Chapter 9
Time Series Analysis

Most of the methods described so far in this book are oriented primarily at cross-sectional analysis, or the study of a sample of data taken at the same point in time. In this chapter, we turn to methods for modeling a time series, or a variable that is observed sequentially at regular intervals over time (e.g., daily, weekly, monthly, quarterly, or annually). Time series data frequently have trends and complex error processes, so failing to account for these features can produce spurious results (Granger and Newbold 1974). Several approaches for time series analysis have emerged to address these problems and prevent false inferences. Within Political Science, scholars of public opinion, political economy, international conflict, and several other subjects regularly work with time-referenced data, so adequate tools for time series analysis are important in political analysis.

Many researchers do not think of R as a program for time series analysis, instead using specialty software such as Autobox, EViews, or RATS. Even SAS and Stata tend to get more attention for time series analysis than R does. However, R actually has a wide array of commands for time series models, particularly through the `TSA` and `vars` packages. In this chapter, we will illustrate three approaches to time series analysis in R: the Box–Jenkins approach, extensions to linear models estimated with least squares, and vector autoregression. This is not an exhaustive list of the tools available for studying time series, but is just meant to introduce a few prominent methods. See Sect. 9.4 for some further reading on time series.

Both the Box–Jenkins approach and extensions to linear models are examples of single equation time series models. Both approaches treat one time series as an outcome variable and fit a model of that outcome that can be stated in one equation, much like the regression models of previous chapters can be stated in a single equation. Since both approaches fall into this broad category, the working

Electronic supplementary material: The online version of this chapter (doi: 10.1007/978-3-319-23446-5_9) contains supplementary material, which is available to authorized users.

J.E. Monogan III, *Political Analysis Using R*, Use R!,
DOI 10.1007/978-3-319-23446-5_9

dataset we use for both the Box–Jenkins approach and extensions to linear models will be Peake and Eshbaugh-Soha's (2008) monthly data on television coverage of energy policy that was first introduced in Chap. 3. By contrast, vector autoregression is a multiple equation time series model (for further details on this distinction, see Brandt and Williams 2007 or Lütkepohl 2005). With a vector autoregression model, two or more time series are considered endogenous, so multiple equations are required to fully specify the model. This is important because endogenous variables may affect each other, and to interpret an input variable's effect, the broader context of the full system must be considered. Since multiple equation models have such a different specification, when discussing vector autoregression the working example will be Brandt and Freeman's (2006) analysis of weekly political actions in the Israeli–Palestinian conflict; more details will be raised once we get to that section.

9.1 The Box–Jenkins Method

The Box–Jenkins approach to time series relies on autoregressive integrated moving average (ARIMA) models to capture the error process in the data. For a comprehensive explanation of this approach, see Box et al. (2008). The basic logic of this approach is that a time series should be filtered, or *prewhitened*, of any trends and error processes before attempting to fit an inferential model. Once a model of the *noise*, or error, has been specified, then the researcher can proceed to test hypotheses.[1] On account of nonlinear functional forms that often emerge in ARIMA and transfer function models, these usually are estimated with maximum likelihood.

To illustrate the process from identifying an error model to testing a hypothesis, we revisit Peake and Eshbaugh-Soha's (2008) monthly time series on energy policy coverage. We start by reloading our data:[2]

```
pres.energy<-read.csv("PESenergy.csv")
```

Our outcome variable is the number of energy-related stories on nightly television news by month (**Energy**). A good first step is to plot the series being studied to see if any trends or other features are immediately apparent. Look back at Fig. 3.6 for the example code we used to plot our outcome variable (**Energy**) and the price of oil, which is one of the predictors (**oilc**). As a first look at the news coverage series, the data appear to be *stationary*, meaning that they hover around a mean without *trending* in one direction or showing an *integrated* pattern.

As a substantively important point about time series in general, the distinction between *stationary* and *nonstationary* series is important. Many time series methods

[1]Many use ARIMA models for forecasting future values of a series. ARIMA models themselves are atheoretical, but often can be effective for prediction. Since most Political Science work involves testing theoretically motivated hypotheses, this section focuses more on the role ARIMA models can serve to set up inferential models.

[2]If you do not have the data file `PESenergy.csv` already, you can download it from the Dataverse (see page vii) or the online chapter content (see page 155).

are designed specifically for modeling stationary series, so applying them to a nonstationary series can be problematic. A *stationary* series is one for which the mean and variance do not change conditional on time, and the series does not trend in one direction. If a stationary series is disturbed, or moved to a higher or lower value, it eventually returns back to an equilibrium level. The two kinds of *nonstationary* series are trending series and integrated series. With a *trending* series, as its name implies, the conditional average changes over time, typically rising or falling consistently. For example, nominal prices of consumer goods may rise or fall from one quarter to the next, but the average value tends to rise over time. An *integrated series*, also called a unit root series, does not have an equilibrium value and has the feature of "long memory." This means that if something changes the value of a series, then future values of the series will be affected by that change long into the future. For instance, suppose the Dow Jones Industrial Average dropped 300 points in 1 day, from 18,200 to 17,900. In such a case, future values of the Dow will be based on the new value of 17,900 plus however much the value rose or fell each subsequent day, and the Dow will not tend to revert back to earlier values. Importantly, by *differencing* the current value of a series from its previous value, or measuring how much a series changed from one time to the next, trending or integrated series can be made stationary. Hence, many models for stationary series can be applied to the difference of a nonstationary series. Whenever doing applied time series work always look at the graph of the series, such as Fig. 3.6 in this case, as well as the diagnostics to be described to determine whether you are studying a stationary, trending, or unit root series.[3]

As a second step, we turn to diagnostic measures and look at the **a**uto**c**orrelation **f**unction (ACF) and **p**artial **a**uto**c**orrelation **f**unction (PACF). The autocorrelation function shows how much current values of a series correlate with previous values at a certain *lag*. *Lags* are important for time series modeling, as will be seen later in the chapter, and they act by changing the index of a series to shift time back and form a new series. For example, a first-order lag is a new series where each observation is the value that occurred one time period previously. A second-order lag is a new series where each observation is the value from two time periods previously, and so forth. This is important when using the autocorrelation function because, if we are studying a series y with time index t, ACF(1) would give us the correlation between y_t and y_{t-1}. (For energy coverage, then, ACF(1) tells us the correlation between current months' coverage with prior months' coverage.) Subsequently, ACF(2) would be the autocorrelation between y_t and y_{t-2}, and more generally ACF(p) is the autocorrelation between y_t and y_{t-p}. The PACF provides the autocorrelation between current and lagged values *that is not accounted for by all prior lags*. So for the first lag, ACF(1)=PACF(1), but for all subsequent lags only the autocorrelation unique to that lag shows up in the PACF. If we wanted to know the ACF and PACF of our energy policy series, we would type:

[3]In addition to examining the original series or the autocorrelation function, an Augmented Dickey–Fuller test also serves to diagnose whether a time series has a unit root. By loading the `tseries` package, the command `adf.test` will conduct this test in R.

