beta^2 = (-\delta_1, 0, -\delta_2, 1)$.

Recall that the coefficient matrix in equation (10.9) can be decomposed into $\alpha\beta'$ where α and β each have four rows and two columns. The $\alpha\beta'$ system with the guess value for the second cointegrating vector inserted can be written as

$$\begin{bmatrix} \alpha_{11} & \alpha_{21} \\ \alpha_{21} & \alpha_{22} \\ \alpha_{31} & \alpha_{23} \\ \alpha_{41} & \alpha_{24} \end{bmatrix} \begin{bmatrix} 1 & -b_1 & -b_2 & 0 \\ -\delta_1 & 0 & -\delta_2 & 1 \end{bmatrix} = \begin{bmatrix} -1 & b_1 & b_2 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & b_1\delta_1 & b_2\delta_1 + \delta_2 & -1 \end{bmatrix} \tag{10.9}$$

This system can be solved to give $\alpha_{12} = \alpha_{13} = \alpha_{21} = \alpha_{22} = \alpha_{23} = 0$, $\alpha_{11} = -1$, $\alpha_{14} = -\delta_1$ and $\alpha_{24} = -1$. This solution confirms that our guess of $\beta^2 = (-\delta_1, 0, -\delta_2, 1)$ for the second cointegrating vector is a correct one. Of course the solution can also by confirmed by multiplying each expression in equation (10.6) by its respective value in the $\beta^2 = (-\delta_1, 0, -\delta_2, 1)$ vector and then noting that the resulting expression depends only on the stationary error terms.

10.1.4 Alternative Interpretation

It is useful to view the concept of cointegration from a more applied perspective. Let's use the prices of the three vegetable oils to illustrate this alternative perspective. The three prices are given by P_t^P, P_t^S and P_t^R for palm oil, soybean oil and rapeseed oil, respectively.

Let Z_t^i for $i = \{1, 2, 3\}$ be a set of exogenous non-stationary I(1) variables which influence the three prices (e.g., the price of crude oil, the interest rate and the exchange rate). Let ϵ_t^i for $i = \{P, S, R\}$ be a set of $I(0)$ error terms. The system of equations can be expressed as

$$\begin{aligned} P_t^P &= \gamma_1 Z_t^1 + \gamma_2 Z_t^2 + \gamma_3 Z_t^3 + \epsilon_t^P \\ P_t^S &= \delta_1 Z_t^1 + \delta_2 Z_t^2 + \delta_3 Z_t^3 + \epsilon_t^S \\ P_t^R &= \phi_1 Z_t^1 + \phi_2 Z_t^2 + \phi_3 Z_t^3 + \epsilon_t^R \end{aligned} \tag{10.10}$$

As noted in the previous section, cointegration requires that a linear combination of the three prices, $aP_t^P + bP_t^S + cP_t^R$, is stationary. This means that $aP_t^P + bP_t^S + cP_t^R$ must not depend on any of the three non-stationary Z_t^i variables. To derive conditions for cointegration multiply the first equation through by a, the second equation through by b and the third equation through by c. Now add the three equations together to obtain:

$$\begin{aligned} aP_t^P + bP_t^S + cP_t^R = &(a\gamma_1 + b\delta_1 + c\phi_1)Z_t^1 \\ &+ (a\gamma_2 + b\delta_2 + c\phi_2)Z_t^2 \\ &+ (a\gamma_3 + b\delta_3 + c\phi_3)Z_t^3 \\ &+ a\epsilon_t^P + b\epsilon_t^S + a\epsilon_t^R \end{aligned} \tag{10.11}$$

We can now define four cases:

CASE $r = 0$

Cointegration requires that all three Z_t^i variables disappear from the right side of equation (10.11). Thus, we seek expressions for a, b and c which solve:

$$\begin{aligned} a\gamma_1 + b\delta_1 + c\phi_1 &= 0 \\ a\gamma_2 + b\delta_2 + c\phi_2 &= 0 \\ a\gamma_3 + b\delta_3 + c\phi_3 &= 0 \end{aligned} \tag{10.12}$$

There is no solution in this particular case (the only solution is $a = b = c = 0$). The absence of a solution implies the absence of a cointegrating vector (i.e., $r = 0$). Thus, with all three Z_t^i values included on the right side of equations (10.10) and (10.11) the set of vegetable oil prices is not cointegrated.

CASE $r = 1$: Equations (10.10) and (10.11) with Z_t^3 excluded.

In this case with only Z_t^1 and Z_t^2 included in the right side of equation (10.11) cointegration requires expressions for a, b and c which solve:

$$\begin{aligned} a\gamma_1 + b\delta_1 + c\phi_1 &= 0 \\ a\gamma_2 + b\delta_2 + c\phi_2 &= 0 \end{aligned} \tag{10.13}$$

Equation (10.13) is a system of two equations and three variables. It has an infinite number of solutions for a, b and c but there is only one linear independent solution (normalized with $a = 1$), which we define as the cointegrating vector. In this case we say the system is cointegrated with $r = 1$ cointegrating vectors.

CASE $r = 2$: Equations (10.10) and (10.11) with Z_t^2 and Z_t^3 excluded.

In this case with only Z_t^1 included in the right side of equation (10.11) cointegration requires expressions for a, b and c which solve:

$$a\gamma_1 + b\delta_1 + c\phi_1 = 0 \tag{10.14}$$

Equation (10.14) is a system of one equation and three variables. As in the previous case there are an infinite number of solutions for a, b and c. However, in this case there are two rather than one linearly independent solutions (normalized with $a = 1$). We can therefore say that the system is cointegrated with $k = 2$ cointegrating vectors.

CASE $r = 3$: Equations (10.10) and (10.11) with no Z_t^i variables.

In this case with no Z_t^i variables included in the right side of equation (10.10) each price is stationary to begin with and so each linear combination of the three prices is also stationary. Cointegration is not a relevant concept in this case since there is no risk of a spurious regression due to non-stationary prices.

To better understand these results consider the third $r = 2$ case where only Z_t^1 exists on the right side of equation (10.10). Suppose Z_t^1 is the price of crude oil. We have a system where each of the three vegetable oil prices is influenced by the price of crude oil and an independent stationary shock. The three prices are cointegrated because relative to the price of crude oil, the three prices move relatively "tightly" together over time (i.e., there are no stochastic trends in the relative prices). If a second Z_t^i variable is added to the system a more restricted cointegration outcome remains. In the $r = 3$ case each vegetable oil price is influenced by its own unique non-stationary Z_t^i variable and so it is impossible for the system of vegetable oil prices to be cointegrated. In general, cointegration exists provided that the number of non-stationary Z_t^i variables is less than the number of prices in the system.

10.2 Data and Unit Root Testing

Two data sets are used for the empirical testing of cointegration. The first data set consists of the prices of three widely traded vegetable oils: palm, soybean and rapeseed. The second data set, which is labeled "biofuels", consists of the prices of crude oil, regular diesel, biodiesel and soybean oil. The prices for the three vegetable oils are believed to be cointegrated because they are strong substitutes in commercial food preparation and other industrial applications. After adjusting for energy content and quality differences, according to the LOP the prices of these three vegetable oils should be equal in the long run. Cointegration is expected in the biofuel data set because of both horizontal and vertical market linkages. The theoretical pricing relationship between soybeans and soybean oil was discussed in the previous section. A similar vertical relationship

will exist for soybean oil and biodiesel prices, and for diesel and crude oil prices. Regular diesel and biodiesel are obviously strong substitutes.

10.2.1 Price Visualization

Recall from Chapter 3 that the vegetable oil data set consists of monthly prices measured in U.S. dollars per metric tonne for palm oil, soybean oil and rapeseed oil, beginning in January of 2003 and ending in December of 2020. This section begins by reading in the data which was saved to a *RDS* file in Chapter 3. For this vegetable oil data set and the biofuel data set which we will examine below we will use the log of price rather than the price itself for both the visualization and the econometric analysis (the reasons for this transformation were discussed in Chapter 4).

The first step is load all of the required R packages for this chapter.

```
pacman::p_load(tidyverse, here, urca, vars, tsibble, feasts, tsibbledata, forecast)
```

With "po" short for palm oil, "so" short for soybean oil and "ro" short for rapeseed oil a sample of the logged price data looks as follows:

```
vegoil <- readRDS(here("data/ch9", "vegoils.RDS"))

lnveg <- vegoil %>% mutate(lnpo = log(palm_oil),lnso=log(soybean_oil), lnro = log(rapes
  dplyr::select(month,lnpo,lnso,lnro)
head(lnveg)
```

```
## # A tsibble: 6 x 4 [1M]
##      month  lnpo  lnso  lnro
##      <mth> <dbl> <dbl> <dbl>
## 1 2003 Jan  6.19  6.29  6.44
## 2 2003 Feb  6.17  6.26  6.37
## 3 2003 Mar  6.12  6.26  6.31
## 4 2003 Apr  6.09  6.29  6.32
## 5 2003 May  6.12  6.31  6.41
## 6 2003 Jun  6.14  6.30  6.42
```

It is useful to plot these vegetable oil prices on the same graph. To do this with a legend it is necessary to create a common variable ("log_price") and then stack the price series vertically.

```
stack_po <- as_tibble(lnveg)  %>%
  dplyr::select(month,lnpo) %>%
  rename(log_price = lnpo) %>%
```

```
  mutate(Oil = "Palm")

stack_so <- as_tibble(lnveg)  %>%
  dplyr::select(month,lnso) %>%
  rename(log_price = lnso) %>%
  mutate(Oil = "Soybean")

stack_ro <- as_tibble(lnveg)  %>%
  dplyr::select(month,lnro) %>%
  rename(log_price = lnro) %>%
  mutate(Oil = "Rapeseed")

stack_veg <- bind_rows(stack_po,stack_so, stack_ro)
```

Figure 10.1 reveals two features which are central to the analysis of this chapter. First, each individual price series appears to be non-stationary due to the presence of stochastic trends. Formal testing to establish non-stationary is provided below. Second, the three price series are highly correlated, with relatively stable prices differences over time. This feature is indicative of cointegration since the differences across the price series (which can be interpreted as a linear combination of the price series) appears to be stationary. Formal testing of cointegration is conducted below.

```
ggplot(stack_veg,aes(y = log_price,x = month,color = Oil)) +
  geom_line() +
  ggtitle("Monthly Average of Logged Prices")
```

Let's now visually analyze the logged prices of the four bioenergy commodities, which begin in week 15 of 2007 and end in week 7 of 2022. In Chapter 3 the four weekly bio-energy prices were cleaned and transformed into a tsibble dataframe. This section begins by reading in the bio-energy data which was saved to a *RDS* file in Chapter 3.

```
biofuel <- readRDS(here("data/ch9", "biofuel.RDS"))

lnfuel <- biofuel %>% mutate(lnbio = log(biodiesel),lnsoy=log(soyoil), lndies = log(diesel), lncr
  dplyr::select(week,lnbio,lnsoy,lndies, lncrude)
head(biofuel)
```

```
## # A tsibble: 6 x 5 [1W]
##       week biodiesel soyoil diesel crude
##     <week>     <dbl>  <dbl>  <dbl> <dbl>
## 1 2007 W15      3.1    29.9   2.84  62.6
## 2 2007 W16      3.1    29.3   2.88  63.1
```

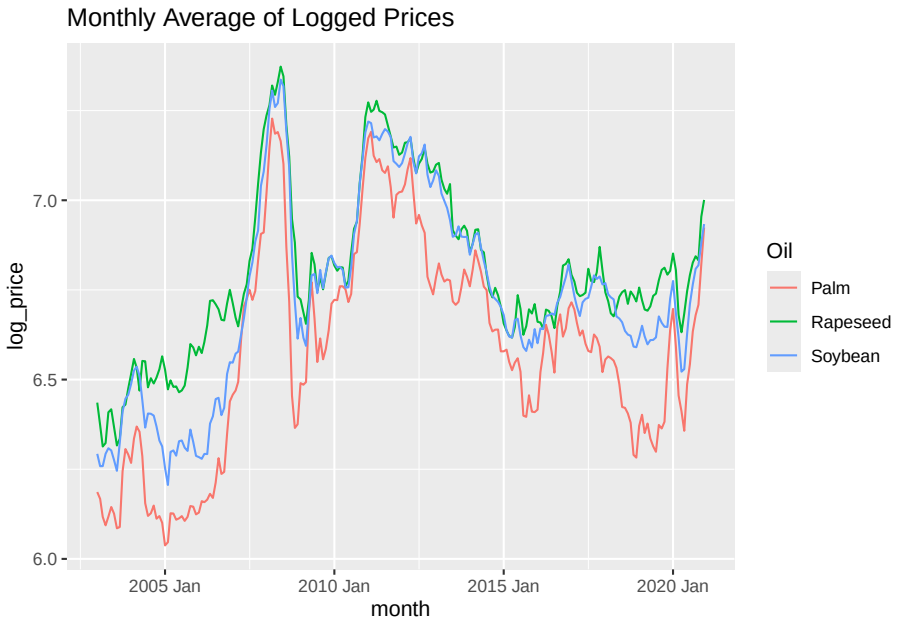

Figure 10.1: Log of Average Monthly Palm, Soybean and Rapeseed Prices

```
## 3 2007 W17      3.08   30.2   2.85  65.3
## 4 2007 W18      3.14   31.1   2.81  63.8
## 5 2007 W19      3.14   31.1   2.79  61.9
## 6 2007 W20      3.18   32.9   2.77  63.6
```

We will once again create a common variable and then stack the four bio-energy prices for graphing. It is useful to plot the log of the four prices on the same graph. To do this with a legend it is necessary to create a common variable ("log_price") and stack the price series vertically.

```
stack_bio <- as_tibble(lnfuel)  %>%
  dplyr::select(week,lnbio) %>%
  rename(log_price = lnbio) %>%
  mutate(Fuel = "Biodiesel, $/gallon")

stack_soy <- as_tibble(lnfuel)  %>%
  dplyr::select(week,lnsoy) %>%
  rename(log_price = lnsoy) %>%
  mutate(Fuel = "Soyoil, cents/lb")

stack_dies <- as_tibble(lnfuel)  %>%
  dplyr::select(week,lndies) %>%
```

```
  rename(log_price = lndies) %>%
  mutate(Fuel = "Diesel, $/gallon")

stack_crude <- as_tibble(lnfuel)  %>%
  dplyr::select(week,lncrude) %>%
  rename(log_price = lncrude) %>%
  mutate(Fuel = "Oil-Crude, $/barrel")

stack_fuel <- bind_rows(stack_bio,stack_soy, stack_dies, stack_crude)
```

Figure 10.2 shows that similar to the case of the three vegetable oils, each individual bio-energy price series is likely characterized by stochastic trends (the ADF tests below confirm this outcome). The bottom pair of schedules confirm the expected result that the prices of biodiesel and diesel are quite strongly correlated. The relationship between the price of biodiesel with the prices of soybean oil and crude oil is noticeably weaker. Later in this chapter formal tests for cointegration will be used to determine if these four prices are jointly cointegrated.

```
ggplot(stack_fuel,aes(y = log_price,x = week,color = Fuel)) +
  geom_line() +
  ggtitle("Log of Average Weekly Biodiesel, Soybean Oil, Diesel and Crude Oil Prices ")
```

10.2.2 Vegetable Oil ADF Unit Root Test

As previously noted, two unit root tests are required for each price series in each of the two data sets. The first test is used to show that the price series in levels has a unit root. The second test is used to show that the price series in first differences is stationary (i.e, the price series is difference stationary). If this outcome emerges for each price series then we can conclude that all prices are $I(1)$ and are therefore candidates for cointegration testing.

In this section the steps used in the ADF testing procedure are used here to test the null hypothesis of a unit root in the price levels of the three vegetable oils. The unit root tests for the vegetable oil prices in differences and for the biofuel prices in both levels and differences are contained in the Appendix.

Formal testing of stationarity is commonly preceded by an examination of the correlogram. Figure 10.3 shows for the case of palm oil that the autocorrelation function (ACF) fades very slowly. As was discussed in Chapter 5, a slowly facing ACF is a marker for a unit root (i.e., non stationarity). The purpose of this section is to use a formal ADF test to confirm this non-stationarity outcome for all three vegetable oil prices.

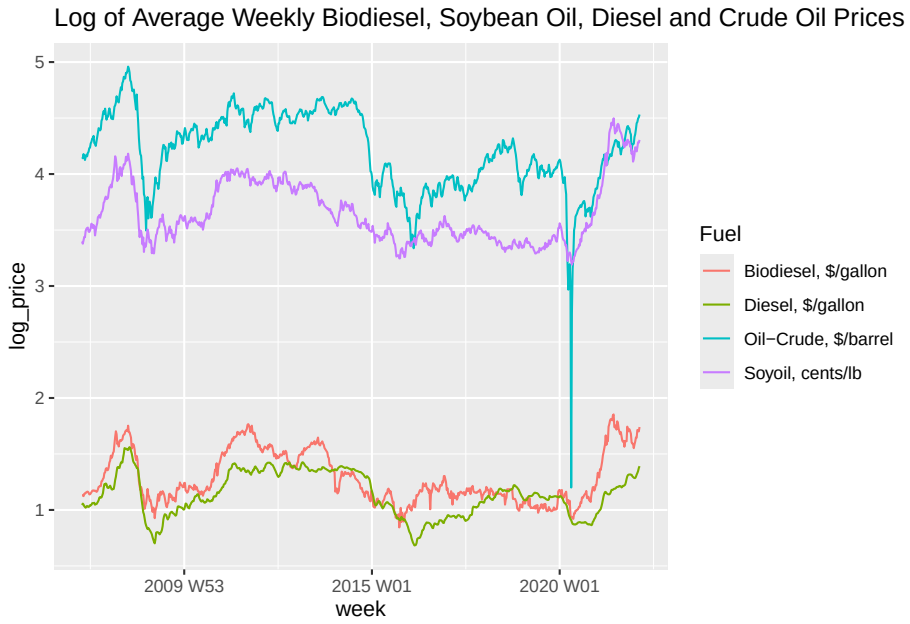

Figure 10.2: Monthly Weekly of Log Prices

```
lnveg %>% ACF(lnso) %>% autoplot()
```

In Chapter 5 a decision chart was used to assess the Augmented Dickey Fuller (ADF) test statistics and arrive at a conclusion regarding the stationarity status of the time series. For convenience this decision chart is repeated in Figure 10.4 below.

Recall that the top of the decision chart asks whether there is an obvious trend in the Δy_t series. Even if there is no obvious trend, which appears to be the case for the price plot in Figure 10.1, it is okay to begin at the top left corner of the decision chart. If after step one the test statistic reveals that there is no trend, then we will be returned to the top right corner of the decision chart. In other words, beginning at the top right rather than the top left serves only as a short cut within the testing procedure.

The decision chart uses the $\{\phi_3, \tau_3\}$ and $\{\phi_1, \tau_2\}$ test statistics and their associated critical values to assess stationarity. The $\{\phi_3, \tau_3\}$ and $\{\phi_1, \tau_2\}$ test statistics are generated with the *ur.df()* function with the *trend* option and *drift* option, respectively. For each of the three vegetable oil prices let's estimate the trend and drift versions of each ADF model, using the automatic lag selection in each case.

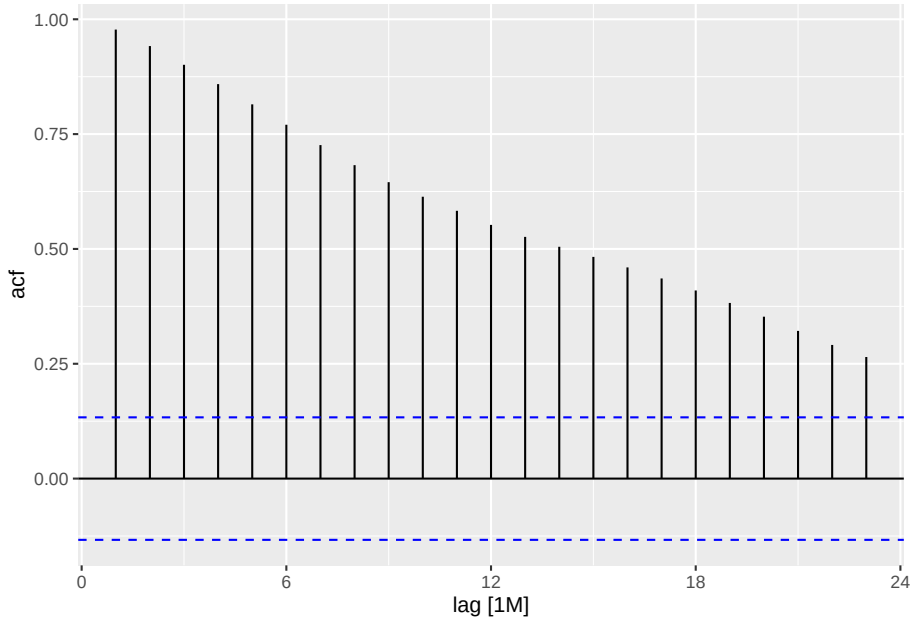

Figure 10.3: Correlogram of Palm Oil Price

```
adfpo <- list(
  trend = ur.df(lnveg$lnpo, type = "trend", selectlags = "AIC"),
  drift = ur.df(lnveg$lnpo, type = "drift", selectlags = "AIC")
)

adfso <- list(
  trend = ur.df(lnveg$lnso, type = "trend", selectlags = "AIC"),
  drift = ur.df(lnveg$lnso, type = "drift", selectlags = "AIC")
)

adfro <- list(
  trend = ur.df(lnveg$lnro, type = "trend", selectlags = "AIC"),
  drift = ur.df(lnveg$lnro, type = "drift", selectlags = "AIC")
)
```

Now generate the ϕ_i and τ_i test statistics

```
potrend <- summary(adfpo$trend)@teststat
podrift <- summary(adfpo$drift)@teststat
adf_po <- cbind(potrend,podrift)
```