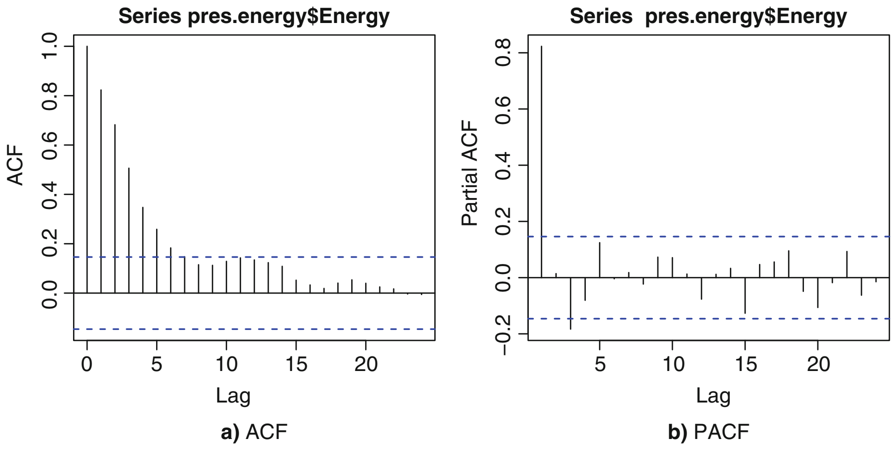

Fig. 9.1 Autocorrelation function and partial autocorrelation function of monthly TV coverage of energy policy through 24 lags. (**a**) ACF. (**b**) PACF

```
acf(pres.energy$Energy,lag.max=24)
pacf(pres.energy$Energy,lag.max=24)
```

The `acf` and `pacf` functions are available without loading a package, though the code changes slightly if users load the `TSA` package.[4] Notice that within the `acf` and `pacf` functions, we first list the series we are diagnosing. Second, we designate `lag.max`, which is the number of lags of autocorrelation we wish to consider. Since these are monthly data, 24 lags gives us 2 years' worth of lags. In some series, *seasonality* will emerge, in which we see evidence of similar values at the same time each year. This would be seen with significant lags around 12 and 24 with monthly data (or around 4, 8, and 12, by contrast, with quarterly data). No such evidence appears in this case.

The graphs of our ACF and PACF are shown in Fig. 9.1. In each of these figures, the horizontal axis represents the lag length, and the vertical axis represents the correlation (or partial correlation) value. At each lag, the autocorrelation for that lag is shown with a solid histogram-like line from zero to the value of the correlation. The blue dashed lines represent the threshold for a significant correlation. Specifically, the blue bands represent the 95 % confidence interval based on an uncorrelated series.[5] All of this means that if the histogram-like line does not

[4]The primary noticeable change is that the default version of `acf` graphs the zero-lag correlation, ACF(0), which is always 1.0. The `TSA` version eliminates this and starts with the first lag autocorrelation, ACF(1).

[5]The formula for these error bands is: $0 \pm 1.96 \times se_r$. The standard error for a correlation coefficient is: $se_r = \sqrt{\frac{1-r^2}{n-2}}$. So in this case, we set $r = 0$ under the null hypothesis, and n is the sample size (or series length).

cross the dashed line, the level of autocorrelation is not discernibly different from zero, but correlation spikes outside of the error band are statistically significant.

The ACF, in Fig. 9.1a starts by showing the zero-lag autocorrelation, which is just how much current values correlate with themselves—always exactly 1.0. Afterward, we see the more informative values of autocorrelation at each lag. At one lag out, for instance, the serial correlation is 0.823. For two lags, it drops to 0.682. The sixth lag is the last lag to show discernible autocorrelation, so we can say that the ACF decays rapidly. The PACF, in Fig. 9.1b, skips the zero lag and starts with first-order serial correlation. As expected, this is 0.823, just like the ACF showed. However, once we account for first-order serial correlation, the partial autocorrelation terms at later lags are not statistically discernible.[6]

At this point, we determine which ARIMA error process would leave an empirical footprint such as the one this ACF and PACF show. For more details on common footprints, see Enders (2009, p. 68). Notationally, we call the error process ARIMA(p,d,q), where p is the number of autoregressive terms, d is how many times the series needs to be differenced, and q is the number of moving average terms. Functionally, a general *ARMA* model, which includes autoregressive and moving average components, is written as follows:

$$y_t = \alpha_0 + \sum_{i=1}^{p} \alpha_i y_{t-i} + \sum_{i=1}^{q} \beta_i \epsilon_{t-i} + \epsilon_t \tag{9.1}$$

Here, y_t is our series of interest, and this ARMA model becomes an ARIMA model when we decide whether we need to difference y_t or not. Notice that y_t is lagged p times for the autoregressive terms, and the disturbance term (ϵ_t) is lagged q times for the moving average terms. In the case of energy policy coverage, the ACF shows a rapid decay and we see one significant spike in the PACF, so we can say we are dealing with a first-order autoregressive process, denoted AR(1) or ARIMA(1,0,0).

Once we have identified our **a**uto**r**egressive **i**ntegrated **m**oving **a**verage model, we can estimate it using the `arima` function:

```
ar1.mod<-arima(pres.energy$Energy,order=c(1,0,0))
```

The first input is the series we are modeling, and the `order` option allows us to specify p, d, and q (in order) for our ARIMA(p,d,q) process. By typing `ar1.mod`, we see the output of our results:

```
Call:
arima(x = pres.energy$Energy, order = c(1, 0, 0))

Coefficients:
         ar1  intercept
      0.8235    32.9020
s.e.  0.0416     9.2403
```

[6]Technically, PACF at the third lag is negative and significant, but the common patterns of error processes suggest that this is unlikely to be a critical part of the ARIMA process.

```
sigma^2 estimated as 502.7: log likelihood=-815.77,
 aic=1637.55
```

Pretty simply, this shows us the estimate and standard error of the autoregressive coefficient (`ar1`) and intercept, as well as the residual variance, log likelihood, and AIC.

The next step in the Box–Jenkins modeling process is to diagnose whether the estimated model sufficiently filters the data.[7] We do this in two ways: First, by studying the ACF and PACF for the residuals from the ARIMA model. The code is:

```
acf(ar1.mod$residuals,lag.max=24)
pacf(ar1.mod$residuals,lag.max=24)
```

As with many other models, we can call our residuals using the model name and a dollar sign (`ar1.mod$residuals`). The resulting graphs are presented in Fig. 9.2. As the ACF and PACF both show, the second and fourth lags barely cross the significance threshold, but there is no clear pattern or evidence of an overlooked feature of the error process. Most (but not all) analysts would be content with this pattern in these figures.

As a second step of diagnosing whether we have sufficiently filtered the data, we compute the Ljung–**Box** *Q*-**test**. This is a joint test across several lags of whether there is evidence of serial correlation in any of the lags. The null hypothesis is that

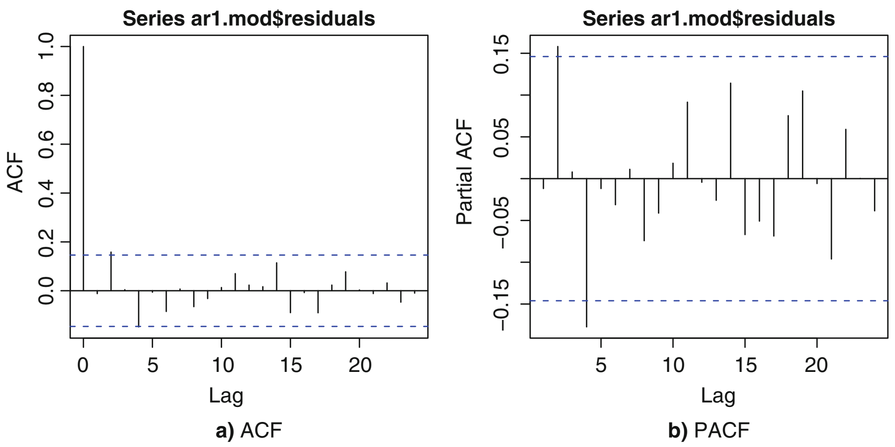

Fig. 9.2 Autocorrelation function and partial autocorrelation function for residuals of AR(1) model through 24 lags. *Blue dashed lines* represent a 95 % confidence interval for an uncorrelated series. (**a**) ACF. (**b**) PACF (Color figure online)

[7]Here we show in the main text how to gather one diagnostic at a time, but the reader also may want to try typing `tsdiag(ar1.mod,24)` to gather graphical representations of a few diagnostics all at once.

the data are independent, so a significant result serves as evidence of a problem. The syntax for this test is:

```
Box.test(ar1.mod$residuals,lag=24,type="Ljung-Box")
```

We first specify the series of interest, then with the `lag` option state how many lags should go into the test, and lastly with `type` specify `Ljung-Box` (as opposed to the Box–Pierce test, which does not perform as well). Our results from this test are:

```
        Box-Ljung test

data:  ar1.mod$residuals
X-squared = 20.1121, df = 24, p-value =
0.6904
```

Our test statistic is not significant ($p = 0.6904$), so this test shows no evidence of serial correlation over 2 years' lags. If we are satisfied that AR(1) characterizes our error process, we can proceed to actual modeling in the next section.

9.1.1 Transfer Functions Versus Static Models

To estimate a theoretically motivated model, we need to draw an important distinction between two types of functional forms: On one hand, a *static* functional form assumes that current values of predictors affect current values of outcomes. For example, we may believe that the price of oil in a given month affects coverage of energy issues on television in the same month, while next month's oil prices will affect next month's energy coverage. If we think that this is the only process going on, then we want a static functional form. On the other hand, we may use a *dynamic* functional form. In the case of oil prices, this would mean that this month's oil prices affect this month's energy coverage, and also coverage in the next month to a lesser degree, the month after to a lesser degree still, and so on. Essentially, a dynamic functional form allows spillover effects from each observation of an input.

Estimating a static regression model with an ARIMA error process only requires two steps in R. First, all of the predictor variables must be placed in a matrix. Second, we can simply add one option to our `arima` function:

```
predictors<-as.matrix(subset(pres.energy,select=c(rmn1173,
    grf0175,grf575,jec477,jec1177,jec479,embargo,hostages,
    oilc,Approval,Unemploy)))
static.mod<-arima(pres.energy$Energy, order=c(1,0,0),
    xreg=predictors)
```

In this code, we first subset our original dataset and treat the subset **as** a **matrix** named `predictors`. Second, we use the same `arima` function as we used to estimate our AR(1) noise model, but add the option `xreg=predictors`. This now estimates a model in which temporal error process is corrected for, but we also include theoretically motivated predictors of interest. If we type `static.mod`, the output is:

```
Call:
arima(x=pres.energy$Energy,order=c(1,0,0),xreg=
 predictors)

Coefficients:
           ar1  intercept  rmn1173  grf0175   grf575
        0.8222     5.8822  91.3265  31.8761  -8.2280
s.e.    0.0481    52.9008  15.0884  15.4643  15.2025
         jec477   jec1177    jec479  embargo  hostages
        29.6446   -6.6967  -20.1624  35.3247  -16.5001
s.e.    15.0831   15.0844   15.2238  15.1200   13.7619
           oilc  Approval  Unemploy
         0.8855   -0.2479    1.0080
s.e.     1.0192    0.2816    3.8909

sigma^2 estimated as 379.3: log likelihood=-790.42,
 aic=1608.84
```

This now shows us the estimate and standard error not only for the autoregressive coefficient and intercept, but also for the partial regression coefficient of every other predictor in the model. Again, the three fit measures are reported at the end of the printout.

If we would like to include one or more predictors that have a dynamic effect, then we turn to the method of *transfer functions*, which specify that a predictor has a spillover effect on subsequent observations. A special case of this is called *intervention analysis*, wherein a treatment is coded with an indicator (Box and Tiao 1975). With intervention analysis, a variety of functional forms can emerge, depending largely on whether the indicator variable is coded a *pulse* (taking a value of 1 only at the treatment time, and 0 otherwise) or a *step* (taking on a value of 0 before the treatment and 1 at all subsequent times). It is advisable to try both codings and to read further into the functional form of each (Enders 2009, Sect. 5.1).

As an illustration, we fit a transfer function for a pulse intervention for Richard Nixon's speech in November 1973. To estimate this model, we now need to load the `TSA` package to access the `arimax` function (**ARIMA** with **x** predictors). Remember that you may need to use `install.packages` to download `TSA`:

```
install.packages("TSA")
library(TSA)
dynamic.mod<-arimax(pres.energy$Energy,order=c(1,0,0),
     xreg=predictors[,-1],xtransf=predictors[,1],
     transfer=list(c(1,0)))
```

The syntax to `arimax` is similar to `arima`, but we are now allowed a few more options for transfer functions. Notice in this case that we use the code `xreg=predictors[-1]` to remove the indicator for Nixon's November 1973 speech from the static predictors. We instead place this predictor with the `xtransf` option. The last thing we need to do is specify the order of our transfer function, which we do with the option `transfer`. The `transfer` option accepts a list of vectors, one vector per transfer function predictor. For our one transfer function, we specify `c(1,0)`: The first term refers to the order of the dynamic decay term (so a 0 here actually reverts back to a static model), and the second term refers to the lag length of the predictor's effect (so if we expected an effect to *grow*, we might put a higher number than 0 here). With these settings, we say that Nixon's speech had an effect in the month he gave it, and then the effect spilled over to subsequent months at a decaying rate.