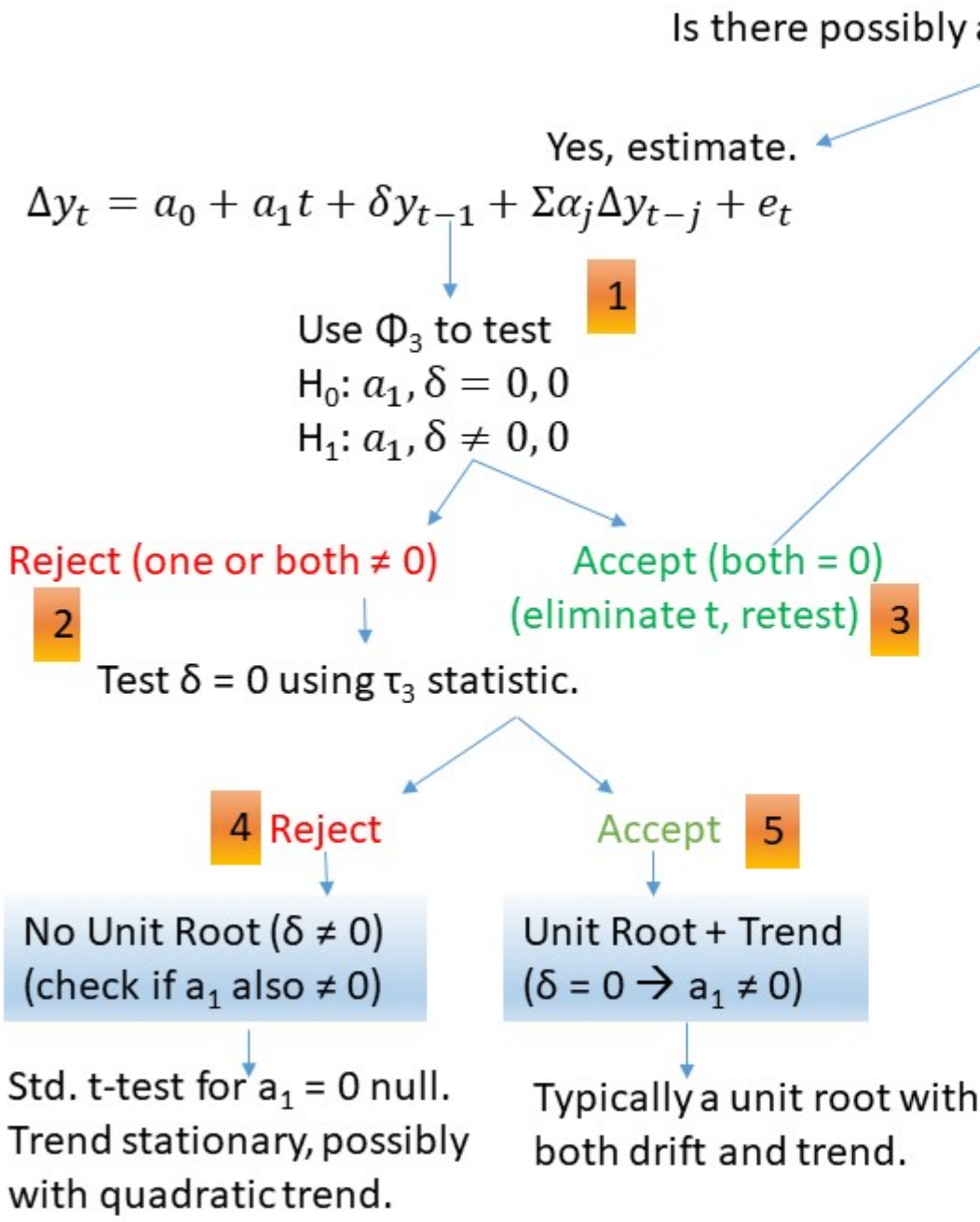

Figure 10.4: ADF Unit Root Decision Chart.

```
sotrend <- summary(adfso$trend)@teststat
sodrift <- summary(adfso$drift)@teststat
adf_so <- cbind(sotrend,sodrift)

rotrend <- summary(adfro$trend)@teststat
rodrift <- summary(adfro$drift)@teststat
adf_ro <- cbind(rotrend,rodrift)

adf_veg <- rbind(adf_po,adf_so,adf_ro)
rownames(adf_veg)  <- c("Palm Oil", "Soybean Oil", "Rapeseed Oil")
adf_veg
```

```
##                    tau3     phi2     phi3      tau2     phi1
## Palm Oil     -2.492929 2.191625 3.107612 -2.410342 3.085195
## Soybean Oil  -2.360723 2.065444 2.871296 -2.383614 3.068666
## Rapeseed Oil -2.468596 2.250485 3.134599 -2.505811 3.381800
```

Finally, generate the set of ADF critical values for the palm oil time series. The critical values are the same for all three vegetable oils and for the trend and drift options. Therefore, the palm oil critical values which are presented below can be interpreted as the common set of critical values for all testing in this section.

```
rbind(adfpo$trend@cval,adfpo$drift@cval)
```

```
##       1pct  5pct 10pct
## tau3 -3.99 -3.43 -3.13
## phi2  6.22  4.75  4.07
## phi3  8.43  6.49  5.47
## tau2 -3.46 -2.88 -2.57
## phi1  6.52  4.63  3.81
```

Notice that the absolute value of the ϕ_3 test statistic is less than its critical value for all three vegetable oils. This ruling out of a deterministic time trend implies that we should move to the right side of the decision chart and re-test. Notice that the absolute value of the ϕ_1 test statistic is less than its critical value for all three vegetable oils. According to the decision chart we should conclude that the log of the prices of all three vegetable oils have unit roots with no intercept (i.e., are pure random walks).

In the Appendix the ADF test on the first difference of the vegetable oil prices confirms that the unit root null hypothesis can be rejected for all three price series. It therefore follows that the three vegetable oil prices are all $I(1)$ and thus eligible to be include in a system for cointegration testing. ADF testing

in the Appendix confirms that the prices for diesel, biodiesel and soybean oil are all $I(1)$. The rather surprising result emerges that the price of crude oil is $I(0)$. It is therefore not possible to include crude oil in the system of equations for cointegration testing. The analysis proceeds with palm oil substituting for crude oil.

10.3 Engle-Granger Cointegration Test

As noted in the Introduction, the two most popular tests for cointegration are the Engle-Granger test, which is applicable for pairs of time series, and the Johansen test, which allows for larger sets of time series. The Engle-Grander test consists of two stages. In the first stage the pair of prices are regressed in order to estimate the parameters of the long run relationship, which is commonly referred to as the cointegrating vector. In the second stage a modified ADF test is used to test for stationarity of the residuals. When there are more than two commodity prices in the data set then the more sophisticated Johansen method must be used, In this case the rank of the adjusted coefficient matrix of an estimated vector error correction model (VECM) determines whether or not the set of prices is cointegrated.

There are two R packages which are used extensively in this chapter. The first is the *vars* package which contains a number of functions for estimating VAR models. The second is the *URCA* package, which contains the *ur.df()* function for initially testing stationarity of the prices and eventually testing for stationarity of the Engle-Granger regression residuals. The *URCA* package also contains the *ca.jo()* function, which is used for implementing the Johansen test for cointegration.

This section provides a conceptual overview of the Engle-Granger test and then applies this testing method to soybean oil and biodiesel prices. The following section features the Johansen test for cointegration.

10.3.1 Conceptual Overview of Engle-Granger Test

Recall that asking whether or not soybean oil and biodiesel prices are cointegrated is equivalent to asking whether or not there is a long run economic relationship between this pair of prices. We expect such a relationship exists because each gallon of biodiesel utilizes a fixed amount of soybean oil during the production process. This relationship suggests that the price of soybean oil may be proportional to the price of biodiesel. Specifically, our hypothesis is $P_t^S = \beta P_t^B + \epsilon_t$ where P_t^S is the price of soybean oil in week t, P_t^B is the price of biodiesel in week t, ϵ_t is an $I(0)$ error term and β is the proportionality constant.

Soybean oil is measured in cents per pound and biodiesel is measured in dollars per gallon. It takes about 7.6 pounds of soybean oil to produce a gallon of biodiesel. In week t the cost of the soybean oil within one gallon of biodiesel is therefor $7.6P_t^S$ cents or $0.076P_t^S$ dollars. A 2008 study (http://deq.mt.gov/files/Energy/EnergizeMT/EnergySites/Documents/Biodiesel_Production_Educ_Presentations/11Montana_Economics_Billings_Jan2008_JVG.pdf) estimated that soybean oil makes up about 85 percent of the total cost of producing biodiesel. In the long run if competition drives the prices of soybean oil and biodiesel to levels which allow refineries to just break even then we can conclude that $0.076 * 1.15P_t^S = P_t^B$. This expression can be rearranged to give $P_t^S/P_t^B = 11.44$

Figure 10.5 is a repeat of Figure 10.2 except it only the contains the logs of the soybean oil and biodiesel price series. If we take the logs of both sides of our long run economic relationship, $P_t^S/P_t^B = 11.44$, we get $log(P_t^S) - log(P_t^B) = 2.437$. Thus, if our break-even hypothesis is correct we should expect the vertical difference between the pair of schedules in Figure 10.5 to average out to 2.437.

```
stack_fuel2 <- stack_fuel %>%
  filter(Fuel == c("Biodiesel, $/gallon","Soyoil, cents/lb"))
ggplot(stack_fuel2,aes(y = log_price,x = week,color = Fuel)) +
  geom_line() +
  ggtitle("Biodiesel and Soybean Oil Price Comparison")
```

We can add the difference between the log of the soybean oil price and the log of the biodiesel price to the *lnfuel* tsibble and then plot this new series together with the mean of this series. Figure 10.6 shows that with the exception of the "pandemic years", the $ln(P_t^S) - ln(P_t^B)$ series is relatively stable around a mean value of approximately 2.37. This value is quite close to the 2.437 which we predicted using the break-even pricing model. This outcome suggests that a cointegration relationship between the prices of soybean oil and the biodiesel is expected.

```
lnfuel <- lnfuel %>%
  mutate(PriceDiff= lnsoy - lnbio) %>%
  mutate(MeanPriceDiff = mean(PriceDiff))

ggplot(lnfuel, aes(week)) +
  geom_line(aes(y=PriceDiff), colour="blue") +
  geom_line(aes(y=MeanPriceDiff), colour="red")
```

```
## Don't know how to automatically pick scale for object of type <ts>. Defaulting
## to continuous.
```

Formally, if a pair of time series are each *I(1)* but a linear combination of this pair is *I(0)* (i.e., stationary) then the pair is said to be cointegrated. In the

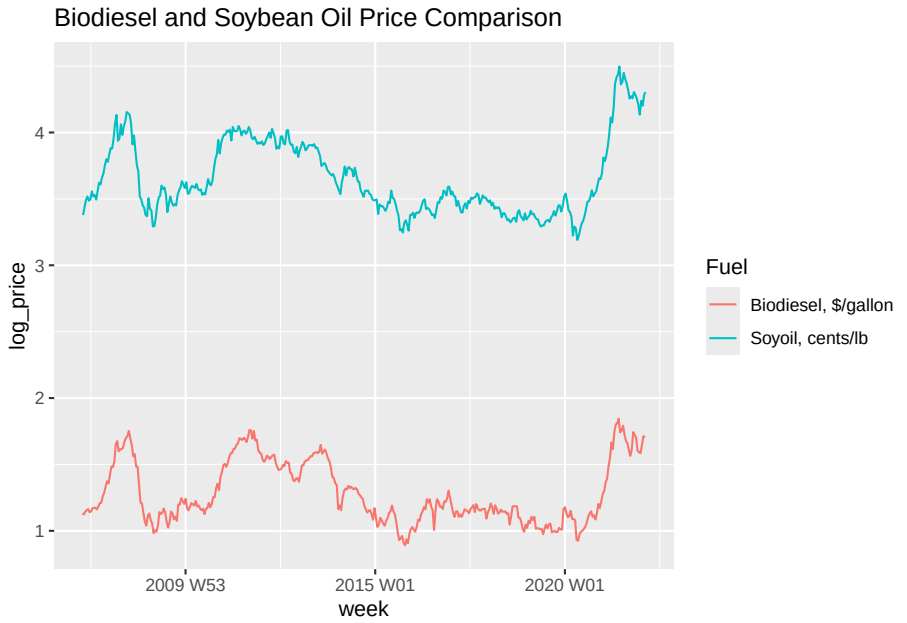

Figure 10.5: Average Weekly Logged Biodiesel and Soybean Oil Prices

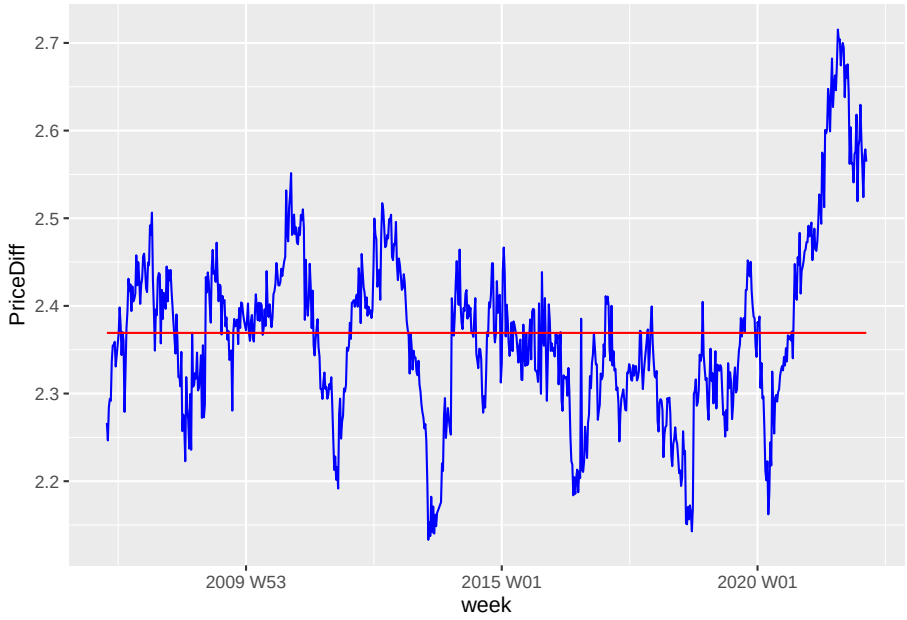

Figure 10.6: Implied Value of Beta

case of soybean oil and biodiesel prices, we believe that $P_t^S - \beta P_t^B$ is stationary. The β is the single element of the *cointegrating vector (also known as the long run parameter vector). If we add an intercept term to the equation then cointegration implies that $P_t^S - \alpha - \beta P_t^B$ is stationary and the cointegrating vector becomes $\{\alpha, \beta\}$.

There are two versions of the Engel-Granger test for cointegration. In the first test we use economic theory to obtain values for the elements of the cointegrating vector and then use an augmented Dickey-Fuller (ADF) test to test if the linear combination, $P_t^S - \alpha - \beta P_t^B$, is stationary. In our case the previous theory of break-even pricing implies $\alpha = 0$ and $\beta = 11.440$. Thus, the first test for cointegration involves using the ADF procedure to test the null hypothesis that $P_t^S - 11.440 P_t^B$ has a unit root.

The second version of the Engle-Granger test is applicable when economic theory does not provide specific values for the elements of the cointegrating vector. In this case α and β must be estimated using the $P_t^S = \alpha + \beta P_t^B$ regression equation. We can then do an ADF test on the fitted equation, $P_t^S - \hat{\alpha} - \hat{\beta} P_t^B$, where $\hat{\alpha}$ and $\hat{\beta}$ are the least squares estimates of α and β, respectively. This is equivalent to collecting the residuals from the $P_t^S = \alpha + \beta P_t^B$ regression and conducting an ADF test on these residuals.

Several comments are in order. First, the first version of the Engle-Granger test is generally more powerful than the second version, and so the first version is recommended when feasible. Second, when using the second method include a time trend in the $\hat{\alpha}$ and $\hat{\beta}$ regression equation if both time series appear to be influenced by a deterministic time trend. Third, when using the second version of the Engle-Granger method the ADF testing critical values must be adjusted. This adjustment process is discussed below.

10.3.2 Application: Soybean Oil and Biodiesel Prices

In this section we use the Engle-Granger method to test whether or not soybean oil and biodiesel prices are cointegrated. Let's begin with the first version of the Engle-Granger test which assumes that the value for the elements of the cointegrating vector are known. Recall that for the case of soybean oil and biodiesel prices, theory tells us that $P_t^S = 11.44 P_t^B$. In logs this expression is $ln(P_t^S) - 2.437 - ln(P_t^B) = 0$. If we let $z_t = ln(P_t^S) - 2.437 - ln(P_t^B)$ then testing for cointegration requires testing whether z_t has a unit root. To conduct this test we will use the ADF procedure which was described in Chapter 5 and used in the previous section.

Begin by adding z_t to the *lnfuel* tsibble and then generate the ADF test statistics and critical values.

```
lnfuel <- lnfuel %>%
  mutate(z= lnsoy - 2.437 - lnbio)
head(lnfuel)
```

```
## # A tsibble: 6 x 8 [1W]
##       week lnbio lnsoy lndies lncrude PriceDiff MeanPriceDiff       z
##     <week> <dbl> <dbl>  <dbl>   <dbl>     <dbl>          <dbl>   <dbl>
## 1 2007 W15  1.13  3.40   1.04    4.14      2.27           2.37 -0.171
## 2 2007 W16  1.13  3.38   1.06    4.14      2.25           2.37 -0.190
## 3 2007 W17  1.12  3.41   1.05    4.18      2.28           2.37 -0.153
## 4 2007 W18  1.14  3.44   1.03    4.16      2.29           2.37 -0.143
## 5 2007 W19  1.14  3.44   1.03    4.13      2.29           2.37 -0.145
## 6 2007 W20  1.16  3.49   1.02    4.15      2.34           2.37 -0.0999
```

```
engle1 <- list(
  trend = ur.df(lnfuel$z, type = "trend", selectlags = "AIC"),
  drift = ur.df(lnfuel$z, type = "drift", selectlags = "AIC")
)

engle1_trend <- summary(engle1$trend)@teststat
engle1_drift <- summary(engle1$drift)@teststat
adf_engle1 <- cbind(engle1_trend,engle1_drift)
adf_engle1
```

```
##                tau3     phi2     phi3      tau2    phi1
## statistic -3.517081 4.201503 6.203826 -3.497882 6.21612
```

```
rbind(engle1$trend@cval,engle1$drift@cval)
```

```
##       1pct  5pct 10pct
## tau3 -3.96 -3.41 -3.12
## phi2  6.09  4.68  4.03
## phi3  8.27  6.25  5.34
## tau2 -3.43 -2.86 -2.57
## phi1  6.43  4.59  3.78
```

Using the left side of the stationary testing decision chart we see that the $\phi_3 = 6.204$ test statistic is slightly below (in absolute value) the 5 percent critical value for ϕ_3, which is 6.25. After moving to the right side of the decision chart we see that the $\phi_1 = 6.216$ is greater than (in absolute value) the 5 percent critical value for ϕ_1, which is 4.59. As well, the $\tau_2 = -3.498$ test statistic is greater than (in absolute value) the 5 percent critical value for τ_2, which is -2.86. This means the unit root null hypothesis can be rejected. Accordingly, the price

of soybean oil and the price of biodiesel are necessarily cointegrated with a long run economic relationship given by $P^s = 11.44P^B$.

Let's now use the second Engle-Granger method, which requires us to conduct the cointegration test in two stages. In the first stage the long run relationship is $ln(P_t^S) = \alpha + \beta ln(P_t^B) + \gamma T_t + \epsilon_t$, where T_t is a time trend variable. In the second stage the residuals from this regression, $\hat{\epsilon}_t$, are collected and ADF test is performed on these residuals. However, "special" critical values for the ADF test statistics are required - the ones used in the first method above should not be used.

To complete the first stage of the second Engle-Granger procedure, the following code adds a time trend to the *lnfuel* tsibble, estimates the the first-stage regression equation and adds the residuals from this regression to the *lnfuel* tsibble.

```
lnfuel <- lnfuel %>%
  mutate(trend= 1:nrow(lnfuel))

engle_reg <- lm(lnsoy ~ lnbio + trend, lnfuel)

lnfuel <- lnfuel %>%
  mutate(engle_resid= engle_reg$resid)
```

For the second stage of the second Engle-Granger procedure, ADF test statistics are generated for the saved regression residuals. Residuals have mean zero and no time trend, which means that the "none" option should be used when using the *ur.df()* function.

```
engle2 <-  ur.df(lnfuel$engle_resid, type = "none", selectlags = "AIC")
summary(engle2)@teststat
```

```
##                tau1
## statistic -4.266553
```

As noted, special critical values must be used for this second method of the Engle-Granger test for cointegration. There is only one τ test statistic and one set of critical values for this test statistic because when testing residuals for stationarity it is not necessary to consider the mean and trend options.

The special critical values have been tabulated by Mackinnon (1990), https://www.researchgate.net/publication/4804830_Critical_Values_for_Cointegration_Tests. To import these values for use in R a function named *englegranger()* has been created. The first argument of this function is the number of time series variables to be included in the test: $N = \{2, 3\}$. The second argument is a 0 if there is no time trend included in the first stage

regression and a 1 if a time trend is included. The third argument is the number of observations in the sample. For the case at hand we will use *englegranger(2,1,nrow(lnfuel))* to generate the special critical values.

```
source(here("data/ch9","englegranger.R"))
englegranger(2,1,nrow(lnfuel))
```

```
##   Confidence Size Trend_TwoVar
## 1         1%  776    -4.346071
## 2         5%  776    -3.793065
## 3        10%  776    -3.505189
```

These results show that the $\tau_1 = -4.266$ test statistic from above is greater (in absolute value) than the special critical value for τ_1 (e.g., -3.793 for the 5 percent threshold). Consequently, we can reject the null hypothesis that the residuals from the $ln(P_t^S) = \alpha + \beta ln(P_t^B) + \gamma T_t + \epsilon_t$ regression do not have a unit root. The absence of a unit root implies that once again we can conclude that the prices of soybean oil and biodiesel are cointegrated.

To conclude this section note that the Engle-Granger test for cointegration can be used when there are more than two variables in the system. To do this it is necessary to choose one variable as the left-side dependent variable and the other variables will serve as right side explanatory variables. The regression residuals can then be tested for stationarity. The problem is that the testing outcome will depend on which variable serves as the dependent variable. The Johansen test, which is examined next, overcomes this problem. It should be noted that the Johansen [1991] test also works well for two-commodity systems.

10.4 Johansen Test for Cointegration

This section begins with a brief conceptual overview of the Johansen [1991] test of cointegration. The Johansen procedure is then used to test for cointegration, first with the three vegetable oils and then with a set of four commodities in the biofuel category, which we examined earlier in this chapter. Of particular interest in each case is the estimated number of cointegrated vectors. For each data set the proper set up of the test is emphasized. This includes choosing the appropriate number of lags and deciding whether to include an intercept and trend in the testing equation.

10.4.1 Conceptual Overview of Johansen Test

At a general level, the Johansen [1991] test can be viewed as a as a multivariate generalization of the augmented Dickey-Fuller test. Specifically, linear combinations of time series variables rather than the individual time series are tested

for unit roots. The relatively complex details of the Johansen procedure go beyond the scope of this chapter. The brief description of the procedure which is provided below follows Chapter 8 of Juselius [2006].

Recall from Section 10.1 that the simplified version of the econometric model in VECM format can be written as $\Delta P_t = \Pi P_{t-1} + \epsilon_t$. Further recall that the number of cointegrating vectors in the system of equations is given by the rank of Π. Eigenvalues enter the analysis because the product of the eigenvalues equal the determinant of a square matrix and the sum of the eigenvalues equal the trace of a square matrix. This explains the names of the two versions of the Johansen test: maximum eigenvalue and trace.