By typing `dynamic.mod`, we get our output:

```
Call:
arimax(x=pres.energy$Energy,order=c(1,0,0),
 xreg=predictors[,-1],
    xtransf = predictors[, 1], transfer = list(c(1, 0)))

Coefficients:
           ar1  intercept  grf0175   grf575   jec477
        0.8262    20.2787  31.5282  -7.9725  29.9820
s.e.    0.0476    46.6870  13.8530  13.6104  13.5013
        jec1177    jec479  embargo  hostages    oilc
        -6.3304  -19.8179  25.9388  -16.9015  0.5927
s.e.    13.5011   13.6345  13.2305   12.4422  0.9205
        Approval  Unemploy  T1-AR1    T1-MA0
         -0.2074    0.1660  0.6087  160.6241
s.e.      0.2495    3.5472  0.0230   17.0388

sigma^2 estimated as 305.1: log likelihood=-770.83,
 aic=1569.66
```

The output is similar to that from the static ARIMA regression model, but now there are two terms for the effect of Nixon's speech. The first, `T1-AR1`, gives the decay term. The closer this term is to 1, the more persistent the variable's effect is. The second term, `T1-MA0`, is the initial effect of the speech on energy coverage in the month it was given.[8] In terms of model fit, notice that the output of every ARIMA or transfer function model we have estimated reports the Akaike information criterion (AIC). With this measure, lower scores indicate a better penalized fit. With a score of 1569.66, this dynamic transfer function model has the lowest AIC of any model

[8]In this case, we have a pulse input, so we can say that in November 1973, the effect of the speech was an expected 161 increase in news stories, holding all else equal. In December 1973, the carryover effect is that we expect 98 more stories, holding all else equal because $161 \times 0.61 \approx 98$. In January 1974, the effect of the intervention is we expect 60 more stories, *ceteris paribus* because $161 \times 0.61 \times 0.61 \approx 60$. The effect of the intervention continues forward in a similar decaying pattern. By contrast, ***if*** we had gotten these results with a *step* intervention instead of a *pulse* intervention, then these effects would accumulate rather than decay. Under this hypothetical, the effects would be 161 in November 1973, 259 in December 1973 (because 161+98=259), and 319 in January 1974 (because 161+98+60=319).

we have fitted to these data. Hence, the dynamic model has a better fit than the static model or the atheoretical AR(1) model with no predictors.

To get a real sense of the effect of an intervention analysis, though, an analyst should always try to draw the effect that they modeled. (Again, it is key to study the functional form behind the chosen intervention specification, as described by Enders 2009, Sect. 5.1.) To draw the effect of our intervention for Nixon's 1973 speech, we type:

```
months<-c(1:180)
y.pred<-dynamic.mod$coef[2:12]%*%c(1,predictors[58,-1])+
     160.6241*predictors[,1]+
     160.6241*(.6087^(months-59))*as.numeric(months>59)
plot(y=pres.energy$Energy,x=months,xlab="Month",
     ylab="Energy Policy Stories",type="l",axes=F)
axis(1,at=c(1,37,73,109,145),labels=c("Jan. 1969",
     "Jan. 1972","Jan. 1975","Jan. 1978","Jan. 1981"))
axis(2)
box()
lines(y=y.pred,x=months,lty=2,col="blue",lwd=2)
```

On the first line, we simply create a time index for the 180 months in the study. In the second line, we create predicted values for the effect of the intervention holding everything else equal. A critical assumption that we make is that we hold all other predictors equal by setting them to their values from October 1973, the 58th month of the series (hence, `predictors[58,-1]`). So considering the components of this second line, the first term multiplies the coefficients for the static predictors by their last values before the intervention, the second term captures the effect of the intervention in the month of the speech, and the third term captures the spillover effect of the intervention based on the number of months since the speech. The next four lines simply draw a plot of our original series' values and manage some of the graph's features. Lastly, we add a dashed line showing the effect of the intervention holding all else constant. The result is shown in Fig. 9.3. As the figure shows, the result of this intervention is a large and positive jump in the expected number of news stories, that carries over for a few months but eventually decays back to pre-intervention levels. This kind of graph is essential for understanding how the dynamic intervention actually affects the model.

As a final graph to supplement our view of the dynamic intervention effect, we could draw a plot that shows how well predictions from the full model align with true values from the series. We could do this with the following code:

```
months<-c(1:180)
full.pred<-pres.energy$Energy-dynamic.mod$residuals
plot(y=full.pred,x=months,xlab="Month",
     ylab="Energy Policy Stories",type="l",
     ylim=c(0,225),axes=F)
points(y=pres.energy$Energy,x=months,pch=20)
legend(x=0,y=200,legend=c("Predicted","True"),
     pch=c(NA,20),lty=c(1,NA))
axis(1,at=c(1,37,73,109,145),labels=c("Jan. 1969",
     "Jan. 1972","Jan. 1975","Jan. 1978","Jan. 1981"))
axis(2)
box()
```

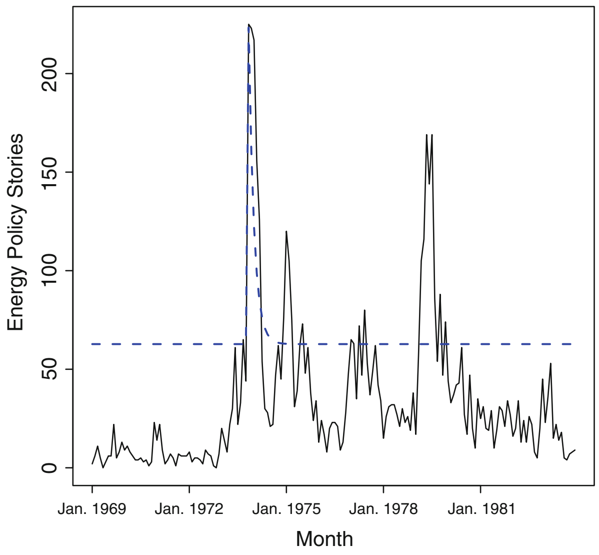

Fig. 9.3 The *dashed line* shows the dynamic effect of Nixon's Nov. 1973 speech, holding all else equal. The *solid line* shows observed values of the series

Again, we start by creating a time index, `months`. In the second line, we create our predicted values by subtracting the residuals from the true values. In the third line of code, we draw a line graph of the predicted values from the model. In the fourth line, we add points showing the true values from the observed series. The remaining lines complete the graph formatting. The resulting graph is shown in Fig. 9.4. As can be seen, the in-sample fit is good, with the predicted values tracking the true values closely.

As a final point here, the reader is encouraged to consult the code in Sect. 9.5 for alternative syntax for producing Figs. 9.3 and 9.4. The tradeoff of the alternative way of drawing these figures is that it requires more lines of code on the one hand, but on the other hand, it is more generalizable and easier to apply to your own research. Plus, the alternative code introduces how the `ts` command lets analysts convert a variable to a *time series object*. Seeing both approaches is worthwhile for illustrating that, in general, many tasks can be performed in many ways in R.