A matrix which has less than full rank, which is always the case for Π assuming that each variable in the system is $I(1)$, has a zero determinant and thus has at least one zero-valued eigenvalue. Testing for zero-valued eigenvalues is central in the Johansen test. The eigenvalues in question are not the eigenvalues of Π directly. Rather they are the eigenvalues of a transformation of Π. In particular, in a system consisting of n time series variables the n eigenvalues in question $(\lambda_1, \lambda_2, ..., \lambda_n)$ are given by the solution to

$$|\lambda S_{11} - S_{01} S_{00}^{-1} S_{10}| = 0 \tag{10.15}$$

See **?** for details about the various S matrices in equation (10.15). As **?** explains, the eigenvalues which are the solution to equation (10.15) can be interpreted as the squared canonical correlations between a linear combination of $\beta_i' P_{t-1}$ and $\omega_i' \Delta P_{t-1}$.

The set of eigenvalues which come from equation (10.15) have non-negative values. The value of the largest eigenvalue is key because if it has a zero value then the rank of Π is zero which in turn means there are no cointegrating vectors. If the largest eigenvalue is positive but the second largest one is zero, then we can conclude that the rank of Π is one, and there is one cointegrating vector. A system with at least one cointegrating vector is said to be cointegrated.

The following example illustrates how the testing procedure works. In Section 10.1 we assumed that the price of gasoline and the price of corn were simple random walks, and the price of ethanol was a weighted average of the gasoline and corn prices. Let's assume the weights are $(0.5, 0.5)$. As well, assume e_t^g, e_t^c and e_t^e are all drawn from a standard normal distribution. With these assumptions we can simulate our system for 50 periods as follows.

```
set.seed(5)
p_g <- rep(NA,50)
p_c <- rep(NA,50)
p_e <- rep(NA,50)
p_g[1] <- 0
p_c[1] <- 0
```

```
p_e[1] <- 0

for (i in 2:50) {
 p_g[i] <- p_g[i-1]+rnorm(1)
 p_c[i] <- p_c[i-1]+rnorm(1)
 p_e[i] <- 0.5*p_g[i]+0.5*p_c[i]+rnorm(1)
}

x <- cbind(p_g,p_c,p_e)
```

This system of equations will have three eigenvalues which correspond to equation (10.15) above (λ_1, λ_2 and λ_3). The code from the *ca.jo()* function can be used to calculate the values for these eigenvalues as follows:

```
lambda
```

```
## [1] 0.47134845 0.12477176 0.02093495
```

For both versions of the Johansen test the null hypothesis is that the largest eigenvalue in the set of three (i.e., 0.4713) is not statistically different than zero. Based on our discussion in Section 10.1 we expect that for this particular system there is one cointegrating vector. Thus, we expect that the largest eigenvalue (i.e., 0.4713) is statistically different from zero.

There are two versions of the Johansen test. The difference between the maximum eigenvalue version and the trace version is how the alternative hypothesis is set up. The first test involves testing whether the largest eigenvalue is zero relative to the alternative hypothesis, which is that the next largest eigenvalue is zero. For the trace test the null hypothesis is $rank(\Pi) = r_0$ and the alternative hypothesis is $r_0 < rank(\Pi) \leq n$ where n is the maximum number of possible cointegrating vectors. If the null is rejected, then testing increments forward by one. For our particular example where we are expecting one cointegrating vector, we expect that after rejecting the first null hypothesis we will not be able to reject the second null hypothesis which is that the second largest eigenvalue (i.e., 0.1248) is equal to zero.

The set of eigenvalues are random variable with a specific distribution. **?** shows how the likelihood ratio (LR) test statistic which is used for the trace version of the test is constructed from the estimated set of eigenvalues. Johansen [1991] constructed the set of critical values which are used to assess the test statistics. In the next section the trace version of the Johansen test is used to estimate the number of cointegrating vectors in the vegetable oil and biofuels data sets.

10.4.2 Johansen Test for Three vegetable oil Prices

This section is devoted to testing the three vegetable oil prices for cointegration. Testing usually requires both enhancing the structure of the simple $\Delta y_t = \Pi y_{t-1} + e_t$ vector error correction model (VECM) and reducing autocorrelation by adding lags in ΔP_t. Most commodity price series trend up over time which means that the VECM model requires an intercept term and occasionally a deterministic trend and seasonal dummies. Purely exogenous variables can also be added.

The *ca.jo()* function can be used for both conducting the Johansen test for cointegration and estimating the VECM. In both cases most of the parameter choices are self explanatory. In addition to choosing the number of lags to add, which is measured by k, the user must choose the *type* to be either "eigen" or "trace"), choose the *ecdet* to be either"none", "const" or "trend" and choose *season* to correspond to the frequency of the data (e.g., $season = 4$ to add quarterly dummies). What is less obvious is choosing *spec* to be either "longrun" or "transitory" and choosing what to enter for *dumvar* (the default is *NULL*). The choice between "longrun" or "transitory" is relevant when estimating the VECM model but both choices lead to the same outcome for the Johansen test of cointegration.

To select the lag length (k parameter) to use for the Johansen test we use the data in levels to estimate a VAR model (see previous chapter for a refresher). The optimal value for k in the Johansen test is equal to the optimal number of lags from the VAR model minus one. We subtract one because the VECM which underlies the Johansen test is estimated in first differences whereas the VAR is estimated in levels. Note that the estimated coefficients from the VAR estimated in levels are likely to be spurious because the data set which is being tested for cointegration is non-stationary. Nevertheless, the estimates of the optimal number of lags to include to eliminate autocorrelation is still acceptable. It is always a good idea to use a Portmanteau tests for autocorrelation (or equivalent) to ensure that the recommended number of lags has eliminated the autocorrelation.

In some cases it is obvious whether or not to include an intercept or trend, both when estimating the VAR for the purpose of choosing k and when conducting the Johansen test of cointegration. If in doubt it is a good idea to generate results for all three model specifications and then choose the specification which best fits the model. The model fit selection criteria is discussed below.

The *VARselect()* function from R's *vars* package is used to estimate k for the alternative versions of the VAR (i.e., intercept, trend, etc.). Because we are using monthly data we will set a maximum of 12 lags.

```
lag_const <- VARselect(lnveg[,2:4], lag.max = 12, type = "const")[["selection"]]
lag_trend <- VARselect(lnveg[,2:4], lag.max = 12, type = "trend")[["selection"]]
```

```
lag_none <- VARselect(lnveg[,2:4], lag.max = 12, type = "none")[["selection"]]
cbind(lag_const,lag_trend,lag_none)
```

```
##        lag_const lag_trend lag_none
## AIC(n)         3         3        3
## HQ(n)          2         2        2
## SC(n)          2         2        2
## FPE(n)         3         3        3
```

The *VARselect()* function returns the number of lags which minimizes various information criterion for the estimated model. For VAR analysis it is common to use either the Akaike information criterion (AIC) in the first row of the above table or the Schwartz criterion (SC), which is also known as the Bayesian information criteria (BIC), in the third row. For each of the three VAR structures, the AIC method recommends two lags and the BIC method recommends three lags. We will go with the conservative choice and use three lags for each of the three VAR specifications.

The next step is to estimate the three different VAR specifications and compare the log likelihood outcomes. The specification with the highest log likelihood is the optimal specification for this particular data set.

```
LL_const <- logLik(VAR(lnveg[,2:4],type="const",lag.max=3))
LL_trend <- logLik(VAR(lnveg[,2:4],type="trend",lag.max=3))
LL_none <- logLik(VAR(lnveg[,2:4],type="none",lag.max=3))
LL <- data.frame(LL_const,LL_trend,LL_none)
row.names(LL)[1] <- "Log Likelihood"
LL
```

```
##                LL_const LL_trend  LL_none
## Log Likelihood 1236.838  1232.17 1231.432
```

The above table shows that the log likelihood function is highest with the "const" specification of the VAR. This means we will also use the "const" specification when conducting Johansen test. With $k = 3$ optimal for the VAR, it follows from the above discussion that $k = 2$ is optimal for the Johansen test. It is useful to note that $k = 2$ is the minimum allowable value when using the *ca.jo()* function for implementing the Johansen test.

As previously noted we will use the trace version of the Johansen test. The null hypothesis we are most interested in is no cointegration, which simultaneously implies a rank of $r = 0$ for the Π matrix in the estimated VECM and a value of zero for the largest eigenvalue from equation (10.15). If we reject the $r = 0$ null then we conclude the three prices are cointegrated. However, we will continue to test in order to isolate the number of cointegrating vectors. Specifically, subject

to rejecting the $r = 0$ null we will test the null that Π has rank $r = 1$, and if this null is rejected we will move on to test the $r = 2$ null.

The function which implements the Johansen test is as follows:

ca.jo(x, type = c("eigen", "trace"), ecdet = c("none", "const", "trend"), K = 2, spec=c("longrun", "transitory"), season = NULL, dumvar = NULL)

Given the above discussion and results we want *type = "trace"*, *ecdat = "const"* and $k = 2$. As noted, *spec* can be set to either "longrun" or "transitory" because the outcome of the Johansen hypothesis test is the same with both options.

The test statistics and the critical values for the Johansen trace test are as follows:

```
jotest <- ca.jo(lnveg[,2:4], type = "trace", ecdet = "const", K = 2, spec = "longrun")
cbind(summary(jotest)@teststat, summary(jotest)@cval)
```

```
##                     10pct  5pct  1pct
## r <= 2 |  6.408816  7.52  9.24 12.97
## r <= 1 | 22.068626 17.85 19.96 24.60
## r = 0  | 59.800459 32.00 34.91 41.07
```

The bottom row of the previous table corresponds to the $(r = 0)$ null hypothesis of no cointegration. The likelihood ratio (LR) test statistic of 59.800 is larger than all three critical values which means we can reject the null hypothesis and conclude that there is cointegration corresponding to at least one cointegrating vector.

We now move to the middle of the previous table of results and then test the null hypothesis that $r = \{0, 1\}$. The test statistic of 22.068 is greater than the 5 percent critical value and so we reject the null that $r = \{0, 1\}$. We can now conclude that there is cointegration corresponding to at least two cointegrating vectors.

Finally, we move to the top row of the previous table and test the null hypothesis that $r = \{0, 1, 2\}$. The test statistic of 6.408 is less than all critical values and so we fail to reject the null that $r = \{0, 1, 2\}$. We can now conclude that there is cointegration with two integrating vectors. We would not expect this last hypothesis to be rejected because accepting $r = 3$ would imply that each individual price is stationary, which we know from the ADF tests earlier in this chapter that this is not the case.

Let's interpret our findings in light of the theoretical cointegration results from Section 10.3. A likely scenario is that the prices of palm oil and soybean oil have one long term relationship, and the prices of rapeseed oil and soybean oil have a second long term relationship. The first relationship emerges because palm oil and soybean oil are strong substitutes on the demand side. The second

relationship emerges because of the substitution potential on both the demand and the supply side. One could work out the technical coefficients of the pricing relationships by determining differences in energy content and discounts or premiums for quality. Alternatively, these long term pricing relationships can be estimated with a VECM. The details of VECM estimation is the topic of the next chapter.

10.4.3 Food - Energy Price Linkage

Of interest in this section is the extent that food prices and energy prices are cointegrated. This is an important question due to its implications for global food affordability and food price volatility. As discussed in Chapter 1, biofuels are an important driver of the the proposed linkage between food and energy prices. In section we will use the biofuel data set to both test for cointegration and estimate the number of cointegrating vectors.

As previously noted the proposed test is not possible with the current data set because the price of crude oil is $I(0)$ rather than $I(1)$. An important restriction when testing for cointegration is that all variables included in the system must be integrated of the same order. To address this problem let's replace the price of crude oil with the price of palm oil, which means that the final data set includes the prices of diesel, biodiesel, soybean oil and palm oil. The original objective of testing if food and energy prices are cointegrated remains intact because soybean oil and palm oil are both used to produce biodiesel, which itself is a strong substitute for regular diesel.

Substituting the price of palm oil for the price of crude oil requires merging the *lnfuel* and *lnveg* data sets. The problem is the former is a measure of weekly prices and the latter is a measure of monthly prices. This means that the weekly prices must be aggregated into monthly prices before the merge can happen. The common dates for the two data sets are April, 2007 to December, 2020.

```
lnveg <- lnveg %>% filter_index("2007-04" ~ "2020-12")

lnfuel <-lnfuel %>%
  index_by(year_month = ~ yearmonth(.)) %>%
  summarise(
    lnsoy_m = mean(lnsoy),
    lnbio_m = mean(lnbio),
    lndies_m = mean(lndies)
  ) %>%
  filter_index("2007-04" ~ "2020-12") %>%
  mutate(lnpalm_m = lnveg$lnpo)
```

Similar to the procedure used in previous section, the first step for setting up the

Johansen test for cointegration is to use the *VARselect()* function to estimate k, which is the optimal number of lags to include in the *ca.jo()* function.

```
lag_const <- VARselect(lnfuel[,2:5], lag.max = 12, type = "const")[["selection"]]
lag_trend <- VARselect(lnfuel[,2:5], lag.max = 12, type = "trend")[["selection"]]
lag_none <- VARselect(lnfuel[,2:5], lag.max = 12, type = "none")[["selection"]]
cbind(lag_const,lag_trend,lag_none)
```

```
##        lag_const lag_trend lag_none
## AIC(n)         2         2        2
## HQ(n)          2         2        2
## SC(n)          2         2        2
## FPE(n)         2         2        2
```

The previous table shows that for all three specifications of the VAR model the optimal number of lags is two, regardless of which information criterion is used. This means that the optimal number of lags for the cointegration test is one since the underlying VECM requires one less lag as compared to the VAR. However, the minimum value of k in the Johansen test is two, and so $k = 2$ is used in the analysis to follow.

The next step is to determine whether to include and intercept and trend when calling the *ca.jo()* function. Once again following the procedure from the previous section, the specification which maximizes the log likelihood of the estimated VAR is deemed optimal.

```
LL_const <- logLik(VAR(lnfuel[,2:5],type="const",lag.max=12))
LL_trend <- logLik(VAR(lnfuel[,2:5],type="trend",lag.max=12))
LL_none <- logLik(VAR(lnfuel[,2:5],type="none",lag.max=12))
LL <- data.frame(LL_const,LL_trend,LL_none)
row.names(LL)[1] <- "Log Likelihood"
LL
```

```
##                LL_const LL_trend  LL_none
## Log Likelihood 1231.019 1222.801 1220.447
```

The above table shows that the log of the likelihood function is at a maximum for the "const" specification of the VAR function. This means we can move forward with the Johansen test using two lags and the "const" specification.

```
jotest2 <- ca.jo(lnfuel[,2:5], type = "trace", ecdet = "const", K = 2, spec = "longrun")
cbind(summary(jotest2)@teststat, summary(jotest2)@cval)
```

```
##                   10pct  5pct  1pct
```

```
## r <= 3 |  8.348196  7.52  9.24 12.97
## r <= 2 | 19.215439 17.85 19.96 24.60
## r <= 1 | 41.376025 32.00 34.91 41.07
## r = 0  | 66.054397 49.65 53.12 60.16
```

As in the previous section, the Johansen hypothesis test begins with the $r = 0$ null in the bottom row of the previous table. The 66.054 test statistic is well above the three critical values and so the $r = 0$ null hypothesis can be rejected. This confirms that the two food prices (palm oil and soybean oil) and the two energy prices (biodiesel and diesel) are cointegrated with at least one cointegrating vector. Similarly, the $r = \{0, 1\}$ null in the second row from the bottom can be rejected based on the 41.376 test statistic. Using the 19.215 test statistic, the $r = \{0, 1, 2\}$ null hypothesis cannot be rejected (at the 5 percent threshold). Consequently, we conclude that the pair of food prices and the pair of energy prices are cointegrated with two cointegrating vectors.

To interpret these results recall that in the four commodity price model which we constructed in Section 10.3 there was also two cointegrating vectors due to price blending. Perhaps with this particular data there is just one blending pricing relationship (the price of biodiesel is a blend of the prices of regular diesel and soybean oil) and also one pair-wise pricing relationship (the price of soybean oil and palm oil are substitutes in consumption). The first blend price relationship is of particular interest because it establishes a long term pricing relationship between the prices of major energy and food commodities.

Let's repeat the cointegration test but with the price of palm oil excluded. If the previous theory is correct then there should be just one statistically significant cointegrating vector. We will continue to assume *type* = "trace", *ecdet* = "const" and $K = 2$.

```
jotest2 <- ca.jo(lnfuel[,2:4], type = "trace", ecdet = "const", K = 2, spec = "longrun"
cbind(summary(jotest2)@teststat, summary(jotest2)@cval)
```

```
##                    10pct  5pct  1pct
## r <= 2 |  7.860773  7.52  9.24 12.97
## r <= 1 | 18.896181 17.85 19.96 24.60
## r = 0  | 42.948737 32.00 34.91 41.07
```

The results above show that the null hypothesis of one or fewer cointegrating vectors cannot be rejected at the 95 level. Thus, as we predicted, there is just one cointegrating vector when palm oil is removed from the system. It is likely that the one long term pricing relationship is that price of biodiesel is a blend of the price of regular diesel and soybean oil. Similar to the previous case there is evidence of a long term pricing linkage between the price of energy and the price of food.

10.5 Conclusions

This chapter on cointegration and the chapter to follow on the vector error correction model (VECM) marks an important branching of the time series econometrics literature. As note in Chapter 1, time series econometrics is an important tool for price forecasting. The reduced form VAR which we featured in the previous chapter is well suited for forecasting because when forecasting there is little value in distinguishing between long term pricing relationships and short term deviations in the price from these long term relationships. Economists who are interested in policy analysis are typically much more interested in long term pricing relationships and the speed of adjustment back to equilibrium when prices are shocked. For these economists testing for cointegration and later estimating VECM models are key for analyzing policy. Chapter 1 provided a good overview of applications of cointegration and the VECM for the analysis of prices in the food and energy sectors.

In this chapter blend pricing was an important reason for cointegration. In vertical marketing systems we will also see additive pricing relationships. For example, in the wheat and flour price example, suppose the cost of processing the wheat into flour was a fixed at m dollars per tonne of wheat. In this case cointegration is expected to reflect the difference between the price of wheat and the price of flour rather than a price ratio. Similarly, it is natural to assume that spot prices and futures prices for food and energy commodities are cointegrated with the long term price difference equal to the long term basis. Finally, commodity prices at spatially distinct locations are expected to be cointegrated with regional transportation costs defining the parameters in the cointegration vector.

An important consideration of testing for cointegration which has not been discussed in this chapter is the implications of structural breaks in the cointegration relationship. One can imagine changes in the long term pricing relationship between the price of corn and the price of ethanol with changes in ethanol pricing policy. Failure to account for these structural breaks when testing for cointegration may result in incorrectly assuming there is no cointegrating vector when in fact one does exist if we allow the vector to have a structural break. The analysis of structural breaks in Chapter 7 provides a framework for incorporating structural breaks into the Engle-Granger test for cointegration.

Gregory and Hansen [1996] allow for a structural break in either the intercept alone or the entire coefficient vector when testing for cointegration with ADF (i.e., Engel-Granger) method. Simulations are used to show that the power of the ADF test for detecting cointegration falls sharply if the simulated data has structural breaks but there is no allowance for these breaks in the ADF test. Only in a limited number of experiments is an ADF test successful in identifying the structural break when the break data is chosen through a grid search procedure, similar to that used in Chapter 7. In the literature since the publication of Gregory and Hansen [1996] there has been limited applications

of cointegration tests which accommodate structural breaks.

10.6 Appendix

10.6.1 Test for Difference Stationary: Vegetable Oil Prices

In this section we test to ensure that the first difference of the logged vegetable oil prices are stationary. Establishing stationarity in this case will ensure that these prices are integrated of order 1. We begin by adding the first difference of the logged prices to the *lnveg* tsibble dataframe.

```
lnveg <- lnveg %>%
  mutate(D.lnpo=difference(lnpo, 1,1),
  D.lnso=difference(lnso, 1,1),
  D.lnro=difference(lnro, 1,1))
head(lnveg)
```

```
## # A tsibble: 6 x 7 [1M]
##      month  lnpo  lnso  lnro  D.lnpo  D.lnso  D.lnro
##      <mth> <dbl> <dbl> <dbl>   <dbl>   <dbl>   <dbl>
## 1 2007 Apr  6.61  6.62  6.70 NA      NA      NA
## 2 2007 May  6.70  6.67  6.74  0.0932  0.0454  0.0439
## 3 2007 Jun  6.73  6.73  6.77  0.0207  0.0577  0.0268
## 4 2007 Jul  6.75  6.79  6.83  0.0253  0.0599  0.0648
## 5 2007 Aug  6.72  6.83  6.87 -0.0283  0.0383  0.0345
## 6 2007 Sep  6.75  6.88  6.95  0.0249  0.0587  0.0864
```

Using the same procedure as in the previous section, the ADF tests for the first difference of logged vegetable prices can be conducted.

```
D.adfpo <- list(
  trend = ur.df(lnveg$D.lnpo[-1], type = "trend", selectlags = "AIC"),
  drift = ur.df(lnveg$D.lnpo[-1], type = "drift", selectlags = "AIC")
)

D.adfso <- list(
  trend = ur.df(lnveg$D.lnso[-1], type = "trend", selectlags = "AIC"),
  drift = ur.df(lnveg$D.lnso[-1], type = "drift", selectlags = "AIC")
)

D.adfro <- list(
  trend = ur.df(lnveg$D.lnro[-1], type = "trend", selectlags = "AIC"),
  drift = ur.df(lnveg$D.lnro[-1], type = "drift", selectlags = "AIC")
)
```

Now generate the ADF test statistics

```
D.potrend <- summary(D.adfpo$trend)@teststat
D.podrift <- summary(D.adfpo$drift)@teststat
D.adf_po <- cbind(D.potrend,D.podrift)

D.sotrend <- summary(D.adfso$trend)@teststat
D.sodrift <- summary(D.adfso$drift)@teststat
D.adf_so <- cbind(D.sotrend,D.sodrift)

D.rotrend <- summary(D.adfro$trend)@teststat
D.rodrift <- summary(D.adfro$drift)@teststat
D.adf_ro <- cbind(D.rotrend,D.rodrift)

D.adf_veg <- rbind(D.adf_po,D.adf_so,D.adf_ro)
rownames(D.adf_veg)  <- c("Diff Palm Oil", "Diff Soybean Oil", "Diff Rapeseed Oil")
D.adf_veg
```

```
##                       tau3     phi2     phi3      tau2     phi1
## Diff Palm Oil     -7.486974 18.73623 28.09810 -7.504203 28.16281
## Diff Soybean Oil  -6.229578 12.99747 19.49621 -6.262118 19.60706
## Diff Rapeseed Oil -6.579738 14.48002 21.71689 -6.610093 21.84982
```

Finally, generate the set of ADF critical value for the difference in the logged palm oil price and note that this set of values is relevant for all tests in this section.

```
rbind(D.adfpo$trend@cval,D.adfpo$drift@cval)
```

```
##       1pct  5pct 10pct
## tau3 -3.99 -3.43 -3.13
## phi2  6.22  4.75  4.07
## phi3  8.43  6.49  5.47
## tau2 -3.46 -2.88 -2.57
## phi1  6.52  4.63  3.81
```

With the data in first differences we can start on the right side of the decision chart because any deterministic trend will have been eliminated through first differencing. A comparison of the ϕ_1 test statistics to its critical value of ϕ_1 reveals that the $a_0 = \delta = 0$ null hypothesis can easily be rejected for all three vegetable oils. The decision chart tells us to use the τ_2 test statistic to test the $\delta = 0$ null hypothesis. Once again the null can easily be rejected for all three vegetable oils and so we conclude there is no unit root in any of the three cases. The decision chart tells us to use a regular t test to determine if there is a

positive intercept. However, this step is not necessary because it is sufficient to show that the differenced price series does not have a unit root and is therefore difference stationary.