9.2 Extensions to Least Squares Linear Regression Models

A second approach to time series analysis draws more from the econometric literature and looks for ways to extend linear regression models to account for the unique issues associated with time-referenced data. Since we already discussed visualization with these data extensively in Sect. 9.1, we will not revisit graphing issues here. As with Box–Jenkins type models, though, the analyst should always

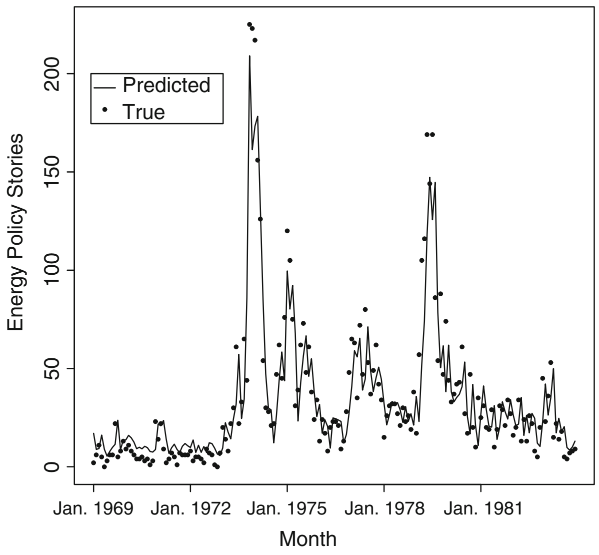

Fig. 9.4 Predicted values from a full transfer function model on a line, with actual observed values as points

begin by drawing a line plot of the series of interest, and ideally a few key predictors as well. Even diagnostic plots such as the ACF and PACF would be appropriate, in addition to residual diagnostics such as we will discuss shortly.

When modeling data from an econometric approach, researchers again have to decide whether to use a static or a dynamic specification of the model. For static models in which current values of predictors affect current values of the outcome, researchers may estimate the model with ordinary least squares (OLS) in the *rare* case of no serial correlation. For efficiency gains on a static model, however, feasible generalized least squares (FGLS) is a better estimator. By contrast, in the case of a dynamic functional form, a lag structure can be introduced into the linear model's specification.

Starting with static models, the simplest kind of model (though rarely appropriate) would be to estimate the model using simple OLS. Returning to our energy policy data, our model specification here would be:

```
static.ols<-lm(Energy~rmn1173+grf0175+grf575+jec477+
     jec1177+jec479+embargo+hostages+oilc+
     Approval+Unemploy,data=pres.energy)
```

By typing `summary(static.ols)` we get our familiar output from a linear regression model. Beware, though, that if there is serial correlation in the disturbances, these estimates of the standard errors are incorrect:

```
Call:
lm(formula=Energy~rmn1173+grf0175+grf575+jec477+
  jec1177+
    jec479 + embargo + hostages + oilc + Approval +
      Unemploy,
    data = pres.energy)

Residuals:
     Min        1Q    Median        3Q       Max
-104.995  -12.921   -3.448     8.973   111.744

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 319.7442    46.8358   6.827 1.51e-10 ***
rmn1173      78.8261    28.8012   2.737  0.00687 **
grf0175      60.7905    26.7006   2.277  0.02406 *
grf575       -4.2676    26.5315  -0.161  0.87240
jec477       47.0388    26.6760   1.763  0.07966 .
jec1177      15.4427    26.3786   0.585  0.55905
jec479       72.0519    26.5027   2.719  0.00724 **
embargo      96.3760    13.3105   7.241 1.53e-11 ***
hostages     -4.5289     7.3945  -0.612  0.54106
oilc         -5.8765     1.0848  -5.417 2.07e-07 ***
Approval     -1.0693     0.2147  -4.980 1.57e-06 ***
Unemploy     -3.7018     1.3861  -2.671  0.00831 **
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 26.26 on 168 degrees of
 freedom
Multiple R-squared:  0.5923,  Adjusted R-squared:
 0.5656
F-statistic: 22.19 on 11 and 168 DF,  p-value:
 < 2.2e-16
```

Yet, we need to diagnose whether serial correlation is present in the residuals before we are content with the results. To do this, we need to load the `lmtest` package (first introduced in Chap. 6) to make a few diagnostics available. Loading this package, we can then compute a Durbin–Watson or Breusch–Godfrey test for autocorrelation:

```
library(lmtest)
dwtest(static.ols)
bgtest(static.ols)
```

For both the `dwtest` and `bgtest` commands, we simply provide the name of the model as the main argument. The **D**urbin-**W**atson **test** (computed with `dwtest`) tests for first-order serial correlation and is not valid for a model that includes a lagged dependent variable. Our test produces the following output:

```
          Durbin-Watson test

data:  static.ols
DW = 1.1649, p-value = 1.313e-09
alternative hypothesis: true autocorrelation is greater
   than 0
```

The Durbin–Watson *d* statistic (1.1649 in this case), does not have a parametric distribution. Traditionally the value of *d* has been checked against tables based on Monte Carlo results to determine significance. R, however, does provide an approximate *p*-value with the statistic. For a Durbin–Watson test, the null hypothesis is that there is no autocorrelation, so our significant value of *d* suggests that autocorrelation is a problem.

Meanwhile, the results of our **B**reusch-**G**odfrey **test** (computed with `bgtest`) offer a similar conclusion. The Breusch–Godfrey test has a χ^2 distribution and can be used to test autocorrelation in a model with a lagged dependent variable. By default, the `bgtest` command checks for first-order serial correlation, though higher-order serial correlation can be tested with the `order` option. Our output in this case is:

```
Breusch-Godfrey test for serial correlation of
   order up to
1

data:  static.ols
LM test = 38.6394, df = 1, p-value = 5.098e-10
```

Again, the null hypothesis is that there is no autocorrelation, so our significant χ^2 value shows that serial correlation is a concern, and we need to do something to account for this.

At this point, we can draw one of two conclusions: The first possibility is that our *static model specification* is correct, and we need to find an estimator that is efficient in the presence of error autocorrelation. The second possibility is that we have overlooked a dynamic effect and need to respecify our model. (In other words, if there is a true spillover effect, and we have not modeled it, then the errors will appear to be serially correlated.) We will consider each possibility.

First, if we are confident that our static specification is correct, then our functional form is right, but under the Gauss–Markov theorem OLS is inefficient with error autocorrelation, and the standard errors are biased. As an alternative, we can use feasible generalized least squares (FGLS), which estimates the level of error correlation and incorporates this into the estimator. There are a variety of estimation techniques here, including the Prais–Winsten and Cochrane–Orcutt estimators. We proceed by illustrating the Cochrane–Orcutt estimator, though users should be wary of the model's assumptions.[9] In short, **Cochrane–Orcutt** reestimates the model several times, updating the estimate of error autocorrelation each time, until it converges to a stable estimate of the correlation. To implement this procedure in R, we need to install and then load the `orcutt` package:

```
install.packages("orcutt")
library(orcutt)
cochrane.orcutt(static.ols)
```