10.6.2 Biofuel ADF Test for a Unit Root

In this section we will conduct the ADF tests for the log of the four bio-energy prices.

```
biofuel <- readRDS(here("data/ch9", "biofuel.RDS"))

lnfuel <- biofuel %>% mutate(lnbio = log(biodiesel),lnsoy=log(soyoil), lndies = log(die
  dplyr::select(week,lnbio,lnsoy,lndies, lncrude)


adfbio <- list(
  trend = ur.df(lnfuel$lnbio, type = "trend", selectlags = "AIC"),
  drift = ur.df(lnfuel$lnbio, type = "drift", selectlags = "AIC")
)

adfsoy <- list(
  trend = ur.df(lnfuel$lnsoy, type = "trend", selectlags = "AIC"),
  drift = ur.df(lnfuel$lnsoy, type = "drift", selectlags = "AIC")
)

adfdies <- list(
  trend = ur.df(lnfuel$lndies, type = "trend", selectlags = "AIC"),
  drift = ur.df(lnfuel$lndies, type = "drift", selectlags = "AIC")
)

adfcrude <- list(
  trend = ur.df(lnfuel$lncrude, type = "trend", selectlags = "AIC"),
  drift = ur.df(lnfuel$lncrude, type = "drift", selectlags = "AIC")
)
```

Now generate the ϕ_i and τ_i test statistics.

```
biotrend <- summary(adfbio$trend)@teststat
biodrift <- summary(adfbio$drift)@teststat
adf_bio <- cbind(biotrend,biodrift)

soytrend <- summary(adfsoy$trend)@teststat
soydrift <- summary(adfsoy$drift)@teststat
adf_soy <- cbind(soytrend,soydrift)
```

```
diestrend <- summary(adfdies$trend)@teststat
diesdrift <- summary(adfdies$drift)@teststat
adf_dies <- cbind(diestrend,diesdrift)

crudetrend <- summary(adfcrude$trend)@teststat
crudedrift <- summary(adfcrude$drift)@teststat
adf_crude <- cbind(crudetrend,crudedrift)

adf_fuel <- rbind(adf_bio,adf_soy,adf_dies, adf_crude)
rownames(adf_fuel)  <- c("Biodiesel", "Soybean Oil", "Diesel", "Crude Oil")
adf_fuel
```

```
##                  tau3      phi2      phi3      tau2     phi1
## Biodiesel   -1.370949 0.9014377 1.1154824 -1.458175 1.300085
## Soybean Oil -1.048233 0.8044669 0.7669672 -1.125340 1.073346
## Diesel      -2.008875 1.4615347 2.1188729 -2.059844 2.195002
## Crude Oil   -3.502844 4.1119210 6.1536042 -3.135445 4.929757
```

Recall that the critical values are the same for the ADF tests for all four bio-energy prices. It is therefore adequate to report the critical values for the first of the four ADF tests.

```
rbind(adfbio$trend@cval,adfbio$drift@cval)
```

```
##       1pct  5pct 10pct
## tau3 -3.96 -3.41 -3.12
## phi2  6.09  4.68  4.03
## phi3  8.27  6.25  5.34
## tau2 -3.43 -2.86 -2.57
## phi1  6.43  4.59  3.78
```

Following the ADF procedure from the previous section we begin at the top left of the decision chart. Here we see that for the first three bio-energy prices (biodiesel, soybean oil and diesel fuel) the absolute value of the ϕ_3 and τ_3 test statistics are less than their respective absolute critical values. We then move to the top right of the decision chart and note that for these three commodities the absolute value of the ϕ_1 test statistics are less in absolute value than the ϕ_1 critical value. We can therefore conclude that biodiesel, soybean oil and diesel fuel are each non-stationary due to a unit root.

In contrast, the absolute value of ϕ_3 and τ_3 are greater than their respective critical values for the case of crude oil. The left side of the decision chart tells us crude oil prices are stationary. The p value for the estimated coefficient for the time trend variable is not significant (this result is not shown) and so we can

conclude that crude oil is stationary without a determinist trend. The outcome that crude oil prices are stationary is surprising because this outcome was not evident in the graph of crude oil prices, which was shown above.

10.6.3 Test for Difference Stationary: Biofuel Prices

In this section we conduct an ADF test on the first difference of the log of biodiesel, soybean oil and diesel prices in order to determine if these prices are difference stationary. Crude oil is excluded from this exercise because it was established in the previous section that the log of crude oil prices is stationary.

We begin by adding the first difference of the logged prices to the **lnfuel** tsibble dataframe.

```
lnfuel <- lnfuel %>%
  mutate(D.lnbio=difference(lnbio, 1,1),
  D.lnsoy=difference(lnsoy, 1,1),
  D.lndies=difference(lndies, 1,1))
head(lnfuel)
```

```
## # A tsibble: 6 x 8 [1W]
##       week lnbio lnsoy lndies lncrude  D.lnbio  D.lnsoy D.lndies
##     <week> <dbl> <dbl>  <dbl>   <dbl>    <dbl>    <dbl>    <dbl>
## 1 2007 W15  1.13  3.40   1.04    4.14 NA       NA       NA
## 2 2007 W16  1.13  3.38   1.06    4.14  0       -0.0199   0.0129
## 3 2007 W17  1.12  3.41   1.05    4.18 -0.00810  0.0296  -0.00908
## 4 2007 W18  1.14  3.44   1.03    4.16  0.0209   0.0307  -0.0141
## 5 2007 W19  1.14  3.44   1.03    4.13  0       -0.00225 -0.00678
## 6 2007 W20  1.16  3.49   1.02    4.15  0.0111   0.0565  -0.00683
```

Using the same procedure as in the previous section, the ADF tests for the first difference of logged vegetable prices can be conducted.

```
D.adfbio <- list(
  trend = ur.df(lnfuel$D.lnbio[-1], type = "trend", selectlags = "AIC"),
  drift = ur.df(lnfuel$D.lnbio[-1], type = "drift", selectlags = "AIC")
)

D.adfsoy <- list(
  trend = ur.df(lnfuel$D.lnsoy[-1], type = "trend", selectlags = "AIC"),
  drift = ur.df(lnfuel$D.lnsoy[-1], type = "drift", selectlags = "AIC")
)

D.adfdies <- list(
```

```
  trend = ur.df(lnfuel$D.lndies[-1], type = "trend", selectlags = "AIC"),
  drift = ur.df(lnfuel$D.lndies[-1], type = "drift", selectlags = "AIC")
)
```

Now generate the ADF test statistics.

```
D.biotrend <- summary(D.adfbio$trend)@teststat
D.biodrift <- summary(D.adfbio$drift)@teststat
D.adf_bio <- cbind(D.biotrend,D.biodrift)

D.soytrend <- summary(D.adfsoy$trend)@teststat
D.soydrift <- summary(D.adfsoy$drift)@teststat
D.adf_soy <- cbind(D.soytrend,D.soydrift)

D.diestrend <- summary(D.adfdies$trend)@teststat
D.diesdrift <- summary(D.adfdies$drift)@teststat
D.adf_dies <- cbind(D.diestrend,D.diesdrift)

D.adf_fuel <- rbind(D.adf_bio,D.adf_soy,D.adf_dies)
rownames(D.adf_fuel)  <- c("Diff Biodiesel", "Diff Soybean Oil", "Diff Diesel")
D.adf_fuel
```

```
##                       tau3      phi2      phi3      tau2      phi1
## Diff Biodiesel   -19.66287 128.87729 193.31500 -19.66366 193.33069
## Diff Soybean Oil -18.62754 115.66340 173.49509 -18.62154 173.38093
## Diff Diesel      -11.65009  45.24627  67.86486 -11.65184  67.88721
```

Finally, generate the set of ADF critical value for the difference in the logged palm oil price and note that this set of values is relevant for all tests in this section.

```
rbind(D.adfbio$trend@cval,D.adfbio$drift@cval)
```

```
##       1pct  5pct 10pct
## tau3 -3.96 -3.41 -3.12
## phi2  6.09  4.68  4.03
## phi3  8.27  6.25  5.34
## tau2 -3.43 -2.86 -2.57
## phi1  6.43  4.59  3.78
```

Conducting the ADF test according to the decision chart procedure reveals that the first difference of the log of the biodiesel, soybean oil and diesel prices are stationary. Thus, we can conclude that biodiesel, soybean oil and diesel prices are difference stationary. This is equivalent to assuming that these prices are integrated of order one.

Chapter 11

Vector Error Correction Model

This chapter builds on the theory of cointegration by introducing the two-variable error correction model (ECM) and the more general n-variable vector error correction model (VECM). The ECM and VECM provide an alternative to estimating a VAR in first differences when the data set is $I(1)$ non-stationary and there exists one or more cointegrating vectors. In the last chapter we learned that the cointegrating vectors are stationary linear combinations of the prices in the data set. As well, the cointegrating vectors reflect the long run relationships, which for the case of commodities is usually implied by the law of one price (LOP). A ECM is expected to be appropriate for estimating the margin in vertical markets such as flour milling because the wholesale price of flour and the farm price of wheat are typically cointegrated. Similarly, a VECM is expected to be appropriate for estimating the relationship among export prices for a globally traded commodity such as palm oil because the set of export prices at different locations are typically cointegrated.

The ECM and the VECM allow us to estimate these long run pricing relationships but at the same time identify short run pricing dynamics. A key component of pricing dynamics is the speed at which prices adjust back to their long run relationship after a pricing shock. This so-called error correction is implied by a single coefficient in the case of the ECM and a vector of coefficients in the case of the VECM. It is this identification of the long run pricing relationship and the error correction which distinguishes the VECM from a VAR estimated in first differences. In fact, for data which is $I(1)$ and cointegrated estimating a VAR in first differences rather than estimating a VECM results in less efficient coefficient estimates with higher standard errors.

This chapter has three empirical applications. In the first application an error correction model (ECM) is estimated using pricing data on soybean oil and

biodiesel. The estimation of this vertical marketing model is an extension of the Engle-Granger two-step method of testing for cointegration. In the second empirical application the estimated VECM for the Canada - U.S. crude oil market is recovered from the Johansen test of cointegration in this market. Here we show that Canadian and U.S. oil are very close to perfect substitutes with relatively rapid error correction. In the last case study a standard ECM is modified by incorporating seasonal differencing. The modified model is used to examine the long and short run relationship between the U.S. consumer price index (CPI) for fruits and vegetables and the price of diesel fuel, with an index of drought in California serving as an exogenous variable. The results show that higher diesel fuel prices and more severe drought conditions result in a higher fruit and vegetable CPI in both the short and long run.

The next section is used to provide more background to the error correction model (ECM) and vector error correction model (VECM). In Section 11.2 the ECM is constructed and the Engle-Granger two step method for its estimation is applied to the soybean oil and biodiesel case study. Section 11.3 provides the theory of the more general VECM. Here we will revisit the Johansen test for cointegration and see that the estimated VECM is implied by the Johansen testing results. In particular, the *cajo()* and *cajorls()* functions together provide the full set of estimated VECM coefficients. In Section 11.4 a VECM is estimated as part of the Canada - U.S. crude oil case study and in Section 11.5 a modified ECM is estimated as part of the fruit and vegetable CPI case study. A summary and concluding remarks are provided in Section 11.6.

11.1 Background

To provide real world context, suppose a researcher who is not familiar with the concept of cointegration and the ECM wished to model the wheat - flour marketing margin. The researcher hypothesized that the price of wheat is a random walk with drift, $P_t^w = a_w + P_{t-1}^w + e_t^w$ and the price of flour is a linear function of the price of wheat, $P_t^f = m + bP_t^w + e_t^f$. The researcher is particularly interested in the fixed component of the marketing margin which is measured by the m coefficient in the previous equation.

The researcher either knows or can verify by testing that both prices are $I(1)$ non-stationary. To avoid a spurious regression, the researcher regresses the first difference of the price of flour on the first difference of the price of wheat. This first difference model implies that $\Delta P_t^f = ba_w + be_t^w + \Delta e_t^f$. In this regression the estimated intercept, ba_w, does not contain the marketing margin coefficient, m. This coefficient has been eliminated through the first differencing of the pair of prices. This example illustrates that important information about long term pricing relationships is typically lost if prices are estimated in first differences rather than in levels.

If the researcher recognized that the price of flour and the price of wheat are

cointegrated then the regression equation in levels, $P_t^f = \gamma_0 + \gamma_1 P_t^w + e_t^f$, can be preserved and the estimated γ_0 coefficient is the desired fixed component of the marketing margin (i.e., m). The cointegration ensures that the important problem of spurious regression is no longer a threat but a secondary problem remains. Cointegration does not mean that a regular regression of the form $P_t^f = \gamma_0 + \gamma_1 P_t^w + e_t^f$ is the best method of estimating the pricing relationship. Rather, an error correction model (ECM) should be used to estimate the long run pricing relationship as measured by the γ_0 and γ_1 coefficients while allowing for on-going error correction. Failing to incorporate the error correction component into the econometric model will typically result in less efficient estimates of the long run pricing coefficients.

Recall from the previous chapter that with the Engle-Granger method of testing for cointegration the first stage requires estimating the long run pricing equation, $P_t^f = \gamma_0 + \gamma_1 P_t^w + e_t^f$, and the second stage requires testing the residuals from this first stage regression for stationarity. In this current chapter the Engle-Granger method uses the lagged values of these residuals to estimate the error correction model (ECM). The lagged residuals are called the error correction term (ECT) and the estimated coefficient on the ECT, which is denoted λ, is a measure of the speed of pricing adjustment in response to a shock. Similarly, recall from the previous chapter that with the Johansen method of testing for cointegration the set of cointegrating vectors are contained in β where $\alpha\beta' = \Pi$ and Π is the estimated matrix of coefficients from the VECM, written as $\Delta Y_t = \Pi Y_{t-1} + \Phi \Delta Y_{t-1} + e_t$. Later in this chapter we show that the α matrix is the multi-equation equivalent of the λ coefficient in the Engle-Granger method.

We will use the *URCA* and *vars* package for estimating the VECM. This pair of packages does not have a function for directly estimating the ECM with the two-step Engle-Granger method. Rather, the researcher must manually collect the residuals from the Engle-Granger test for cointegration and then use these residuals to estimate the ECM. The *VECM()* from the *tsDyn* package does have an option to directly estimate the Engle-Granger ECM for those wishing to avoid the manual programming step as described above. It should be noted that the *vars* package has the very useful *vec2var()* function, which is used to transform an estimated VECM back into its VAR representation. This allows the VECM to indirectly forecast, conduct Granger causality testing and conduct impulse response analysis.

If the pricing system has more than two variables then the Johansen method of estimating the VECM should always be used instead of the Engle-Granger method. If the system has two variables then either method can be used but the Johansen method is recommended, especially if there is no clear consensus on which of the two variables belongs on the left side of the regression equation in the Engle-Granger procedure. Another important advantage of the Johansen method over the Engle-Granger method in a two-variable system is that pretesting of the variables to ensure both are $I(1)$ is necessary with the Engle-Granger method whereas no pretesting is required with the Johansen method.

Engle and Granger [1987] used the cointegration framework proposed by **?** to show that a cointegrated system has an error correction representation. This important result is now known as the Granger representation theorem. Johansen [1991] built on the Engle and Granger [1987] model by constructing a procedure for testing for cointegration in a multi-equation system. **?** provided a detailed description of the vector error correction model (VECM) and its connection to the Johansen [1991] test for cointegration. Recall that a brief review of applications of the VECM for food and energy prices was provided in Chapter 1.

11.2 Error Correction Model

In this section we convert a two equation vector autoregression (VAR) model into a two equation error correction model (ECM). The ECM is then estimated using the soybean oil - biodiesel pricing data from the previous chapter. Previous testing with the Engle-Granger two-step method established that the price of soybean oil and the price of biodiesel are cointegrated and are therefore suitable for ECM estimation.

11.2.1 ECM Theory

The two equation VAR for modelling the price of soybean oil (P_t^s) and the price of biodiesel (p_t^b) can be expressed as

$$\begin{bmatrix} 1 & -a_{12} \\ a_{21} & -a_{21} \end{bmatrix} \begin{bmatrix} P_t^s \\ P_t^b \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} + \begin{bmatrix} b_{11} & -b_{12} \\ b_{21} & -b_{21} \end{bmatrix} \begin{bmatrix} P_{t-1}^s \\ P_{t-1}^b \end{bmatrix} + \begin{bmatrix} e_t^s \\ e_t^b \end{bmatrix} \quad (11.1)$$

The first expression can be written as follows

$$P_t^s = c_1 + a_{12}P_t^b + b_{11}P_{t-1}^s - b_{12}P_{t-1}^b + e_t^s \quad (11.2)$$

Now subtract P_{t-1}^s from both sides of equation (11.2), add $a_{12}P_{t-1}^s - a_{12}P_{t-1}^s$ to the right side, and then combine terms.

$$P_t^s - P_{t-1}^s = c_1 + a_{12}\left(P_t^b - P_{t-1}^b\right) - \left[\left(1 - b_{11}\right)P_{t-1}^s - \left(a_{12} - b_{12}\right)P_{t-1}^b\right] + e_t^c \quad (11.3)$$

The final step is to create a new parameter α and then add $(1 - b_{11})\,\alpha$ to the right side of equation (11.3). This allows equation (11.3) to be rewritten as

$$\Delta P_t^s = c + a_{12}\Delta P_t^b - \lambda\left(P_{t-1}^s - \alpha - \beta P_{t-1}^b\right) + e_t^c \quad (11.4)$$

where: $c = c_1 - (1 - b_{11})\,\alpha$, $\lambda = 1 - b_{11}$ and $\beta = \frac{a_{12} - b_{12}}{1 - b_{11}}$.

The error correction model (ECM) which is given by equation (11.4) has two important properties. The first is that all variables in the ECM are stationarity and the second is that the ECM has an error correction component. Given that the two prices are $I(1)$ it follows that ΔP_t^s and ΔP_t^b are both stationary. What remains is to show that the error correction term (ECT) which is given by $P_{t-1}^s - \alpha - \beta P_{t-1}^b$ is also stationary. The assumption that P_t^s and P_t^b are cointegrated implies that there exists a linear combination of this price pair which is stationary. Thus, there exists values for α and β which ensure $P_{t-1}^s - \alpha - \beta P_{t-1}^b$ is stationary. It now follows that even though the individual prices are non-stationary all variables within the ECM are stationary and can therefore be estimated without resulting in a spurious outcome.

The error correction coefficient in equation (11.4) is given by λ. To understand why this is coefficient enters the equation with a negative sign suppose the prices are in long term equilibrium so that $P_{t-1}^s = \alpha + \beta P_{t-1}^b$. Now, if either $P_{t_1}^s$ is shocked upward or P_{t-1}^b shocked downward in period $t-1$ then the price pair is no longer in equilibrium and $P_{t-1}^s = \alpha + \beta P_{t-1}^b$ will take on a positive value. Assuming $\lambda > 0$, equation (11.4) shows that a positive value for $P_{t-1}^s = \alpha + \beta P_{t-1}^b$ implies a negative value for ΔP_t^s and thus a decrease in P_t^s from period t to period $t+1$. This is the error correction because when transitioning from period t to period $t+1$ the pair of prices are pushed back toward their long run equilibrium values. The size of λ, which is expected to have a value between zero and one, is a measure of the speed of adjustment in the pair of prices back toward the long run equilibrium.

11.2.2 ECM for Soybean Oil and Biodiesel

In this section the ECM model which is given by equation (11.4) is estimated using the two-step Engle-Granger method. The first stage of this procedure is identical to the first stage of the Engle-Granger method for testing for cointegration, which was examined in the previous chapter. Specifically, P_t^s is regressed on P_t^b and the set of residuals, $\hat{e}_t^s$, are collected. These residuals are the difference between the actual and fitted values for P_t^f. Thus, in period $t-1$ we have $e_{t-1}^s = P_{t-1}^s - \alpha - \beta P_{t-1}^b$. This expression allows equation (11.4) to be rewritten as

$$\Delta P_t^s = c + a_{12}\Delta P_t^b - \lambda \hat{e}_{t-1}^f + e_t^c \tag{11.5}$$

The second stage of the ECM estimation procedure is to estimate equation (11.5) in order to obtain an estimate of the speed of adjustment parameter λ. In other words, the second stage involves regressing ΔP_t^s on ΔP_t^b and the error correction term (ECT), which is the collection of lagged residuals from the first stage regression.

Let's begin with the first stage regression for the soybean oil – biodiesel price data which was used at the beginning of the previous chapter. Both prices are in logs and a time trend, T_t, is added to the first stage regression. Thus, the model to be estimated is

$$ln(P_t^S) = \alpha + \beta ln(P_t^B) + \gamma T_t + \epsilon_t$$

The first step is to load the required packages and then read in the *biofuel.RDS* file which contains the data.

```
pacman::p_load(tidyverse, readxl, lubridate, here, janitor, tsibble, feasts, gridExtra

biofuel <- readRDS(here("data/ch10", "biofuel.RDS"))

lnfuel <- biofuel %>% mutate(lnbio = log(biodiesel),
                             lnsoy=log(soyoil),
                            trend= 1:nrow(biofuel))  %>%
  dplyr::select(week,lnbio,lnsoy,trend)
head(lnfuel)
```

```
## # A tsibble: 6 x 4 [1W]
##       week lnbio lnsoy trend
##     <week> <dbl> <dbl> <int>
## 1 2007 W15  1.13  3.40     1
## 2 2007 W16  1.13  3.38     2
## 3 2007 W17  1.12  3.41     3
## 4 2007 W18  1.14  3.44     4
## 5 2007 W19  1.14  3.44     5
## 6 2007 W20  1.16  3.49     6
```

The $ln(P_t^S) = \beta_0 + \beta_1 ln(P_t^B) + \gamma T_t + \epsilon_t$ equation can now be estimated and the results of this first stage regression are discussed below.

```
biomod <- lm(lnsoy ~ lnbio + trend,data=lnfuel )
biomod$coefficients
```

```
##  (Intercept)        lnbio        trend
## 2.135573e+00 1.163013e+00 6.264737e-05
```

To create the second stage regression add the negative of the lagged residuals from the first stage regression to our *lnfuel* data frame along with the first differences of the soyoil and biodiesel prices.

```
lnfuel <- lnfuel %>% mutate(
  resid = biomod$resid,
  L.resid = -lag(resid),
  D.lnsoy = difference(lnsoy),
  D.lnbio = difference(lnbio)) %>%
slice(-1)

head(lnfuel)
```

```
## # A tsibble: 6 x 8 [1W]
##       week lnbio lnsoy trend   resid L.resid  D.lnsoy  D.lnbio
##     <week> <dbl> <dbl> <int>   <dbl>   <dbl>    <dbl>    <dbl>
## 1 2007 W16  1.13  3.38     2 -0.0736  0.0536 -0.0199   0
## 2 2007 W17  1.12  3.41     3 -0.0347  0.0736  0.0296  -0.00810
## 3 2007 W18  1.14  3.44     4 -0.0284  0.0347  0.0307   0.0209
## 4 2007 W19  1.14  3.44     5 -0.0307  0.0284 -0.00225  0
## 5 2007 W20  1.16  3.49     6  0.0128  0.0307  0.0565   0.0111
## 6 2007 W21  1.16  3.51     7  0.0306 -0.0128  0.0178   0
```

In a typical Engle-Granger application lagged values of ΔP_t^S are also added to the regression to control for autocorrelation. We will skip this step and revisit this issue when we estimate the more general vector error correction model (VECM) in the analysis below. We can estimate the second stage error correction model (ECM) as follows:

```
ecm <- lm(D.lnsoy ~ D.lnbio + L.resid,data = lnfuel)
summary(ecm)
```

```
##
## Call:
## lm(formula = D.lnsoy ~ D.lnbio + L.resid, data = lnfuel)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.10758 -0.01822 -0.00082  0.01614  0.10783
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.0007506  0.0010835   0.693   0.4887
## D.lnbio     0.5296276  0.0329185  16.089   <2e-16 ***
## L.resid     0.0292995  0.0120248   2.437   0.0151 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.03015 on 772 degrees of freedom
## Multiple R-squared:  0.2522, Adjusted R-squared:  0.2502
## F-statistic: 130.2 on 2 and 772 DF,  p-value: < 2.2e-16
```

The results from the two stages or the ECM regression tell us three things:

- The estimated long run (law of one price) relationship between the price of soybean oil and the price of biodiesel is given by $P_t^S = 2.136 + 1.163 P_t^B + 0.000062 T_t$. This equation reflects the technical/conversion and long run economic relationship when soybean oil is converted to biodiesel.
- A one unit increase in the price of biodiesel causes a 0.530 short run increase in the price of soybean oil. This coefficient on "D.lnbio" reflects the extent by which a shock in the biodiesel market spills over into the soybean oil market the following period.
- The speed of adjustment coefficient, which theoretically varies between 0 and 1, is estimated to equal 0.029 and is statistically significant at the 90 percent level. The small value for this coefficient implies very slow price adjustment to regain long run equilibrium. One possibility is that market arbitrage in the soybean oil – biodiesel market is not very efficient. A more likely explanation is that if the prices of soybean oil and biodiesel were included in a larger model then additional long run pricing relationships would emerge, and the error correction for these additional long run relationships would be much stronger than the current 0.029 error correction.

11.3 Vector Error Correction Model (VECM)

The Wikipedia page for the error correction model (ECM) [https://en.wikipedia.org/wiki/Error_correction_model] identifies a number of problems with the two-stage ECM estimation method. Most of these problems are eliminated if a single stage vector error correction model (VECM) is estimated instead. When modelling commodity prices systems of three or more equations are typically required to capture the full set of relevant long term pricing relationships (e.g., crude oil, biodiesel and soybean oil). The ECM is designed for two-equation systems whereas the VECM method allows n-equation systems to be efficiently estimated. Moreover, with the VECM there are no left and right hand side variables and so unlike the ECM model the ordering of the variables is not a relevant issue. A third important advantage of the VECM is that pretesting to establish that all variables are $I(1)$ is not necessary because a positive cointegration test outcome automatically implies that the variables within the model are $I(1)$.

11.3.1 VAR to VECM Transformation

The derivation of the VECM is similar to the derivation of the ECM except now the construction uses vectors and matrices rather than individual variables. The analysis begins by showing how the VECM can be derived from the VAR. It then shows that the VECM estimates can be decomposed into one or more cointegrating vectors and corresponding speed of adjustment vectors. For this theory section we will derive the VECM from a VAR with three variables and two lags. The results are easily generalizable to m variables and n lags.

The VAR in question can be expressed as as

$$Y_t = C + \Phi_1 Y_{t-1} + \Phi_2 Y_{t-2} + \epsilon_t \tag{11.6}$$

The following identity matrix is required to create the VECM.

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Let $\Delta Y_t = Y_t - Y_{t-1}$. Subtract Y_{t-1} from both sides of the VAR to obtain

$$\Delta Y_t = C + [\Phi_1 - I]\, Y_{t-1} + \Phi_2 Y_{t-2} + \epsilon_t$$

Now subtract and add $[\Phi_1 - I]\, Y_{t-2}$ on the right side of the previous equation to obtain:

$$\Delta Y_t = C + [\Phi_1 - I]\, Y_{t-1} - [\Phi_1 - I]\, Y_{t-2} + [\Phi_2 + \Phi_1 - I]\, Y_{t-2} + \epsilon_t \tag{11.7}$$

Rewrite equation (11.7) as

$$\Delta Y_t = C + [\Phi_1 - I]\, \Delta Y_{t-1} + [\Phi_2 + \Phi_1 - I]\, Y_{t-2} + \epsilon_t \tag{11.8}$$

If the brackets are opened and the terms recombined, equation (11.8) can be rewritten as

$$\Delta Y_t = C + [\Phi_1 + \Phi_2 - I]\, Y_{t-1} - \Phi_2 \Delta Y_{t-1} + \epsilon \tag{11.9}$$

Equation (11.9) is the VECM representation of the VAR model. The ΔY_t and ΔY_{t-1} vectors are stationary because by the assumption all variables within Y_t are $I(1)$. The $[\Phi_1 + \Phi_2 - I]\, Y_{t-1}$ component of equation (11.9) represents linear combinations of the individual y_t variables within Y_t. If the system is cointegrated then $[\Phi_1 + \Phi_2 - I]\, Y_{t-1}$ has less than full rank, which in turn implies stationarity for $[\Phi_1 + \Phi_2 - I]\, Y_{t-1}$ and thus for the VECM as a whole.

When analyzing the Johansen test for cointegration in the previous chapter we noted that $\alpha\beta' = \Phi_1 + \Phi_2 - I$ where β contains the sets of cointegrating vectors and α contains the error correction (speed of adjustment) coefficients. Keep in mind that there is one error correction coefficient for each variable in the system. For example, suppose the system consists of three prices and there are two cointegrating vectors, which can be expressed as $P_t^1 + \beta_1^1 P_t^2 + \beta_2^1 P_t^3$ and $P_t^1 + \beta_1^2 P_t^2 + \beta_2^2 P_t^3$. We require a speed of adjustment coefficient, λ_1^1, which is a measure of how P_t^1 responds when prices move away from $P_t^1 + \beta_1^1 P_t^2 + \beta_2^1 P_t^3$. Similarly, we need λ_1^2, which is a measure of how P_t^1 responds when prices move away from $P_t^1 + \beta_1^2 P_t^2 + \beta_2^2 P_t^3$. For P_t^2 we require λ_2^1 and λ_2^2, and for P_t^3 we require λ_3^1 and λ_3^2. The complete set consists of six speed of adjustment coefficients.

Suppose we use the *cajo()* function to test a system of four equations for cointegration. Suppose also testing reveals that there are two cointegrating vectors. Based on the theory we would expect the β matrix to have two columns which correspond to the pair of cointegrating vectors. Similarly, we would expect the α matrix to have two columns which correspond to the pair of the pair of cointegrating vectors. Instead, *cajo()* returns β and α as four by four matrices. In this case the left two columns of α and β contain the desired information, and the right two columns should be ignored. If a combination of the *cajo()* and *cajorts()* functions are used then the output from *cajorts()* reports only the statistically significant cointegrating vectors and error correction coefficients.

11.3.2 Simulated Prices

Before estimating a VECM with real-world data it is useful to estimate a VECM with simulated data since this will allow us to have control over the coefficients in the cointegration and error correction matrices. To keep things simple we will simulate a system with three prices, one cointegrating vector and no constant or trend term. This means that in the absence of cointegration the three prices are each a random walk without drift. We will specify the coefficients which define the VECM and then convert the model into a VAR for the purpose of simulation.

Let's begin by constructing a matrix, Φ, which defines the short fun dynamics of the VAR. Specifically, if ΔP_t is the vector of price differences then $\Delta P_t = \Phi \Delta P_{t-1}$ defines the short run dynamics. The parameters of Φ must ensure that ΔP_t is stationary. The following parameters meet this stationarity restriction.

```
Phi <- matrix(c(0.45,-0.25,-0.125,-0.4,0.50,-0.4,-0.125,-0.25,0.55),nrow=3)
Phi
```

```
##        [,1] [,2]   [,3]
## [1,]  0.450 -0.4 -0.125
## [2,] -0.250  0.5 -0.250
## [3,] -0.125 -0.4  0.550
```

Let I denote a three by three identify matrix. This allows the short run dynamics equation to be expanded and a expression for the P_t vector to be written as

$$P_t = (I + \Phi) P_{t-1} - \Phi P_{t-2} + \epsilon_t \tag{11.10}$$

Equation (11.10) is the VAR expressed in price levels. These level prices are individually non-stationary because we moved from modeling an $I(0)$ to an $I(1)$ random price vector. To simulate these prices we assume that $P_1 = P_2 = 0$. We further assume that ϵ_t is a vector of three independent normally distributed random variables with mean 0 and standard deviation $sd = 0.015$ We can create the P_t vector in equation (11.10) as follows:

```
set.seed(20)
sd <- 0.015
n <- 500

P1 <- rep(NA,n)
P2 <- rep(NA,n)
P3 <- rep(NA,n)
P1[1:2] <- 0
P2[1:2] <- 0
P3[1:3] <- 0

rn1 <- rep(NA,n)
rn2 <- rep(NA,n)
rn3 <- rep(NA,n)
rn1[1:1] <- 0
rn2[1:1] <- 0
rn3[1:1] <- 0

I <- diag(x=1,nrow=3)
A <- I + Phi

for (t in 3:n){
  rn1[t]=rnorm(1,sd=sd)
  rn2[t]=rnorm(1,sd=sd)
  rn3[t]=rnorm(1,sd=sd)
  P1[t] <- A[1,1]*P1[t-1]+A[1,2]*P2[t-1]+A[1,3]*P3[t-1]-Phi[1,1]*P1[t-2]-Phi[1,2]*P2[t-2]-Phi[1,3
  P2[t] <- A[2,1]*P1[t-1]+A[2,2]*P2[t-1]+A[2,3]*P3[t-1]-Phi[2,1]*P1[t-2]-Phi[2,2]*P2[t-2]-Phi[2,3
  P3[t] <- A[3,1]*P1[t-1]+A[3,2]*P2[t-1]+A[3,3]*P3[t-1]-Phi[3,1]*P1[t-2]-Phi[3,2]*P2[t-2]-Phi[3,3
}

Price <- cbind(P1,P2,P3)
Price2 <- as_tibble(Price) %>% mutate(period = 1:n)
```

Figure 11.1 below shows the path over 500 periods for the three random walk prices within P_t. It should be clear from Figure 11.1 that the three prices are individually non stationary. For this reason we will not take the time to use an ADF test to confirm that each price has a unit root.

```
ggplot(Price2, aes(period)) +
  geom_line(aes(y = P1), color = "black") +
  geom_line(aes(y = P2), color = "red") +
  geom_line(aes(y = P3), color = "green")
```

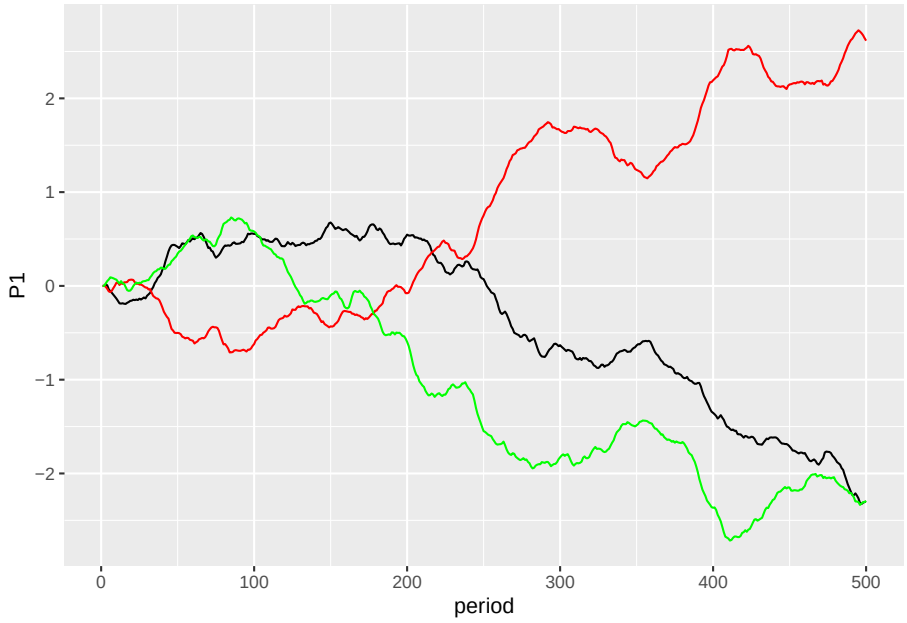

Figure 11.1: Simulated Prices for Random Walk VAR

The next step is the most important because it requires us to specify the coefficients in the cointegrating vector and the error correction vector. To do this it is useful to write the expressions for both the VECM and the VAR representation of the VECM. Note that a VECM is typically viewed as an alternative representation of a VAR but it is equally valid to view a VAR as an alternative representation of a VECM.

The respective expressions for the VECM and VAR can be written as follows.

$$\Delta P_t = \Pi P_{t-1} + \Phi \Delta P_{t-1} + \epsilon_t \tag{11.11}$$

$$P_t = (I + \Pi + \Phi) P_{t-1} - \Phi P_{t-2} + \epsilon_t \tag{11.12}$$

In equation (11.11) the Π matrix is critical because it is here that we attach values to the cointegration and error correction vectors. The cointegrating vector is given by β where $\beta_1 P_t^1 + \beta_2 P_t^2 + \beta_3 P_t^3$ is assumed to be stationary. Let's set $\beta = \{1, -1, 1\}$, which implies $P_t^2 = P_t^1 + P_t^3$ in the long run (remember that $\beta_1 = 1$ is used to normalize the vector).

Recall from the previous chapter that $\Pi = \alpha\beta'$. Consequently, the next step is to specify values for the error correction vector, α, and the cointegrating vector, β. In the α vector, α_1 is the speed that P_t^1 adjusts when the set of prices deviate from $\beta_1 P_t^1 + \beta_2 P_t^2 + \beta_3 P_t^3$. This value must be negative because a positive value for $\beta_1 P_t^1 + \beta_2 P_t^2 + \beta_3 P_t^3$ requires a reduction in P_t^1 to move back toward the long run equilibrium. Given the cointegrating vector $\beta = \{1, -1, 1\}$ it follows that α_3 must also have a negative value whereas α_2 must have a positive value. Let's set $\alpha_2 = 0.05$ and $\alpha_3 = -0.075$. These values were arbitrarily selected. A wide range of values for these three coefficients will give simulation outcomes with the same general properties.

With the β and α vectors now specified, the $\Pi = \alpha\beta'$ matrix can calculated and the values for Π can then be used to simulate prices within equation (11.10).

```
alpha <- c(-0.025,0.05,-0.075)
beta <- c(1,-1,1)
Pi <- alpha%*%t(beta)
A <- I + Pi + Phi
for (t in 3:n){
  P1[t] <- A[1,1]*P1[t-1]+A[1,2]*P2[t-1]+A[1,3]*P3[t-1]-Phi[1,1]*P1[t-2]-Phi[1,2]*P2[t-2]-Phi[1,3
  P2[t] <- A[2,1]*P1[t-1]+A[2,2]*P2[t-1]+A[2,3]*P3[t-1]-Phi[2,1]*P1[t-2]-Phi[2,2]*P2[t-2]-Phi[2,3
  P3[t] <- A[3,1]*P1[t-1]+A[3,2]*P2[t-1]+A[3,3]*P3[t-1]-Phi[3,1]*P1[t-2]-Phi[3,2]*P2[t-2]-Phi[3,3

}

Price <- cbind(P1,P2,P3)
Price2 <- as_tibble(Price) %>% mutate(period = 1:n)
```

Let's re-graph the prices so we can identify how incorporating the cointegration restriction changed the relative pricing paths. Figure 11.2 shows the expected result. The individual price series are still non-stationary but the pricing paths have shifted to conform to the $P_t^2 = P_t^1 + P_t^3$ long run pricing restriction. Specifically, the P_t^2 price path tends to hover around zero, and the P_t^1 and P_t^3 price paths have offsetting signs to ensure $P_t^2 = P_t^1 + P_t^3$. A comparison of Figures 11.1 and 11.2 reveal that this revised set of prices appear to be cointegrated.

```
ggplot(Price2, aes(period)) +
  geom_line(aes(y = P1), color = "black") +
  geom_line(aes(y = P2), color = "red") +
  geom_line(aes(y = P3), color = "green")
```

Figure 11.2: Simulated Prices for Cointegrated VAR

11.3.3 VECM Estimation

We can now use the simulated prices to estimate the VECM. We begin by using the *ca.jo()* function from the *urca* package to test for cointegration using the Johansen procedure. The test results give both the cointegrating vectors which are statistically significant, and the corresponding error correction vectors. The *cajorls()* function uses the test results from *ca.jo()* to recover all of estimated VECM coefficients.

The simulated data was generated with two lags for the VAR. The value for k in the *ca.jo()* function is the number of lags in the VAR representation of the VECM rather than the number of lags in the VECM. This means we should set $k = 2$ for the VECM estimation with the *ca.jo()* function. There is no intercept or trend in the simulated data and so we will use "none" for the *ecdet* parameter. Finally, set "trace" for *type* and "longrun" for *spec*. Note that setting *type* to "trace" rather than *eigenvalue* does not affect the estimated outcome of the VECM coefficients.

```
jotest <- ca.jo(Price, type = "trace", ecdet = "none", K = 2, spec = "longrun")
summary(jotest)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: trace statistic , with linear trend
##
## Eigenvalues (lambda):
## [1] 0.3688507464 0.0114620391 0.0008300116
##
## Values of teststatistic and critical values of test:
##
##            test 10pct  5pct  1pct
## r <= 2 |   0.41  6.50  8.18 11.65
## r <= 1 |   6.15 15.66 17.95 23.52
## r = 0  | 235.34 28.71 31.52 37.22
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##            P1.l2    P2.l2     P3.l2
## P1.l2  1.0000000 1.000000 1.0000000
## P2.l2 -1.1405754 1.860916 0.4507613
## P3.l2  0.9694834 1.264906 0.1991036
##
## Weights W:
## (This is the loading matrix)
##
##            P1.l2        P2.l2         P3.l2
## P1.d -0.02308948 -0.001093697  0.0015931123
## P2.d  0.04332358 -0.007255891  0.0000732538
## P3.d -0.07180432 -0.004061542 -0.0005672761
```

This output from the *ca.jo()* function confirms that the simulated data has one cointegrating vector. Indeed, as the top table shows, the null hypothesis of zero rank (i.e., $r = 0$) can be rejected but the null hypothesis of $r \leq 1$ cannot be rejected. Knowing that there is just one cointegrating vector we restrict our attention to the left column of the middle table. Here we see that the estimated cointegrating vector is given by $\{1, -1.119, 0.968\}$. These values are reasonably close to the true values which we assumed for the cointegration vector: $\{1, -1, 1\}$.

The left column of the bottom table of the previous *ca.jo()* output contain the three α error correction terms. Notice that this matrix is labelled "Weights" and described as the "loading matrix". The "weights" label is appropriate because the α values determine the relative size of the Π matrix in equation (11.12). The size of Π relative to $I + \Phi$ determines the extent that the non-stationary pricing paths in Figure 11.2 conform to the long run equilibrium condition.

The estimated values in the left column of the weight matrix are given by $\alpha = \{-0.022, 0.0431, -0.0723\}$. These estimates are reasonably close to the true values which were assumed for the simulation: $\alpha = \{-0.025, 0.050, -0.075\}$. These weights are quite small since theoretically their values lie anywhere within $(-1, 1)$.

The remaining coefficients for the VECM are in the Φ matrix. These coefficients reflect the short run pricing dynamics which are given by $\Delta P_t = \Phi_1 \Delta P_{t-1}$. The Φ coefficients can be recovered by using the output of the *ca.jo*() function as input into the *cajorls*() function. In addition it is necessary to specify a value for r, which is the number of cointegrating vectors. Let's show the estimated VECM only for the first pricing equation, which can be accomplished using *reg.number=1*

```
vecm_sim <-cajorls(jotest, r = 1, reg.number = 1)
vecm_sim$beta
```

```
##             ect1
## P1.12  1.0000000
## P2.12 -1.1405754
## P3.12  0.9694834
```

```
summary(vecm_sim$rlm)
```

```
##
## Call:
## lm(formula = substitute(form1), data = data.mat)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.042619 -0.011113  0.000209  0.010177  0.042186
##
## Coefficients:
##            Estimate Std. Error t value Pr(>|t|)
## ect1     -0.0230895  0.0049813  -4.635 4.57e-06 ***
## constant -0.0007801  0.0006788  -1.149 0.251062
## P1.dl1    0.4366057  0.0371106  11.765  < 2e-16 ***
## P2.dl1   -0.3389749  0.0385921  -8.784  < 2e-16 ***
```

```
## P3.dl1   -0.1103294  0.0315636  -3.495 0.000516 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01509 on 493 degrees of freedom
## Multiple R-squared:  0.4278, Adjusted R-squared:  0.422
## F-statistic: 73.71 on 5 and 493 DF,  p-value: < 2.2e-16
```

In addition to repeating the coefficients from the cointegrating vector and the error correcting vector the above output now shows the coefficients in the top row of the Φ matrix. For example, the coefficient labeled *p1.dl1* and equal to 0.4366 is a measure of how P_t^1 responds to a change in P_{t-1}^1. Similarly, the coefficient labeled *p2.dl1* and equal to -0.3389 is a measure of how P_t^1 responds to a change in P_{t-1}^2.