Once we have loaded this package, we insert the name of a linear model we have estimated with OLS into the `cochrane.orcutt` command. This then iteratively reestimates the model and produces our FGLS results as follows:

```
$Cochrane.Orcutt

Call:
lm(formula = YB ~ XB - 1)

Residuals:
    Min       1Q   Median       3Q      Max
-58.404   -9.352   -3.658    8.451  100.524

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
XB(Intercept)  16.8306    55.2297   0.305   0.7609
XBrmn1173      91.3691    15.6119   5.853  2.5e-08 ***
XBgrf0175      32.2003    16.0153   2.011   0.0460 *
XBgrf575       -7.9916    15.7288  -0.508   0.6121
XBjec477       29.6881    15.6159   1.901   0.0590 .
XBjec1177      -6.4608    15.6174  -0.414   0.6796
XBjec479      -20.0677    15.6705  -1.281   0.2021
XBembargo      34.5797    15.0877   2.292   0.0232 *
XBhostages    -16.9183    14.1135  -1.199   0.2323
XBoilc          0.8240     1.0328   0.798   0.4261
XBApproval     -0.2399     0.2742  -0.875   0.3829
XBUnemploy     -0.1332     4.3786  -0.030   0.9758
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 20.19 on 167 degrees of
 freedom
```

[9] In particular, at each stage of the iterative process, the linear model is estimated by regressing $y_t^* = y_t - \rho y_{t-1}$ on $\mathbf{x}_t^* = \mathbf{x}_t - \rho \mathbf{x}_{t-1}$ (Hamilton 1994, p. 223). This procedure assumes that the dynamic adjustment process is the same for the outcome and the input variables, which is unlikely. Hence, a dynamic specification such as an autoregressive distributive lag model would be more flexible.

```
Multiple R-squared:  0.2966,    Adjusted R-squared:
   0.2461
F-statistic:  5.87 on 12 and 167 DF,  p-value:
 1.858e-08


$rho
[1] 0.8247688

$number.interaction
[1] 15
```

The first portion of table looks like the familiar linear regression output (though the letters `XB` appear before the name of each predictor). All of these coefficients, standard errors, and inferential statistics have the exact same interpretation as in a model estimated with OLS, but our estimates now should be efficient because they were computed with FGLS. Near the bottom of the output, we see `$rho`, which shows us our final estimate of error autocorrelation. We also see `$number.interaction`, which informs us that the model was reestimated in 15 iterations before it converged to the final result. FGLS is intended to produce efficient estimates if a static specification is correct.

By contrast, if we believe a *dynamic specification* is correct, we need to work to respecify our linear model to capture that functional form. In fact, if we get the functional form wrong, our results are biased, so getting this right is critical. Adding a lag specification to our model can be made considerably easier if we install and load the `dyn` package. We name our model `koyck.ols` for reasons that will be apparent shortly:

```
install.packages("dyn")
library(dyn)
pres.energy<-ts(pres.energy)
koyck.ols<-dyn$lm(Energy~lag(Energy,-1)+rmn1173+
     grf0175+grf575+jec477+jec1177+jec479+embargo+
     hostages+oilc+Approval+Unemploy,data=pres.energy)
```

After loading `dyn`, the second line uses the `ts` command to declare that our data are time series data. In the third line, notice that we changed the linear model command to read, `dyn$lm`. This modification allows us to include lagged variables within our model. In particular, we now have added `lag(Energy,-1)`, which is the lagged value of our dependent variable. With the `lag` command, we specify the variable being lagged and how many times to lag it. By specifying `-1`, we are looking at the immediately prior value. (Positive values represent future values.) The default lag is 0, which just returns current values.

We can see the results of this model by typing `summary(koyck.ols)`:

```
Call:
lm(formula = dyn(Energy ~ lag(Energy, -1) + rmn1173 +
   grf0175 +
   grf575 + jec477 + jec1177 + jec479 + embargo +
     hostages +
    oilc + Approval + Unemploy), data = pres.energy)

Residuals:
    Min      1Q  Median      3Q     Max
-51.282  -8.638  -1.825   7.085  70.472

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)      62.11485   36.96818   1.680  0.09479 .
lag(Energy, -1)   0.73923    0.05113  14.458  < 2e-16 ***
rmn1173         171.62701   20.14847   8.518 9.39e-15 ***
grf0175          51.70224   17.72677   2.917  0.00403 **
grf575            7.05534   17.61928   0.400  0.68935
jec477           39.01949   17.70976   2.203  0.02895 *
jec1177         -10.78300   17.59184  -0.613  0.54075
jec479           28.68463   17.83063   1.609  0.10958
embargo          10.54061   10.61288   0.993  0.32206
hostages         -2.51412    4.91156  -0.512  0.60942
oilc             -1.14171    0.81415  -1.402  0.16268
Approval         -0.15438    0.15566  -0.992  0.32278
Unemploy         -0.88655    0.96781  -0.916  0.36098
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 17.42 on 166 degrees of
 freedom
  (2 observations deleted due to missingness)
Multiple R-squared:  0.822,     Adjusted R-squared:
 0.8092
F-statistic: 63.89 on 12 and 166 DF,  p-value:
 < 2.2e-16
```

Our specification here is often called a Koyck model. This is because Koyck (1954) observed that when a lagged dependent variable is included as a predictor, each predictor will have spillover effects in subsequent months.

Consider two examples of predictor spillover effects. First, our coefficient for Nixon's speech is approximately 172. Here, we are interested in an *impulse effect* whereby the predictor increased to 1 in the month the speech was given, and then went back to 0. Therefore, in the month of November 1973 when the speech was given, the expected effect of this speech holding all else equal is a 172 story increase in energy policy coverage. However, in December 1973, November's level of coverage is a predictor, and November's coverage was shaped by the speech. Since our coefficient on the lagged dependent variable is approximately 0.74, and since $0.74 \times 172 \approx 128$, we therefore expect that the speech increased energy coverage in December by 128, *ceteris paribus*. Yet the effect would persist into January as well because December's value predicts January 1974s value. Since

$0.74 \times 0.74 \times 172 \approx 94$, we expect the effect of Nixon's speech to be a 94 story increase in energy coverage in January, *ceteris paribus*. This kind of decaying dynamic effect persists for an impulse effect in any of these variables, so this is a powerful way to specify a dynamic model. Also, bear in mind that a researcher could easily plot these decaying intervention effects over time to visualize the dynamic impact of an intervention. Such a graph is valid for a Koyck model and would resemble the output of Fig. 9.3.

Second, consider the effect of unemployment. We should not make much of this predictor's impact because the coefficient is not discernible, but it does serve as an example of interpreting a continuous predictor's dynamic effect. If we are interested in a *step effect*, we would like to know what the long-run impact would be if a predictor increased by a single unit and stayed at that higher level. So although it is not statistically discernible, the coefficient for unemployment is −0.89, meaning that a percentage point increase in unemployment decreases news attention to energy policy by nearly nine-tenths of a story, in the same month, on average, and *ceteris paribus*. But if unemployment stayed a percentage point higher, how would coverage change in the long run? If β_{13} is the coefficient on unemployment and β_2 is the coefficient on the lagged dependent variable, then the long-term effect is computed by (Keele and Kelly 2006, p. 189):

$$\frac{\beta_{13}}{1-\beta_2} \tag{9.2}$$

We can compute this in R simply by referencing our coefficients:

```
koyck.ols$coefficients[13]/(1-koyck.ols$coefficients[2])
```

Our output is −3.399746, which means that a persistent 1 % point rise in unemployment would reduce TV news coverage of energy policy in the long term by 3.4 stories on average and all else equal. Again, this kind of long-term effect could occur for any variable that is not limited to a pulse input.