The VAR representation of the estimated VECM can also be recovered using the *vec2var()* function from the *vars* package. Converting an estimated VECM into its VAR representation is often useful because after the conversion the standard VAR operations such as forecasting and impulse response can be conducted. For example, if the goal was to forecast vegetable oil prices, the first step is to estimate a VECM, the second step is to use *vec2var()* to transform the estimated VECM into a VAR and the third step is to use the *predict()* function for the *var* object to do the price forecasting.

11.4 Canada - U.S. Crude Oil Case Study

Canada is a net exporter of crude oil to the U.S. The oil produced in western Canada is known as "Western Canadian Select", or *wcs* for short. The oil produced in the U.S. is known as "West Texas Intermediate", or *wti* for short. These two types of oil are highly substitutable in the production of refined petroleum products such as gasoline. Consequently, the spatial law of one price tells us that the Canadian price quoted in U.S. dollars should equal the U.S. price adjusted for quality differences and the cost of transporting the crude oil from Canada to the U.S.. The majority of oil which Canada exports to the U.S. moves by a series of pipelines. Pipeline constraints imply significant fluctuations over time in the discount which Canadian crude receives relative to U.S. crude.

Economic theory tells us that a stronger U.S. dollar will result in lower oil price for U.S. oil exporters because international buyers will turn to lower cost suppliers. For this reason we expect a negative relationship between a U.S. dollar index and the price of U.S. oil. The Canadian and U.S. price of oil and the U.S. dollar index are endogenous variables, which means that causality runs in all directions between the three variables. For this reason we expect at least one cointegrating vector for the pair of prices and the U.S. dollar index. In this section we will test this data for cointegration and if we can establish this

property then we will estimate a VECM and analyze the results. Of particular interest is the speed by which the Canadian price of oil adjusts in response to a shock in the U.S. price of oil or the U.S. dollar index.

11.4.1 Price Visualization

Let's read in the oil price and U.S. dollar index from the stored *RDS* file and then prepare the three series for plotting.

```
oil <- readRDS(here("data/ch10","oil.RDS"))
head(oil)
```

```
## # A tsibble: 6 x 4 [1M]
##      month   wti   wcs dollar
##      <mth> <dbl> <dbl>  <dbl>
## 1 2006 Jan  65.5  38.8  100
## 2 2006 Feb  61.9  28.7  100.
## 3 2006 Mar  63.0  36.8  100.
## 4 2006 Apr  70.2  52.0   99.7
## 5 2006 May  71.0  57.3   97.5
## 6 2006 Jun  71.0  51.6   98.7
```

```
oil_long <- melt(oil, id = "month", measure = c("wti", "wcs"))
```

Figure 11.3 below shows that the pair of oil prices track each other fairly closely. Nevertheless, there is still considerable fluctuations in the price discount for Canadian oil. The discount was particularly large in 2013 - 2014. As previously noted, the discount varies over time because of supply and demand shocks combined with pipeline capacity constraints.

```
ggplot(oil_long, aes(month, value, colour = variable)) + geom_line() + xlab("Date") + y
```

Let's look at the correlogram for the Canadian price of crude oil. Figure 11.4 reveals a slowly declining plot of the autocorrelation function, which is a classic symptom of non-stationary data. It is very likely that both price series are $I(1)$ and cointegrated. When using the Johansen test for cointegration there is no need to pretest the data. If the test outcome shows more than zero and less than three cointegrating vectors then this confirms our assessment that the pair of prices are $I(1)$ variables.

```
oil %>%
  ACF(wcs) %>%
  autoplot()
```

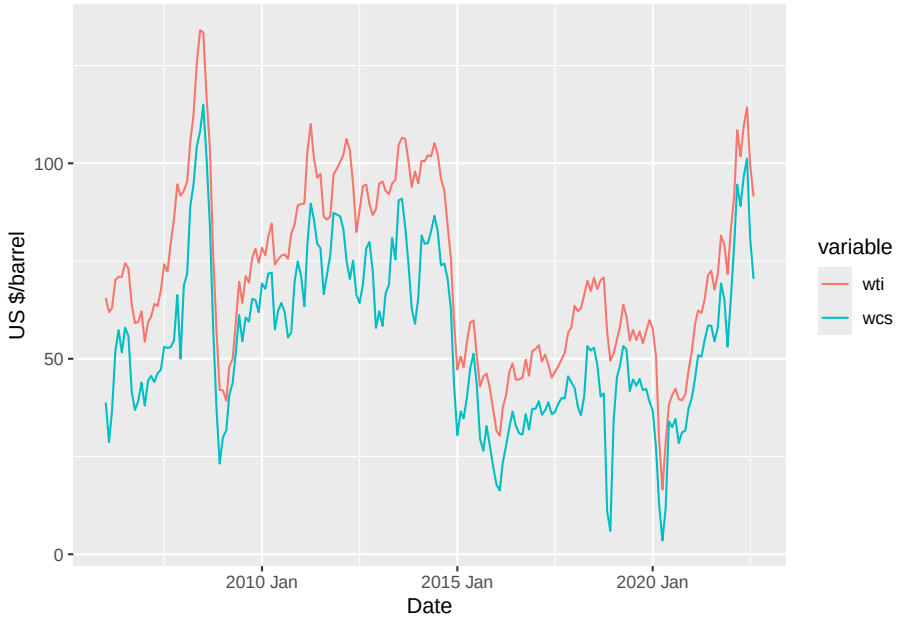

Figure 11.3: Monthly Price of Canadian (WCS) and U.S. (WTI) Crude Oil: 2005 - 2022

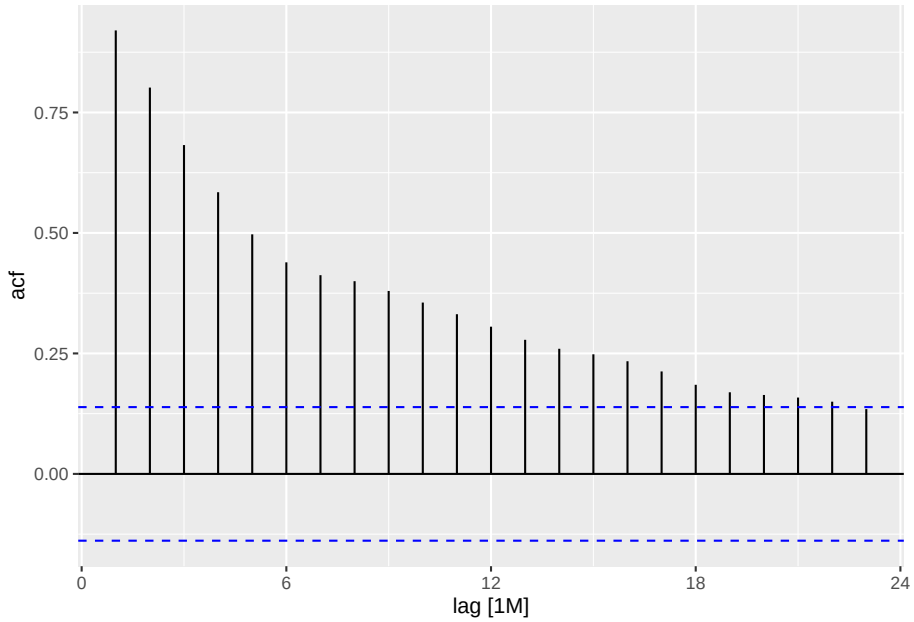

Figure 11.4: Correlogram for Canadian (WCS) Crude Oil: 2005 - 2022

Figure 11.5 is a plot of the monthly average U.S. dollar index. Notice the strong strengthening of the index after 2015. As previously noted, economic theory tells us we should expect a negative relationship and thus cointegration between the U.S. dollar index and the U.S., price of oil. The blog post https://seekingalpha.com/article/4205756-dollar-drive-oil-prices provides a good discussion about the reason for this negative relationship.

```
index_plot <- oil %>% autoplot(dollar) +
  labs(y = "U.S. Dollar Index", x="Date")
index_plot
```

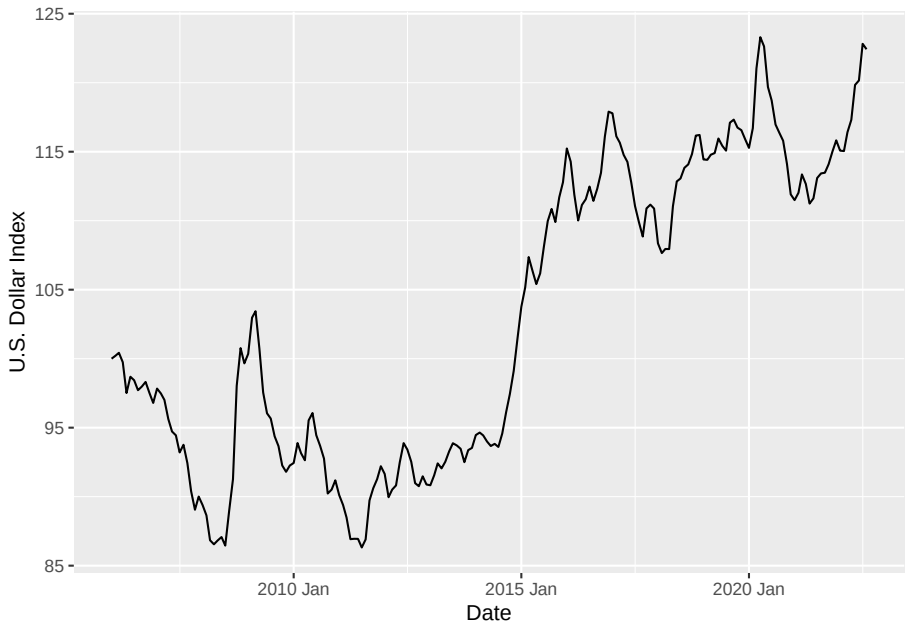

Figure 11.5: Monthly U.S. dollar index: 2006 - 2022

How should we think about the presence of a second cointegrating vector? Suppose we believe that in the long run

$$\begin{aligned} P_t^C &= -Z + \beta P_t^U + \gamma_C D_t + e_t^C \\ P_t^U &= \alpha + \gamma_U D_t + e_t^U \end{aligned} \tag{11.13}$$

Within equation (11.13) P_t^C and P_t^U are the Canadian and U.S. prices, respectively (both measured in U.S. dollars), D_t is the U.S. dollar index and Z is the average per unit discount for Canadian oil relative to U.S. oil (i.e., the price spread which is illustrated in Figure 11.3). It would appear that the set of coefficients in equation (11.13) comprise two distinct cointegrating vectors.

One possibility is that the two expressions in equation (11.13) are not linearly independent, in which case the Π matrix has rank one and there is just one cointegrating vector. To examine this possibility let's regress the Canadian price of oil on the U.S. price and the U.S. dollar index, with all variables in first differences to avoid spurious estimates.

```
oil_lm1 <- lm(diff(wcs) ~ diff(wti) + diff(dollar), data = oil)
summary(oil_lm1)
```

```
##
## Call:
## lm(formula = diff(wcs) ~ diff(wti) + diff(dollar), data = oil)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16.1042  -2.4966  -0.2821   2.0629  26.6326
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.008651   0.361126   0.024    0.981
## diff(wti)    1.036846   0.065533  15.822   <2e-16 ***
## diff(dollar) 0.135228   0.311561   0.434    0.665
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.062 on 196 degrees of freedom
## Multiple R-squared:  0.6384, Adjusted R-squared:  0.6347
## F-statistic:   173 on 2 and 196 DF,  p-value: < 2.2e-16
```

The above results reveal that $\beta_C \approx 1$ and $\gamma_C \approx 0$. These two outcomes imply that with the exception of the constant term, the pair of equations in equation (11.13) have approximately the same coefficients. This means that the linear independence assumption does not hold and we should expect just one cointegrating vector. This hypothesis is now tested using logs of the two oil prices.

11.4.2 Johansen Test for Cointegration

To implement the Johansen test for cointegration let's arrange the variables so that the price of WTI crude is the first variable in the system, the price of WCS crude is the second variable and the U.S. dollar index is the third variable. It is now necessary to identify a value for k, which is the number of lags to include in the VAR representation of the VECM. The *VARselect()* function can be used to select a value for k. In order to use the *VARselect()* function the log of the two prices must be added to the *oil* data frame.

```
oil <- oil %>% mutate(lnwti = log(wti),
                      lnwcs = log(wcs)) %>%
  dplyr::select(month,lnwti,lnwcs,dollar,wti,wcs)
head(oil)
```

```
## # A tsibble: 6 x 6 [1M]
##      month lnwti lnwcs dollar   wti   wcs
##      <mth> <dbl> <dbl>  <dbl> <dbl> <dbl>
## 1 2006 Jan  4.18  3.66  100    65.5  38.8
## 2 2006 Feb  4.13  3.36  100.   61.9  28.7
## 3 2006 Mar  4.14  3.60  100.   63.0  36.8
## 4 2006 Apr  4.25  3.95   99.7  70.2  52.0
## 5 2006 May  4.26  4.05   97.5  71.0  57.3
## 6 2006 Jun  4.26  3.94   98.7  71.0  51.6
```

The *VARselect()* function returns the following information criteria for lag selection:

```
lag_const <- VARselect(oil[,2:4], lag.max = 12, type = "const")[["selection"]]
lag_trend <- VARselect(oil[,2:4], lag.max = 12, type = "trend")[["selection"]]
lag_none <- VARselect(oil[,2:4], lag.max = 12, type = "none")[["selection"]]
cbind(lag_const,lag_trend,lag_none)
```

```
##        lag_const lag_trend lag_none
## AIC(n)         2         2        2
## HQ(n)          2         2        2
## SC(n)          2         2        2
## FPE(n)         2         2        2
```

Based on these results it is clear that $k = 2$ is the optimal number of lags to include in the Johansen cointegration test. The next step is to choose the specification of the Johansen test which maximizes the log of the likelihood function. The results below indicate that the "const" specification is optimal.

```
LL_const <- logLik(VAR(oil[,2:4],type="const",lag.max=3))
LL_trend <- logLik(VAR(oil[,2:4],type="trend",lag.max=3))
LL_none <- logLik(VAR(oil[,2:4],type="none",lag.max=3))
LL <- data.frame(LL_const,LL_trend,LL_none)
row.names(LL)[1] <- "Log Likelihood"
LL
```

```
##                 LL_const  LL_trend   LL_none
## Log Likelihood -24.25612 -24.49488 -29.38751
```

We can now follow the procedures from the previous chapter and implement the Johansen trace test for cointegration.

```
oil_johan <- ca.jo(oil[,2:4],ecdet="const",type="trace",K=2,spec="longrun")
summary(oil_johan)
```

```
##
## ######################
## # Johansen-Procedure #
## ######################
##
## Test type: trace statistic , without linear trend and constant in cointegration
##
## Eigenvalues (lambda):
## [1] 3.207670e-01 7.165093e-02 7.235941e-03 2.949030e-17
##
## Values of teststatistic and critical values of test:
##
##           test 10pct  5pct  1pct
## r <= 2 |  1.44  7.52  9.24 12.97
## r <= 1 | 16.16 17.85 19.96 24.60
## r = 0  | 92.74 32.00 34.91 41.07
##
## Eigenvectors, normalised to first column:
## (These are the cointegration relations)
##
##              lnwti.l2    lnwcs.l2   dollar.l2     constant
## lnwti.l2   1.00000000  1.00000000  1.00000000   1.00000000
## lnwcs.l2  -0.99480169 -0.27089792 -0.32206307  -0.17623885
## dollar.l2 -0.00487746  0.01056357 -0.02386985   0.07899164
## constant   0.16746275 -4.23324007 -5.81437524 -11.67466257
##
## Weights W:
## (This is the loading matrix)
##
##             lnwti.l2   lnwcs.l2     dollar.l2      constant
## lnwti.d   0.07916117 -0.1377156 -0.0008847862  9.471559e-16
## lnwcs.d   0.63344192 -0.1709945 -0.0015653582 -3.034307e-15
## dollar.d -0.82514540  1.6346025 -0.0104836794 -1.151004e-14
```

Recall from the previous section that the top table of these results are read and interpreted from the bottom up. The bottom row is the $r = 0$ null hypothesis, which is equivalent to saying that the rank of Π is zero and thus there are no cointegrating vectors. This null can be rejected and so we move to the row which is second from the bottom. This revised null which is that there are either zero or

one cointegrating vectors cannot be rejected. This outcome supports our earlier hypothesis that there is just one cointegrating vector in the system because the pair of long run pricing relationships are not linearly independent.

11.4.3 Estimating the VECM

As was discussed in the previous section the combinations of the *ca.jo()* and *cajorls()* functions are used to estimate the underlying VECM model. Let's begin by examining the estimated cointegrating vector.

```
vecm_est <-cajorls(oil_johan, r = 1)
vecm_est$beta
```

```
##                   ect1
## lnwti.l2   1.00000000
## lnwcs.l2  -0.99480169
## dollar.l2 -0.00487746
## constant   0.16746275
```

The values from the estimated cointegrated vector can be expressed in equation format as

$$P^{U}_{t-1} - 0.9948 P^{C}_{t-1} - 0.00488 D_{t-1} + 0.1675 = 0$$

This equation has three important features. The first is that the coefficient which links the logged prices is approximately equal to one, which is consistent with the previous regression results and which imply that the Canadian and U.S. oil prices are essentially perfect substitutes in the long run. The second is that the long run relationship between the price of U.S. crude oil and the U.S. dollar index is negative, which is consistent with the theory discussed above. The mean of the U.S. dollar index is approximately 100 and so it is necessary to multiply the U.S. dollar coefficient in the cointegrating vector by 100 to convert it to an elasticity. After doing this we notice that the long run elasticity of the U.S. price of oil to the U.S. dollar index is about 0.5. The third important feature is that the exponential of the constant term with value 0.1675 is a measure of the average discount facing Canadian oil producers.

Let's now examine the short run pricing dynamics for the estimated VECM. We do this by supplying the output of the *ca.jo()* function which is given by *oil_johan* as input into the *cajorls()* function. In this particular case we need to specify $r = 1$ because there is just one cointegrating vector.

```
#summary(vecm_est$rlm)
vecm_est$rlm
```

```
##
## Call:
## lm(formula = substitute(form1), data = data.mat)
##
## Coefficients:
##              lnwti.d   lnwcs.d   dollar.d
## ect1          0.07916   0.63344  -0.82515
## lnwti.dl1     0.34297   1.12940  -2.73858
## lnwcs.dl1    -0.08494  -0.44283   1.21970
## dollar.dl1   -0.01601  -0.01681   0.38318
```

The results above provide a summary of the short run coefficients. The columns correspond to the three equations: ΔP_t^U, ΔP_t^C and ΔD_t. The first row is the set of error correction coefficients, all three of which are significant at the 90 percent level or higher. The middle value in the top row (0.6334) shows that if $P_{t-1}^U - 0.9948 P_{t-1}^C - 0.00488 D_{t-1} + 0.1675$ takes on a positive value in period $t-1$ due a negative shock in P_{t-1}^C or D_{t-1} and/or a positive shock in P_{t-1}^U then in period t the Canadian price of crude will rise in order to correct about 63.3 percent of the previous period shock(s).

The last three rows in the above table reflect the lagged responses. For example, a one percent increase in ΔP_{t-1}^U is expected to raise ΔP_t^C by 1.129 percent. In contrast, a one percent increase in ΔP_{t-1}^C is expected to decrease ΔP_t^U by about 0.08 percent. An increase in ΔD_{t-1} decreases both ΔP_t^C and ΔP_t^U in period t.

Recall that the cointegrating vector is a stationary linear combination of the three variables. For this particular case $P_{t-1}^U - 0.9948 P_{t-1}^C - 0.00488 D_{t-1} + 0.1675$ is necessarily stationary. Figure 11.6 below shows that with the exception of a small number of outlier spikes toward the end of the time horizon, the cointegrated time series has the appearance of a stationary data series.

```
oil <- oil %>% mutate(
  cointvar = lnwti - 0.9948*lnwcs - 0.00488*dollar + 0.1675)

ect_plot <- oil %>% autoplot(cointvar)
ect_plot
```

11.4.4 Impulse Response

Let's use the *vec2var* function to convert the estimated oil price VECM into its VAR representation. Let Y_t denote vector which holds the pair of crude oil prices and the U.S. dollar index The *vec2var* function generates the Γ_1 and Γ_2 coefficients for the following VAR model.

$$Y_t = \alpha + Y_{t-1}\Gamma_1' + Y_{t-2}\Gamma_2' \tag{11.14}$$

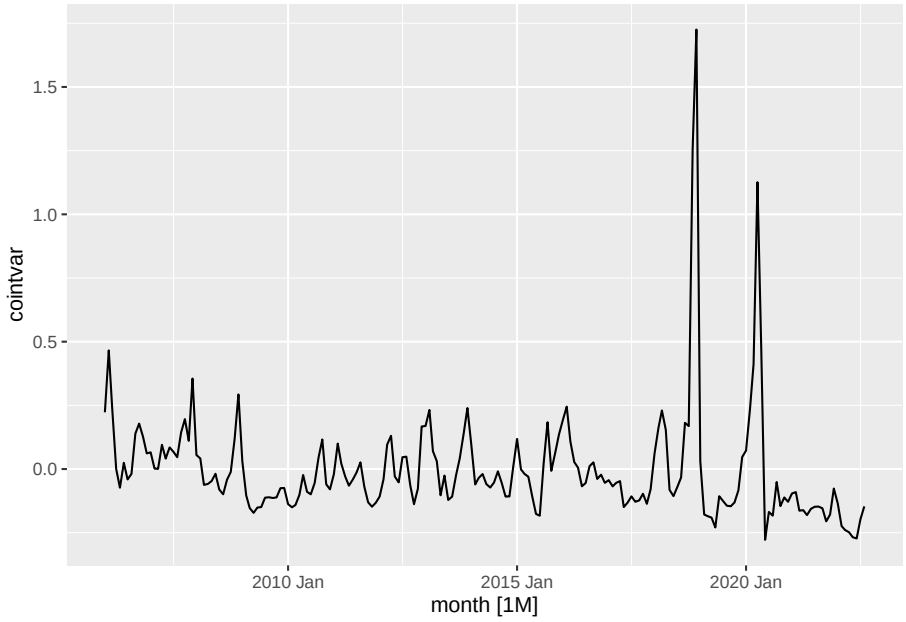

Figure 11.6: Stationarity of the Cointegrated Oil Price Series

After the conversion from VECM to VAR the *predict()* function from the *vars* package can be used to forecast values for the two oil prices and the U.S. dollar index (see Chapter 9 for details). Let's instead use the VAR representation for impulse response. In particular, let's examine how a shock in the price of U.S. oil affects the price of Canadian oil for future time periods.

```
var_oil <- vec2var(oil_johan,r=1)

irf_oil <- irf(var_oil, impulse = "lnwti", response = "lnwcs", n.ahead = 30, ortho = F
```

Figure 11.7 below shows the plot of the impulse response function (IRF) where the price of WTI creates the shock and the price of WCS receives the shock. The plot shows a positive response and some initial overshooting. However, unlike the impulse response functions which were featured in Chapter 10 in this case the IRF does not die away and eventually disappear. This is because the data which was used to estimate the VECM and which therefore underlies the VAR is non-stationary. An IRF for non-stationary data typically does not die away and for this reason is less valuable for policy analysis.

```
plot(irf_oil)
```

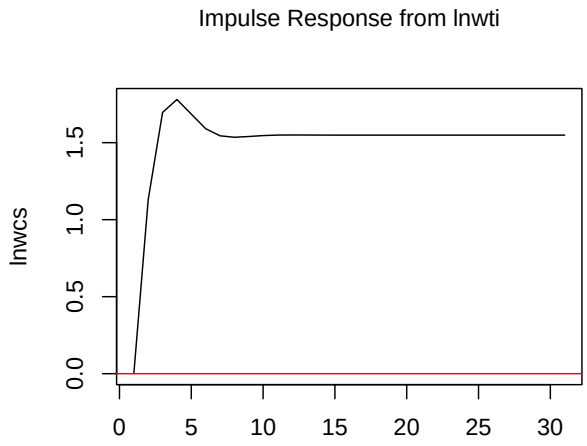

Figure 11.7: Impulse Response: U.S. Oil Price on Canadian Oil Price

11.5 Energy and Food Prices

In Canada and the U.S. most households consume fruits and vegetables which have been grown in intense agricultural regions in California, Florida and other parts of the country. Because of the long distances between these intensive agricultural regions and the towns and cities where the fruits and vegetables are consumed it follows that local prices are influenced by the cost of truck and train transportation. This suggests that we should expect a positive long term relationship between the monthly fruit and vegetable consumer price index (F&V CPI) and the U.S. price of diesel fuel.