As a final strategy, we could include one or more lags of one or more predictors without including a lagged dependent variable. In this case, any spillover will be limited to whatever we directly incorporate into the model specification. For example, if we only wanted a dynamic effect of Nixon's speech and a static specification for everything else, we could specify this model:

```
udl.mod<-dyn$lm(Energy~rmn1173+lag(rmn1173,-1)+
     lag(rmn1173,-2)+lag(rmn1173,-3)+lag(rmn1173,-4)+
     grf0175+grf575+jec477+jec1177+jec479+embargo+
     hostages+oilc+Approval+Unemploy,data=pres.energy)
```

In this situation, we have included the current value of the Nixon's speech indictor, as well as four lags. For an intervention, that means that this predictor will have an effect in November 1973 and for 4 months afterwards. (In April 1974, however, the effect abruptly drops to 0, where it stays.) We see the results of this model by typing `summary(udl.mod)`:

```
Call:
lm(formula=dyn(Energy~rmn1173+lag(rmn1173,-1)+lag
    (rmn1173,-2)+lag(rmn1173,-3)+lag(rmn1173,-4)
    +grf0175+grf575+jec477+jec1177+jec479+embargo
    +hostages+oilc+Approval+Unemploy),data=pres.energy)

Residuals:
    Min      1Q  Median      3Q     Max
-43.654 -13.236  -2.931   7.033 111.035

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)      334.9988    44.2887   7.564 2.89e-12 ***
rmn1173          184.3602    34.1463   5.399 2.38e-07 ***
lag(rmn1173, -1) 181.1571    34.1308   5.308 3.65e-07 ***
lag(rmn1173, -2) 154.0519    34.2151   4.502 1.29e-05 ***
lag(rmn1173, -3) 115.6949    34.1447   3.388 0.000885 ***
lag(rmn1173, -4)  75.1312    34.1391   2.201 0.029187 *
grf0175           60.5376    24.5440   2.466 0.014699 *
grf575            -3.4512    24.3845  -0.142 0.887629
jec477            45.5446    24.5256   1.857 0.065146 .
jec1177           14.5728    24.2440   0.601 0.548633
jec479            71.0933    24.3605   2.918 0.004026 **
embargo           -9.7692    24.7696  -0.394 0.693808
hostages          -4.8323     6.8007  -0.711 0.478392
oilc              -6.1930     1.0232  -6.053 9.78e-09 ***
Approval          -1.0341     0.1983  -5.216 5.58e-07 ***
Unemploy          -4.4445     1.3326  -3.335 0.001060 **
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 24.13 on 160 degrees of freedom
  (8 observations deleted due to missingness)
Multiple R-squared:  0.6683,    Adjusted R-squared:   0.6372
F-statistic: 21.49 on 15 and 160 DF,  p-value: < 2.2e-16
```

As we would expect, the effect shrinks every month after the onset. For this unrestricted distributed lag model, we have to estimate several more parameters than the Koyck model requires, but in some cases it may make sense theoretically.

9.3 Vector Autoregression

The final approach that we will describe in this chapter is vector autoregression (VAR). The VAR approach is useful when studying several variables that are endogenous to each other because there is reciprocal causation among them. The basic framework is to estimate a linear regression model for each of the endogenous variables. In each linear model, include several lagged values of the outcome variable itself (say p lags of the variable) as well as p lags of all of the other endogenous variables. So for the simple case of two endogenous variables, x and y, in which we set our lag length to $p = 3$, we would estimate two equations that could be represented as follows:

$$\begin{aligned} y_t &= \beta_0 + \beta_1 y_{t-1} + \beta_2 y_{t-2} + \beta_3 y_{t-3} + \beta_4 x_{t-1} + \beta_5 x_{t-2} + \beta_6 x_{t-3} + \epsilon_t \quad (9.3) \\ x_t &= \gamma_0 + \gamma_1 x_{t-1} + \gamma_2 x_{t-2} + \gamma_3 x_{t-3} + \gamma_4 y_{t-1} + \gamma_5 y_{t-2} + \gamma_6 y_{t-3} + \delta_t \end{aligned}$$

A fuller treatment on this methodology, including notation for models with more endogenous variables and examples of models that include exogenous variables as well, can be found in Brandt and Williams (2007). With this model, tests such as Granger causality tests can be used to assess if there is a causal effect from one endogenous variable to another (Granger 1969).

As an example of how to implement a VAR model in R, we turn to work by Brandt and Freeman (2006), who analyze weekly data regarding the Israeli–Palestinian conflict. Their data are drawn from the Kansas Event Data System, which automatically codes English-language news reports to measure political events, with a goal of using this information as an early warning to predict political change. The endogenous variables in these data are scaled political actions taken by either the USA, Israel, or Palestine, and directed to one of the other actors. This produces six variables **a2i**, **a2p**, **i2a**, **p2a**, **i2p**, and **p2i**. The abbreviations are "a" for American, "i" for Israeli, and "p" for Palestinian. As an example, **i2p** measures the scaled value of Israeli actions directed toward the Palestinians. The weekly data we will use run from April 15, 1979 to October 26, 2003, for a total of 1278 weeks.

To proceed, we need to install and load the `vars` package to make the relevant estimation commands available. We also need the `foreign` package because our data are in Stata format:[10]

```
install.packages("vars")
library(vars)
library(foreign)
levant.0 <- read.dta("levant.dta")
levant<- subset(levant.0,
     select=c("a2i","a2p","i2p","i2a","p2i","p2a"))
```

After loading the packages, we load our data on the third line, naming it `levant.0`. These data also contain three date-related indices, so for analysis purposes we actually need to create a second copy of the data that only includes our six endogenous variables without the indices. We do this using the `subset` command to create the dataset named `levant`.