Local prices of fruits and vegetables are also likely to be influenced by the growing conditions in the intensive agricultural regions of California. An important climate variable for a particular month is the extent that these regions are experiencing drought during that month. The level of drought is purely exogenous and as such should be modeled as such rather than being included in the F&V CPI and diesel fuel endogenous system of equations. While we shouldn't expect a large diesel fuel price response to a change in the California drought severity it is reasonable to expect the F&V CPI to respond to a change in the California drought conditions.

11.5.1 Preliminary Analysis

The data consists of the monthly U.S. CPI for fruits and vegetables, the monthly price of U.S. diesel fuel and a California agricultural drought index. The index is the average drought index for six top agricultural counties in California: Fresno, Monterey, Kern, Ventura, Sand Diego and Imperial. The drought index is available by U.S. county from the National Drought Mitigation Center (https://droughtmonitor.unl.edu/DmData/DataTables.aspx?state,ca). There are five indexes for each county: D0-D4, D1-D4, D2-D4, D3-D4, D4. The number following the "D" is a measure of drought severity with "0" meaning "abnormally dry" and "5" meaning "exceptional drought". This study defines the average value of the "D4" index across the six counties as a single measure of California drought. The other less extreme drought variables could also be used in the analysis but to keep things simple only the *d4* "exceptional" drought variable will be used. The weekly drought index data must be aggregated into monthly averages in order to use it with the CPI and diesel fuel price data.

The data resides in a *RDS* file, which was prepared in Chapter 3. Let's read this data in and plot the fruits and vegetables CPI and the log of the monthly diesel fuel price. Figure 11.8 reveals a strong upward trend in both data series, with significantly more volatility in the diesel fuel price then the fruits and vegetable CPI. It is not immediately obvious that there exists a cointegration relationship between these two variables.

```
veggie <- readRDS(here("data/ch10","veggie.RDS"))
head(veggie)
```

```
## # A tsibble: 6 x 7 [1M]
##      month veggies diesel  d2_4  d3_4    d4  mnth
##      <mth>   <dbl>  <dbl> <dbl> <dbl> <dbl> <int>
## 1 2000 Jan    202.  0.305     0     0     0     1
## 2 2000 Feb    203.  0.379     0     0     0     2
## 3 2000 Mar    202.  0.391     0     0     0     3
## 4 2000 Apr    201.  0.352     0     0     0     4
## 5 2000 May    202.  0.350     0     0     0     5
## 6 2000 Jun    200.  0.352     0     0     0     6
```

```
veggie_plot <- veggie %>% autoplot(veggies) +
  labs(y = "U.S. Veggie CPI: Base = 1982-1984", x="Date")
diesel_plot <- veggie %>% autoplot(diesel) +
  labs(y = "Log of U.S. Diesel Fuel Price", x="Date")
```

```
grid.arrange(veggie_plot, diesel_plot, ncol = 2)
```

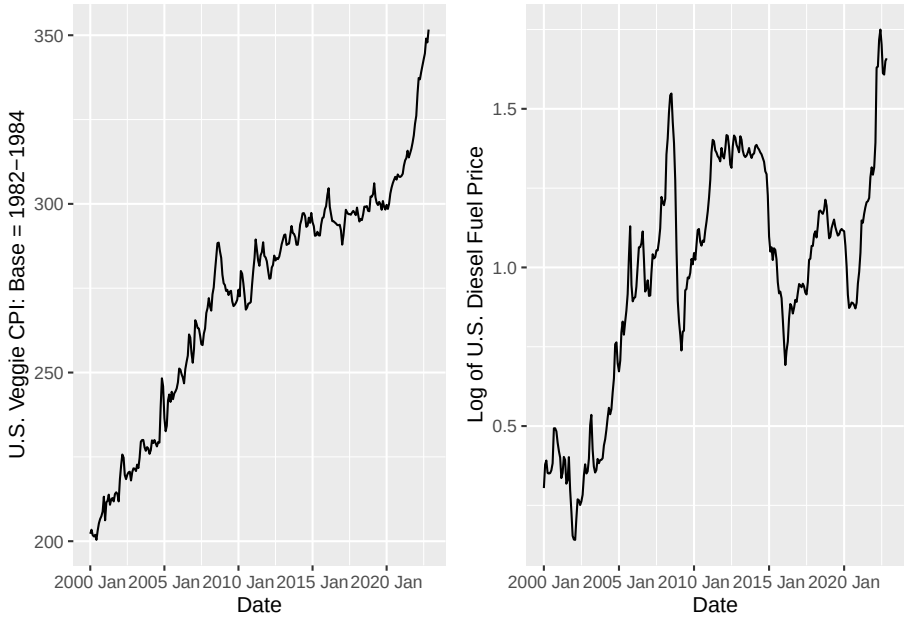

Figure 11.8: Fruits and Vegetable CPI and Logged Price of Diesel Fuel

The *d4* drought index is the percentage of the county which is experiencing exceptional drought, averaged across the six agricultural-intensive counties. Figure 11.9 reveals four exceptional drought periods in the six California counties, with the most severe taking place during 2015-16. Of course we expect a positive shock to the F&V CPI when then *d4* drought index bumps up in a particular month.

```
veggie %>% autoplot(d3_4) +
  labs(y = "D3-4 Drought Index for California Study Regions", x="Date")
```

11.5.2 Modeling Diesel and Drought Impacts

The standard cointegration and VECM methods are not appropriate for the case at hand because of the seasonal nature of the drought impacts. There is minor seasonality in the F&V CPI but this seasonality is not statistically significant. What is important is that drought conditions should be compared with a 12 month lag rather than a one month lag for the purpose of identifying the CPI impacts. This seasonal difference approach is required because there is strong seasonality in the drought index, with higher values emerging in the summer and lower values (mostly zero) emerging in the winter. This particular

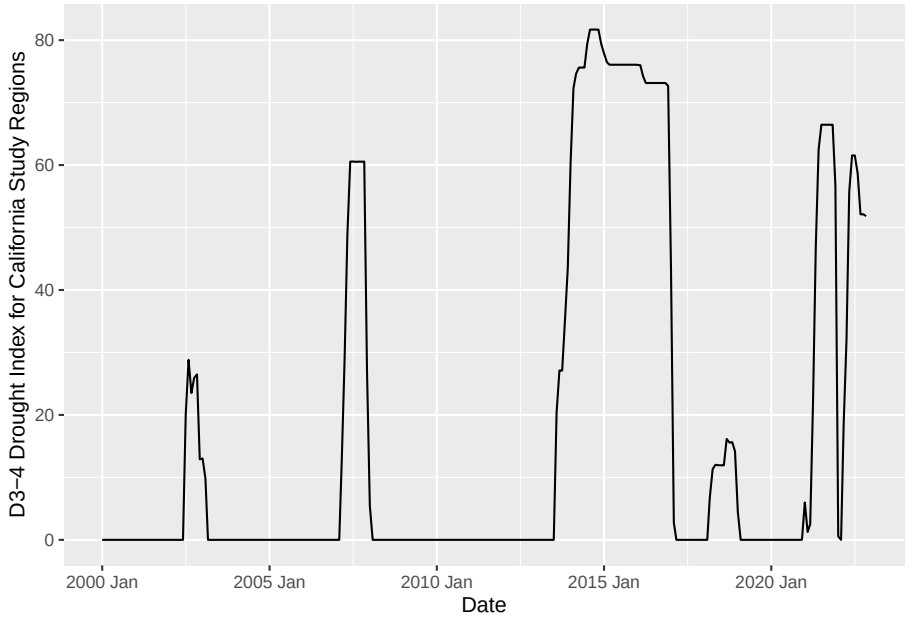

Figure 11.9: Monthly Rainfall in San Franciso, California

approach is closely related to the seasonal differencing which was featured in Chapter 6.

We begin with a structural bi-variate VAR model with a single lag and with the inclusion of the current and lagged values of the exogenous drought index variable. To write down this model the F&V CPI, the price of diesel fuel and the *d4* drought index in period t are denoted p_t, f_t and d_t respectively. The CPI equation in the VAR model can be expressed as

$$p_t = a_0 + a_2 f_t + a_3 d_t + b_1 p_{t-1} + b_2 f_{t-1} + b_3 d_{t-1} + e_t \tag{11.15}$$

We won't formally prove that the F&V CPI and the price of diesel fuel are non-stationary but the non-stationarity outcome should be obvious from Figure 11.8. To make all variables in the model stationary we need to represent equation (11.15) as an error correction model (ECM) with a potential single cointegrating vector. This is accomplished using the steps which were presented in Section 11.2.

$$p_t - p_{t-1} = c_0 + a_2 (f_t - f_{t-1}) + a_3 (d_t + p_{t-1}) + \lambda (p_{t-1} - \alpha - \beta_1 f_{t-1} + \beta_2 d_{t-1}) + e_t \tag{11.16}$$

To incorporate the seasonal differencing, let L_{12} denote the 12 month lag operator and let Δ_{12} denote the 12 month difference operator. Specifically, L_{12}P_t = P_{t-12}) and *{12}P_t = P_t-P*{t-12}. Now multiply equation (11.16) through by $1 - L_{12}$. If we define $\Phi_t = \Delta p_t - \Delta_{12} p_t$, $\Gamma_t = \Delta f_t - \Delta_{12} f_t$ and $\Sigma_t = \Delta d_t - \Sigma_{12} d_t$ we can write a revised expression for equation (11.16) as follows:

$$\Phi_t = c_0 + a_2\Gamma_t + a_3\Sigma_t + \lambda\left(\Delta_{12} p_{t-1} - \alpha - \beta_1 \Delta_{12} f_{t-1} - \beta_2 \Delta_{12} d_{t-1}\right) + e_t \quad (11.17)$$

In this case the long run pricing relationship has been expressed as a seasonal difference.

$$\Delta_{12} p_{t-1} = \alpha + \beta_1 \Delta_{12} f_{t-1} + \beta_2 \Delta_{12} d_{t-1} \quad (11.18)$$

If the two endogenous variables, $\Delta_{12} p_{t-1}$ and $\Delta_{12} f_{t-1}$, are cointegrated then the cointegrating vector is $\{1, \beta_1, \beta_2\}$. Notice that the coefficient of the exogenous *d4* drought index is part of the cointegrating vector. The rest of equation (11.16) has the usual interpretation. The a_2 and a_3 coefficients are measures of the short-run dynamics, and λ is the error correction coefficient.

11.5.3 Empirical Analysis

We begin the empirical analysis by testing to determine if the F&V CPI and the price of diesel fuel are seasonally cointegrated while accounting for the exogenous influence of the California drought index. Using the Engle-Granger two-step procedure and equation (11.18) we will first estimate the long run relationship between $\Delta_{12} p_{t-1}$, $\Delta_{12} f_{t-1}$ and $\Delta_{12} d_{t-1}$. For the purpose of programming let *D.p*, *D.f* and *D.d* denote the regular first difference of the three variables. Similarly, let *p_s*, *f_s* and *d_s* denote the 12 month seasonal difference of the three variables. Finally, let *Phi*, *Gamma* and *Sigma* be the programming variables which correspond to Φ_t, Γ_t and Σ_t, respectively. Let's create these variables and insert them into a new data frame.

```
veggie2 <- veggie %>%
  mutate(p = veggies,
         f = diesel,
         d = d4,
        D.p = difference(p),
         p_s = difference(p,12),
         D.f = difference(f),
         f_s = difference(f,12),
         D.d = difference(d),
         d_s = difference(d4,12)) %>%
```

```
  dplyr::select(month,D.p,D.f,D.d,p_s,f_s,d_s) %>%
  slice(-(1:14))
```

Figure 11.8 shows that the F&V CPI surged to a very high level, beginning in about January of 2022. This surge is clearly an outlier relative to the rest of the data in Figure 11.8. To mitigate the impact of these outlier observations a dummy variable will be included in the cointegrating equation. Specifically, let *dumout* take on a value of one if the data is January 2022 or later and zero otherwise. Let's add the outlier dummy variable to the *veggie2* data frame and then estimate the cointegrating equation, which is given by equation (11.18)

```
veggie2 <- veggie2 %>% mutate(dumout = ifelse(as.Date(month)>="2022-01-01",1,0))

stage1_lm <- lm(p_s~f_s +d_s+dumout,data=veggie2)
summary(stage1_lm)
```

```
##
## Call:
## lm(formula = p_s ~ f_s + d_s + dumout, data = veggie2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -19.1500  -4.6376   0.4498   4.4599  21.2113
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.78931    0.43922  10.904  < 2e-16 ***
## f_s         10.47593    1.90407   5.502 9.08e-08 ***
## d_s          0.05885    0.02444   2.408   0.0168 *
## dumout      17.87529    2.24506   7.962 5.47e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 6.841 on 257 degrees of freedom
## Multiple R-squared:  0.3531, Adjusted R-squared:  0.3456
## F-statistic: 46.76 on 3 and 257 DF,  p-value: < 2.2e-16
```

The regression results show that all of the estimated coefficients in the long run equation are statistically significant and have the anticipated signs. The price of diesel has been logged and so the coefficient on the p_s variable (10.47) divided by the mean of the F&V CPI (272) provides an estimate of the long run relationship between the F&V CPI and the price of diesel fuel, expressed as a percentage. The calculated value is 0.038 which means that a permanent 10 percent increase in the price of diesel fuel will result in slightly less than a half

of one percent increase in the F&V CPI. Given the importance of diesel fuel in the transportation of fruits and vegetables in the U.S. this long term impact is surprisingly small.

One has to be careful when calculating a long run elasticity for the drought index because the majority of values for this index take on a value of zero. Nevertheless, accounting for the zero values the mean value of the *d4* index is 8.44. If we multiply the estimated coefficient on the d_s variable (0.0588) by the 8.44/272 ratio, the resulting elasticity is 0.0018. This very small value implies that a permanent increase in the *d4* price index by 10 percent will raise the F&V CPI by slightly less than a fifth of one percent. This small impact suggests that fruit and vegetable production in California is likely not particularly sensitive to droughts, possible because most crops rely on ground water irrigation rather then being rain fed.

The discussion in the previous paragraph assumes that the variables in question are cointegrated. We can establish cointegration by conducting an ADF test on the regression residuals and rejecting the unit root null hypothesis. The ADF test statistic with the *none* option and lags automatically selected by according to the *AIC* criteria is as follows.

```
df_stage1 <- ur.df(stage1_lm$residuals,type="none",selectlags="AIC")
summary(df_stage1)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.3163  -2.2881  -0.4723   2.0190  14.7163
##
## Coefficients:
##            Estimate Std. Error t value Pr(>|t|)
## z.lag.1    -0.24102    0.03545  -6.798 7.36e-11 ***
## z.diff.lag  0.32724    0.05890   5.556 6.89e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 3.697 on 257 degrees of freedom
## Multiple R-squared:  0.1888, Adjusted R-squared:  0.1825
## F-statistic: 29.91 on 2 and 257 DF,  p-value: 2.103e-12
##
##
## Value of test-statistic is: -6.7984
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.58 -1.95 -1.62
```

The ADF test statistic is $\tau = -6.798$. In the previous chapter we noted that the special critical for this test statistic when used in the Engle-Granger test is -4.35 at the 99 percent confidence level. We can therefore strongly reject the unit root null and conclude that Δp_t, Δf_t and Δd_t set of variables are cointegrated.

We can now estimate the error correction model which is given by equation (11.16). This requires lagging the residuals from the first stage regression and replacing the cointegrating expression in equation (11.16) with these lagged values. One lag of the Φ variable is included in this stage two regression to control for autocorrelation.

```
Lag.resid <- lag(stage1_lm$residuals)
veggie2 <- veggie2[-1,] %>% mutate(L.resid = -1*Lag.resid[-1],
                                   Phi = D.p-p_s,
                                   Gamma = D.f-f_s,
                                   Sigma = D.d - d_s)

stage2_lm <- lm(Phi~Gamma+Sigma+L.resid+dumout+lag(Phi),data=veggie2)
summary(stage2_lm)
```

```
##
## Call:
## lm(formula = Phi ~ Gamma + Sigma + L.resid + dumout + lag(Phi),
##     data = veggie2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.9916 -1.5967 -0.2938  1.2480 13.1277
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -3.98213    0.32244 -12.350  < 2e-16 ***
## Gamma        10.59165    0.93905  11.279  < 2e-16 ***
## Sigma         0.04162    0.01046   3.977 9.11e-05 ***
```

```
## L.resid       0.82698    0.06191  13.358  < 2e-16 ***
## dumout      -14.39792    1.29082 -11.154  < 2e-16 ***
## lag(Phi)      0.08588    0.06081   1.412    0.159
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.716 on 253 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.8904, Adjusted R-squared:  0.8883
## F-statistic: 411.3 on 5 and 253 DF,  p-value: < 2.2e-16
```

All of the estimated coefficients except for the lag of Φ_t are statistically significant and all have the anticipated signs. The coefficients on the Γ and Σ variables are similar in magnitude to the coefficients on Δf_t and Δd_t in the cointegrating equation. This outcome is expected because in this particular case the long run equation is a seasonal difference, and thus not too different than a regular difference. The most important coefficient in the regression results is the 0.827 estimate of the error correction coefficient. This relatively high value implies that the F&V CPI and diesel price adjusts quite rapidly to regain a long run equilibrium relationship when one or more of the variables are shocked.

11.6 Conclusions

The error correction model (ECM) and vector error correction model (VECM) are highly useful for policy analysis because they allow show and long term relationships to be distinguished. Commodity prices are commonly non stationary and so cannot be directly included in a vector autoregression (VAR). Although it is possible to estimate a VAR with the first difference of commodity prices doing so generally results in a loss of information about the long term pricing relationships. For cointegrated commodity prices the ECM and VECM can be used the estimate the long term pricing relationships even though the individual data series are non stationary.

The key results in this chapter concern the magnitude of the speed of adjustment coefficients within the estimated ECM or VECM. In all of the empirical applications these coefficients are statistically significant. In the first application with biodiesel and soybean oil the relatively small speed of adjustment coefficient is equal to 0.029. In contrast, the speed of adjustment of the price of Canadian WCS is relatively large at 0.633. This outcome is not surprising because Canadian WCS and U.S. WTI are strong substitutes. For the case of the fruits and vegetables CPI and diesel fuel, the speed of adjustment is even larger at 0.827.

Throughout this chapter it has been emphasized that cointegrated data should be estimated with a VECM rather than a VAR because the former gives more

efficient estimates. It was also recommended that if the goal is to forecast then the VECM should be estimated and the VAR representation of the estimated VECM should be used to generate the forecast. This conclusion does not neccesarily hold in all situations. **?** and **?** discuss the literature which debates this topic.

Engle and Granger [1987] argue that estimating a VAR directly with cointegrated data omits important long run restrictions on the coefficients and the resulting efficiency loss is expected to give rise to reduced forecasting performance relative to an estimated VECM with a VAR representation. Others note that if forecasting with cointegrated data is the only goal then using a directly estimated VAR is both consistent and doesn't run the risk that the estimated cointegrating vector is not accurate. There is no theoretical winner or loser in this debate and so the question as whether it is better to forecast with a directly estimated VAR or a VAR representation of an estimate VECM is an empirical question. When forecasting macroeconomic variables **?** find evidence which supports the VECM approach. Moreover, the advantage of the VECM approach increases as the strength of the cointegration increases. In contrast, **?** compare a direct and indirect VAR approach when forecasting cattle prices and conclude that the direct VAR results in a lower mean squared error.

? use a threshold vector error correction model (TVECM) to model the price of apples in the EU market. A distinguishing feature of a TVECM is that there exists a threshold value for the error correction term (i.e., the deviation of prices from their long run equilibrium values), above which the speed of adjustment is relatively rapid and below which the speed of adjustment is relatively slow. **?** note that a TVECM is appropriate for the German apple market because of strong seasonality in the available domestic supply of apples. They estimate that the speed of adjustment coefficient is relatively small in the winter (-0.0044) and relatively large in the summer (-0.118).

Bibliography

Donald W. K. Andrews. Tests for parameter instability and structural change with unknown change point. *Econometrica*, 61(4):821–856, 1993. ISSN 00129682, 14680262. URL http://www.jstor.org/stable/2951764.

Donald W. K. Andrews and Werner Ploberger. Optimal tests when a nuisance parameter is present only under the alternative. *Econometrica*, 62(6):1383–1414, 1994. ISSN 00129682, 14680262. URL http://www.jstor.org/stable/2951753.

Frank Asche, Bård Misund, and Atle Oglend. Determinants of the atlantic salmon futures risk premium. *Journal of Commodity Markets*, 2(1):6–17, 2016. ISSN 2405-8513. doi: https://doi.org/10.1016/j.jcomm.2016.07.001. URL https://www.sciencedirect.com/science/article/pii/S2405851315300210.

Jungho Baek and Won W. Koo. On the upsurge of u.s. food prices revisited. *Economic Modelling*, 42:272–276, 2014.

Jushan Bai and Pierre Perron. Estimating and testing linear models with multiple structural changes. *Econometrica*, 66(1):47–78, 1998. ISSN 00129682, 14680262. URL http://www.jstor.org/stable/2998540.

Christiane Baumeister, Lutz Kilian, Wolf Wagner, and Deren Unalmis. Do oil price increases cause higher food prices? *Economic Policy*, 29(80):691–747, 2014.

Dan Ben-David and David H. Papell. Slowdowns and Meltdowns: Postwar Growth Evidence From 74 Countries. *The Review of Economics and Statistics*, 80(4):561–571, 11 1998.

Luca Benati. Drift and breaks in labor productivity. *Journal of Economic Dynamics and Control*, 31(8):2847–2877, 2007. ISSN 0165-1889. doi: https://doi.org/10.1016/j.jedc.2006.11.004. URL https://www.sciencedirect.com/science/article/pii/S0165188906002090.

Ben S Bernanke. Alternative explanations of the money-income correlation. Working Paper 1842, National Bureau of Economic Research, February 1986. URL http://www.nber.org/papers/w1842.

Olivier Jean Blanchard and Danny Quah. The Dynamic Effects of Aggregate Demand and Supply Disturbances. *American Economic Review*, 79 (4):655–673, September 1989. URL https://ideas.repec.org/a/aea/aecrev/v79y1989i4p655-73.html.

Catherine Chambers, Paul Chambers, and John Whitehead. Economic growth and threatened and endangered species listings: A var analysis. *Open Environmental Sciences*, 3, 02 2008. doi: 10.2174/1876325100903010079.

Zhan-Ming Chen, Liyuan Wang, Xiao-Bing Zhang, and Xinye Zheng. The comovement and asymmetry between energy and grain prices: Evidence from the crude oil and corn markets. *Energies*, 12(7), 2019.

Gregory C. Chow. Tests of equality between sets of coefficients in two linear regressions. *Econometrica*, 28(3):591–605, 1960. ISSN 00129682, 14680262.

Pavel Ciaian and d'Artis Kancs. Interdependencies in the energy–bioenergy–food price systems: A cointegration analysis. *Resource and Energy Economics*, 33(1):326–348, 2011.

Rex F. Daly. Houthakker, h. s., and lester d. taylor, consumer demand in the united states, 1929–1970, analyses and projections, cambridge, harvard university press, 1966, x + 214 pp. ($6.00). *American Journal of Agricultural Economics*, 49(2):528–530, 1967. doi: https://doi.org/10.2307/1237232. URL https://onlinelibrary.wiley.com/doi/abs/10.2307/1237232.

Matt Dancho and Davis Vaughan. *timetk: A Tool Kit for Working with Time Series in R*, 2022. URL https://CRAN.R-project.org/package=timetk. R package version 2.8.1.

James E. H. Davidson, David F. Hendry, Frank Srba, and Stephen Yeo. Econometric modelling of the aggregate time-series relationship between consumers' expenditure and income in the united kingdom. *The Economic Journal*, 88 (352):661–692, 1978. ISSN 00130133, 14680297. URL http://www.jstor.org/stable/2231972.

Michael C. Davis. Environmental regulations and the increasing seasonality of gasoline prices. *Applied Economics Letters*, 16(16):1613–1616, 2009. doi: 10.1080/13504850701591291.

David A. Dickey and Wayne A. Fuller. Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American Statistical Association*, 74(366):427–431, 1979. ISSN 01621459. URL http://www.jstor.org/stable/2286348.

Brian M. Dillon and Christopher B. Barrett. Global oil prices and local food prices: Evidence from east africa. *American Journal of Agricultural Economics*, 98(1):154–171, 2016.

Xiaodong Du and Lihong Lu McPhail. Inside the black box: the price linkage and transmission between energy and agricultural markets. *The Energy Journal*, 33(2):171–194, 2012.

Mardi Dungey and Renée Fry. The identification of fiscal and monetary policy in a structural var. *Economic Modelling*, 26(6):1147–1160, 2009. ISSN 0264-9993. doi: https://doi.org/10.1016/j.econmod.2009.05.001. URL https://www.sciencedirect.com/science/article/pii/S0264999309000807.

Robert F. Engle and C. W. J. Granger. Co-integration and error correction: Representation, estimation, and testing. *Econometrica*, 55(2):251–276, 1987.

Victoria Eriksson and Robert Lundmark. A cointegration analysis of the nordic roundwood markets. *Forests*, 11(9), 2020. ISSN 1999-4907. URL https://www.mdpi.com/1999-4907/11/9/1007.

Alvaro Escribano, J. Ignacio Peña, and Pablo Villaplana. Modelling electricity prices: International evidence. *Oxford Bulletin of Economics and Statistics*, 73(5):622–650, 2011. doi: https://doi.org/10.1111/j.1468-0084.2011.00632.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1468-0084.2011.00632.x.

Ondrej Filip, Karel Janda, Ladislav Kristoufek, and David Zilberman. Food versus fuel: An updated and expanded evidence. *Energy Economics*, 82:152–166, 2019. ISSN 0140-9883.

Philip Garcia and Raymond M. Leuthold. A selected review of agricultural commodity futures and options markets. *European Review of Agricultural Economics*, 31(3):235–272, 09 2004. ISSN 0165-1587. doi: 10.1093/erae/31.3.235. URL https://doi.org/10.1093/erae/31.3.235.

Christopher L. Gilbert, Luc Christiaensen, and Jonathan Kaminski. Food price seasonality in africa: Measurement and extent. *Food Policy*, 67:119–132, 2017. ISSN 0306-9192. doi: https://doi.org/10.1016/j.foodpol.2016.09.016. URL https://www.sciencedirect.com/science/article/pii/S0306919216303840. Agriculture in Africa – Telling Myths from Facts.

C. W. J. Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica*, 37(3):424–438, 1969.

C.W.J. Granger and P. Newbold. Spurious regressions in econometrics. *Journal of Econometrics*, 2(2):111–120, 1974.

Allan W. Gregory and Bruce E. Hansen. Residual-based tests for cointegration in models with regime shifts. *Journal of Econometrics*, 70(1):99–126, 1996. ISSN 0304-4076. doi: https://doi.org/10.1016/0304-4076(69)41685-7. URL https://www.sciencedirect.com/science/article/pii/0304407669416857.

Luciano Gutierrez, Francesco Piras, and Pier Paolo Roggero. A global vector autoregression model for the analysis of wheat export prices. *American Journal of Agricultural Economics*, 97(5):1494–1511, 2015. ISSN 00029092, 14678276. URL http://www.jstor.org/stable/24477218.

Niels Haldrup, Frank S. Nielsen, and Morten Ørregaard Nielsen. A vector autoregressive model for electricity prices subject to long memory and regime switching. *Energy Economics*, 32(5):1044–1058, 2010. ISSN 0140-9883. doi: https://doi.org/10.1016/j.eneco.2010.02.012. URL https://www.sciencedirect.com/science/article/pii/S0140988310000368.

Håvard Halland, Rüya Perincek, and Jan Rieländer. Links between energy and food must be weakened. *Financial Times*, May 27 2022.

Amna Awad Abdel Hameed and Fatimah Mohamed Arshad. The impact of petroleum prices on vegetable oil prices: Evidence from co-integration tests. *Oil Palm Industry Economic Journal*, 9:31–40, September 2009.

Bruce E. Hansen. Approximate asymptotic p values for structural-change tests. *Journal of Business and Economic Statistics*, 15(1):60–67, 1997. ISSN 07350015. URL http://www.jstor.org/stable/1392074.

Bruce E. Hansen. The new econometrics of structural change: Dating breaks in u.s. labour productivity. *Journal of Economic Perspectives*, 15(4):117–128, December 2001. doi: 10.1257/jep.15.4.117. URL https://www.aeaweb.org/articles?id=10.1257/jep.15.4.117.

Ardian Harri, Lanier Nalley, and Darren Hudson. The relationship between oil, exchange rates, and commodity prices. *Journal of Agricultural and Applied Economics*, 41(2):501–510, 2009.

A. C. HARVEY and G. D. A. PHILLIPS. Maximum likelihood estimation of regression models with autoregressive-moving average disturbances. *Biometrika*, 66(1):49–58, 04 1979. ISSN 0006-3444. doi: 10.1093/biomet/66.1.49. URL https://doi.org/10.1093/biomet/66.1.49.

R.J. Hyndman and G. Athanasopoulos. *Forecasting: principles and practice*. OTexts: Melbourne, Australia, 2021. https://otexts.com/fpp3/(visited 2022-09-06).

Hyun Joung Jin and Dragan Miljkovic. An analysis of multiple structural breaks in us relative farm prices. *Applied Economics*, 42(25):3253–3265, 2010. doi: 10.1080/00036840802600368.

Soren Johansen. Estimation and hypothesis testing of cointegration vectors in gaussian vector autoregressive models. *Econometrica*, 59(6):1551–1580, 1991.

Dale W. Jorgenson. Capital theory and investment behavior. *The American Economic Review*, 53(2):247–259, 1963. ISSN 00028282. URL http://www.jstor.org/stable/1823868.

Katarina Juselius. *The Cointegrated VAR Model: Methodology and Applications*. 12 2006. ISBN 9780199285662. doi: 10.1093/oso/9780199285662.001.0001.

Lutz Kilian. Not all oil price shocks are alike: Disentangling demand and supply shocks in the crude oil market. *The American Economic Review*, 99(3):1053–1069, 2009.

Lutz Kilian and Xiaoqing Zhou. The Econometrics of Oil Market VAR Models. CESifo Working Paper Series 8153, CESifo, 2020. URL https://ideas.repec.org/p/ces/ceswps/_8153.html.

Surender Kumar, Shunsuke Managi, and A. Matsuda. Stock prices of clean energy firms, oil and carbon markets: A vector autoregressive analysis . *Energy Economics*, 34:215–226, 2012.

David K. Lambert and Dragan Miljkovic. The sources of variability in U.S. food prices. *Journal of Policy Modeling*, 32(2):210–222, March 2010.

Manuel Landajo and María José Presno. The prices of renewable commodities: a robust stationarity analysis. *Australian Journal of Agricultural and Resource Economics*, 66(2):447–470, 2022. doi: https://doi.org/10.1111/1467-8489.12468.

Junsoo Lee and Mark Strazicich. Minimum lagrange multiplier unit root test with two structural breaks. *The Review of Economics and Statistics*, 85(4): 1082–1089, 2003.

Peter Levi and Gergely Molnar. How the energy crisis is exacerbating the food crisis. Technical report, xxx, Paris: France, 2022.

H.C. Liao and Y.B. Suen. Dating breaks for global crude oil prices and their volatility: a possible price band for global crude prices. *Energy Studies Review*, 14(2):189–206, 2006. doi: 10.1080/00036840802600368.

Robert B. Litterman and Laurence Weiss. Money, real interest rates, and output: A reinterpretation of postwar u.s. data. *Econometrica*, 53(1):129–156, 1985.

Robert E. Lucas. Econometric policy evaluation: A critique. *Carnegie-Rochester Conference Series on Public Policy*, 1:19–46, 1976. ISSN 0167-2231. doi: https://doi.org/10.1016/S0167-2231(76)80003-6. URL https://www.sciencedirect.com/science/article/pii/S0167223176800036.

Julio J. Lucia and Eduardo S. Schwartz. Electricity prices and power derivatives: Evidence from the nordic power exchange. *Review of Derivatives Research*, 5:5–50, 2002.

Matthew J. MacLachlan, Carolyn A. Chelius, and Gianna Short. Time-series methods for forecasting and modeling uncertainty in the food price outlook. Technical Bulletin Number 1957 August, USDA Economic Research Service, 2022.

Svetlana Maslyuk and Russell Smyth. Unit root properties of crude oil spot and futures prices. *Energy Policy*, 36(7):2591–2600, 2008. ISSN 0301-4215. doi: https://doi.org/10.1016/j.enpol.2008.03.018. URL https://www.sciencedirect.com/science/article/pii/S0301421508001547.

Terence C. Mills and Raphael N. Markellos. *The Econometric Modelling of Financial Time Series*. Cambridge University Press, 3 edition, 2008. doi: 10.1017/CBO9780511817380.

M.K. Minlah, X. Zhang, P.N. Ganyoh, and A. Bibi. Does the environmental kuznets curve for deforestation exist for ghana? evidence from the bootstrap rolling window granger causality test approach. *Forestry Economics Review*, 3:38–52, 2021.

Andrés García Mirantes, Javier Población, and Gregorio Serna. The stochastic seasonal behaviour of natural gas prices. *European Financial Management*, 18(3):410–443, 2012. doi: https://doi.org/10.1111/j.1468-036X.2009.00533.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1468-036X.2009.00533.x.

Manuel Moreno, Alfonso Novales, and Federico Platania. Long-term swings and seasonality in energy markets. *European Journal of Operational Research*, 279(3):1011–1023, 2019. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2019.05.042. URL https://www.sciencedirect.com/science/article/pii/S0377221719304722.

Robert J. Myers, Stanley R. Johnson, Michael Helmar, and Harry Baumes. Long-run and short-run co-movements in energy prices and the prices of agricultural feedstocks for biofuel. *American Journal of Agricultural Economics*, 96(4):991–1008, 2014.

Valeri Natanelov, Mohammad J. Alam, Andrew M. McKenzie, and Guido Van Huylenbroeck. Is there co-movement of agricultural commodities futures prices and crude oil? *Energy Policy*, 39(9):4971–4984, 2011.

Saban Nazlioglu. World oil and agricultural commodity prices: Evidence from nonlinear causality. *Energy Policy*, 39(5):2935–2943, 2011.

Saban Nazlioglu and Ugur Soytas. World oil prices and agricultural commodity prices: Evidence from an emerging market. *Energy Economics*, 33(3):488–496, 2011.

Saban Nazlioglu and Ugur Soytas. Oil price, agricultural commodity prices, and the dollar: A panel cointegration and causality analysis. *Energy Economics*, 34(4):1098–1104, 2012.

Charles R. Nelson and Charles R. Plosser. Trends and random walks in macroeconmic time series: Some evidence and implications. *Journal of Monetary Economics*, 10(2):139–162, 1982a. ISSN 0304-3932. doi: https:

//doi.org/10.1016/0304-3932(82)90012-5. URL https://www.sciencedirect.com/science/article/pii/0304393282900125.

Charles R. Nelson and Charles R. Plosser. Trends and random walks in macroeconmic time series: Some evidence and implications. *Journal of Monetary Economics*, 10(2):139–162, 1982b.

Atle Oglend. Recent trends in salmon price volatility. *Aquaculture Economics & Management*, 17(3):281–299, 2013. doi: 10.1080/13657305.2013.812155.

Mitchell O'Hara-Wild, Rob Hyndman, and Earo Wang. *fable: Forecasting Models for Tidy Time Series*, 2021. URL https://CRAN.R-project.org/package=fable. R package version 0.3.1.

David Oluseun Olayungbo. Global oil price and food prices in food importing and oil exporting developing countries: A panel ardl analysis. *Heliyon*, 7(3): e06357, 2021. ISSN 2405-8440.

David Orden and Paul L. Fackler. Identifying monetary impacts on agricultural prices in var models. *American Journal of Agricultural Economics*, 71(2): 495–502, 1989. ISSN 00029092, 14678276. URL http://www.jstor.org/stable/1241620.

Massimo Peri and Lucia Baldi. Vegetable oil market and biofuel policy: An asymmetric cointegration approach. *Energy Economics*, 32(3):687–693, 2010.

Pierre Perron. The great crash, the oil price shock, and the unit root hypothesis. *Econometrica*, 57(6):1361–1401, 1989. ISSN 00129682, 14680262. URL http://www.jstor.org/stable/1913712.

B. Pfaff. *Analysis of Integrated and Cointegrated Time Series with R*. Springer, New York, second edition, 2008a. URL http://www.pfaffikus.de. ISBN 0-387-27960-1.

Bernhard Pfaff. Var, svar and svec models: Implementation within r package vars. *Journal of Statistical Software*, 27(4):1–32, 2008b. URL https://www.jstatsoft.org/index.php/jss/article/view/v027i04.

P.C.B. Phillips. Understanding spurious regressions in econometrics. *Journal of Econometrics*, 33(3):311–340, 1986.

Charles I. Plosser and G.William Schwert. Money, income, and sunspots: Measuring economic relationships and the effects of differencing. *Journal of Monetary Economics*, 4(4):637–660, 1978.

Duo Qin. Rise of var modelling approach*. *Journal of Economic Surveys*, 25(1): 156–174, 2011. doi: https://doi.org/10.1111/j.1467-6419.2010.00637.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-6419.2010.00637.x.

Richard E. Quandt. Tests of the hypothesis that a linear regression system obeys two separate regimes. *Journal of the American Statistical Association*, 55(290):324–330, 1960. ISSN 01621459. URL http://www.jstor.org/stable/2281745.

Anthony N. Rezitis. The relationship between agricultural commodity prices, crude oil prices and us dollar exchange rates: a panel var approach and causality analysis. *International Review of Applied Economics*, 29(3):403–434, 2015.

Jeffrey A. Ryan and Joshua M. Ulrich. *xts: eXtensible Time Series*, 2020. URL https://CRAN.R-project.org/package=xts. R package version 0.12.1.

Sayed Saghaian, Mehdi Nemati, Cory Walters, and Bo Chen. Asymmetric price volatility transmission between u.s. biofuel, corn, and oil markets. *Journal of Agricultural and Resource Economics*, 43(1):46–60, 2018.

Thomas Sargent and Christopher Sims. Business cycle modeling without pretending to have too much a priori economic theory. Working Papers 55, Federal Reserve Bank of Minneapolis, 1977. URL https://EconPapers.repec.org/RePEc:fip:fedmwp:55.

Thomas J. Sargent and Neil Wallace. Rational expectations and the dynamics of hyperinflation. *International Economic Review*, 14(2):328–350, 1973. ISSN 00206598, 14682354. URL http://www.jstor.org/stable/2525924.

Teresa Serra and David Zilberman. Biofuel-related price transmission literature: A review. *Energy Economics*, 37:141–151, 2013.

Mahdi Shahrazi, Saman Ghaderi, and Bahram Sanginabadi. Commodity prices and inflation: an application of structural var. *Applied Economics*, 0(0):1–11, 2022. doi: 10.1080/00036846.2022.2108753.

Christopher A. Sims. Money, income, and causality. *The American Economic Review*, 62(4):540–552, 1972. ISSN 00028282. URL http://www.jstor.org/stable/1806097.

Christopher A. Sims. Macroeconomics and reality. *Econometrica*, 48(1):1–48, 1980.

Christopher A. Sims. Are forecasting models usable for policy analysis? *Quarterly Review*, 10(Win):2–16, 1986.

Carsten Sorensen. Modeling seasonality in agricultural commodity futures. *Journal of Futures Markets*, 22(5):393–426, 2002. doi: https://doi.org/10.1002/fut.10017. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/fut.10017.

James H. Stock and Mark W. Watson. Evidence on structural instability in macroeconomic time series relations. *Journal of Business & Economic Statistics*, 14(1):11–30, 1996. doi: 10.1080/07350015.1996.10524626.

James H. Stock and Mark W. Watson. Vector autoregressions. *Journal of Economic Perspectives*, 15(4):101–115, December 2001. doi: 10.1257/jep.15.4.101. URL https://www.aeaweb.org/articles?id=10.1257/jep.15.4.101.

Ståle Størdal, Christian-Oliver Ewald, Gudbrand Lien, and Erik Haugom. Trading time seasonality in electricity futures. *Journal of Commodity Markets*, page 100291, 2022. ISSN 2405-8513. doi: https://doi.org/10.1016/j.jcomm.2022.100291. URL https://www.sciencedirect.com/science/article/pii/S2405851322000484.

Farhad Taghizadeh-Hesary, Ehsan Rasoulinezhad, and Naoyuki Yoshino. Energy and food security: Linkages through price volatility. *Energy Policy*, 128: 796–806, 2019. ISSN 0301-4215.

Hiro Y. Toda and Taku Yamamoto. Statistical inference in vector autoregressions with possibly integrated processes. *Journal of Econometrics*, 66(1): 225–250, 1995.

Andreas Wagner, Enislay Ramentol, Florian Schirra, and Hendrik Michaeli. Short- and long-term forecasting of electricity prices using embedding of calendar information in neural networks. *Journal of Commodity Markets*, 28:100246, 2022. ISSN 2405-8513. doi: https://doi.org/10.1016/j.jcomm.2022.100246. URL https://www.sciencedirect.com/science/article/pii/S2405851322000046.

Hiroki Wakamatsu. The impact of msc certification on a japanese certified fishery. *Marine Resource Economics*, 29(1):55–67, 2014. doi: 10.1086/676287.

Dabin Wang and William G. Tomek. Commodity prices and unit root tests. *American Journal of Agricultural Economics*, 89(4):873–889, 2007. ISSN 00029092, 14678276. URL http://www.jstor.org/stable/4492867.

Earo Wang, Dianne Cook, and Rob J Hyndman. A new tidy data structure to support exploration and modeling of temporal data. *Journal of Computational and Graphical Statistics*, 29(3):466–478, 2020. doi: 10.1080/10618600.2019.1695624. URL https://doi.org/10.1080/10618600.2019.1695624.

Yudong Wang, Chongfeng Wu, and Li Yang. Oil price shocks and agricultural commodity prices. *Energy Economics*, 44:22–35, 2014.

Mark W. Watson. Univariate detrending methods with stochastic trends. *Journal of Monetary Economics*, 18(1):49–75, 1986.

Hadley Wickham, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, Alex Hayes, Lionel Henry, Jim Hester, Max Kuhn, Thomas Lin Pedersen, Evan Miller, Stephan Milton Bache, Kirill Müller, Jeroen Ooms, David Robinson, Dana Paige Seidel, Vitalie Spinu, Kohske Takahashi, Davis Vaughan, Claus Wilke, Kara Woo, and Hiroaki Yutani. Welcome to the tidyverse. *Journal of Open Source Software*, 4(43):1686, 2019. doi: 10.21105/joss.01686.

Diego Winkelried. Unit roots in real primary commodity prices? a meta-analysis of the grilli and yang data set. *Journal of Commodity Markets*, 23:100168, 2021. ISSN 2405-8513. doi: https://doi.org/10.1016/j.jcomm.2021.100168. URL https://www.sciencedirect.com/science/article/pii/S2405851321000027.

Tun-hsiang Yu, David Bessler, and Stephen W. Fuller. Cointegration and causality analysis of world vegetable oil and crude oil prices. 2006 Annual meeting, July 23-26, Long Beach, CA 21439, American Agricultural Economics Association (New Name 2008: Agricultural and Applied Economics Association), 2006.

Achim Zeileis and Gabor Grothendieck. zoo: S3 infrastructure for regular and irregular time series. *Journal of Statistical Software*, 14(6):1–27, 2005. doi: 10.18637/jss.v014.i06.

Zibin Zhang, Luanne Lohr, Cesar Escalante, and Michael Wetzstein. Food versus fuel: What do prices tell us? *Energy Policy*, 38(1):445–451, 2010.

Eric Zivot and Donald W. K. Andrews. Further evidence on the great crash, the oil-price shock, and the unit-root hypothesis. *Journal of Business and Economic Statistics*, 10(3):251–270, 1992. ISSN 07350015. URL http://www.jstor.org/stable/1391541.