A key step at this point is to choose the appropriate lag length, p. The lag length needs to capture all error processes and causal dynamics. One approach to

[10]This example requires the file `levant.dta`. Please download this file from the Dataverse (see page vii) or this chapter's online content (see page 155).

determining the appropriate lag length is to fit several models, choose the model with the best fit, and then see if the residuals show any evidence of serial correlation. The command `VARselect` automatically estimates several **v**ector **au**to**r**egression models and **select**s which lag length has the best fit. Since our data are weekly, we consider up to 104 weeks' lags, or 2 years' worth of data, to consider all possible options. The following command can take a minute to run because 104 models are being estimated:

```
levant.select<-VARselect(levant,type="const",lag.max=104)
```

To find out which of the lag lengths fits best, we can type `levant.select$ selection`. Our output is simply the following:

```
AIC(n)  HQ(n)  SC(n) FPE(n)
    60      4      1     47
```

This reports the chosen lag length for the Akaike information criterion, Hannan–Quinn information criterion, Schwarz criterion, and forecast prediction error. All four of these indices are coded so that lower values are better. To contrast extremes, the lowest value of the Schwarz criterion, which has a heavy penalty for additional parameters, is for the model with only one lag of the endogenous variables. By contrast, the best fit on the AIC comes from the model that requires 60 lags—perhaps indicating that annual seasonality is present. A much longer printout giving the value of each of the four fit indices for all 104 models can be seen by typing: `levant.select$criteria`. Ideally, our fit indices would have settled on models with similar lag lengths. Since they did not, and since we have 1278 observations, we will take the safer route with the long lag length suggested by the AIC.[11]

To estimate the **v**ector **au**to**r**egression model with $p = 60$ lags of each endogenous variable, we type:

```
levant.AIC<-VAR(levant,type="const",p=60)
```

The `VAR` command requires the name of a dataset containing all of the endogenous variables. With the `type` option, we have chosen `"const"` in this case (the default). This means that each of our linear models includes a constant. (Other options include specifying a `"trend,"` `"both"` a constant and a trend, or `"none"` which includes neither.) With the option `p` we choose the lag length. An option we do not use here is the `exogen` option, which allows us to specify exogenous variables.

Once we have estimated our model using `VAR`, the next thing we should do is diagnose the model using a special call to the `plot` function:

```
plot(levant.AIC,lag.acf=104,lag.pacf=104)
```

By using the name of our VAR model, `levant.AIC`, as the only argument in `plot`, R will automatically provide a diagnostic plot for each of the endogenous

[11] You are encouraged to examine the models that would have been chosen by the Hannan–Quinn criterion (4 lags) or the Schwarz criterion (1 lag) on your own. How do these models perform in terms of diagnostics? How would inferences change?

variables. With the options of lag.acf and lag.pacf, we specify that the ACF and PACF plots that this command reports should show us 2 years' worth (104 weeks) of autocorrelation patterns. For our data, R produces six plots. R displays these plots one at a time, and between each, the console will pose the following prompt:

```
Hit <Return> to see next plot:
```

R will keep a plot on the viewer until you press the Return key, at which point it moves on to the next graph. This process repeats until the graph for each outcome has been shown.

Alternatively, if we wanted to see the diagnostic plot for one endogenous variable in particular, we could type:

```
plot(levant.AIC,lag.acf=104,lag.pacf=104,names="i2p")
```

Here, the names option has let us specify that we want to see the diagnostics for **i2p** (Israeli actions directed towards Palestine). The resulting diagnostic plot is shown in Fig. 9.5. The graph has four parts: At the top is a line graph that shows the true values in a solid black line and the fitted values in a blue dashed line. Directly beneath this is a line plot of the residuals against a line at zero. These first two graphs can illustrate whether the model consistently makes unbiased predictions of the outcome and whether the residuals are homoscedastic over time. In general, the forecasts consistently hover around zero, though for the last 200 observations the error variance does seem to increase slightly. The third graph, in the bottom left, is the ACF for the residuals on **i2p**.[12] Lastly, in the bottom right of the panel, we see the PACF for **i2p**'s residuals. No spikes are significant in the ACF and only one spike is significant in the PACF over 2 years, so we conclude that our lag structure has sufficiently filtered-out any serial correlation in this variable.

When interpreting a VAR model, we turn to two tools to draw inferences and interpretations from these models. First, we use Granger causality testing to determine if one endogenous variable causes the others. This test is simply a block F-test of whether all of the lags of a variable can be excluded from the model. For a joint test of whether one variable affects the other variables in the system, we can use the causality command. For example, if we wanted to test whether Israeli actions towards Palestine caused actions by the other five directed dyads, we would type:

```
causality(levant.AIC, cause="i2p")$Granger
```

The command here calls for the name of the model first (levant.AIC), and then with the cause option, we specify which of the endogenous variables we wish to test the effect of. Our output is as follows:

[12]Note that, by default, the graph R presents actually includes the zero-lag perfect correlation. If you would like to eliminate that, given our long lag length and the size of the panel, simply load the TSA package before drawing the graph to change the default.

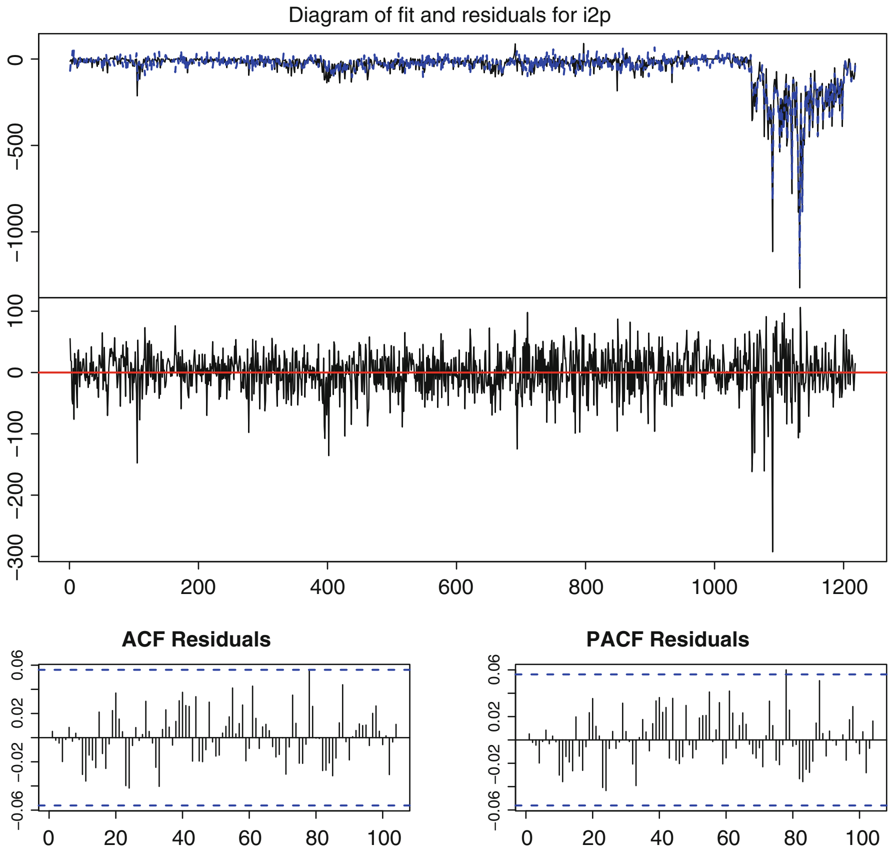

Fig. 9.5 Predicted values, residual autocorrelation function, and residual partial autocorrelation function for the Israel-to-Palestine series in a six-variable vector autoregression model

```
Granger causality H0: i2p do not Granger-cause
   a2i a2p i2a
p2i p2a

data:  VAR object levant.AIC
F-Test = 2.1669, df1 = 300, df2 = 5142, p-value
 < 2.2e-16
```

The null hypothesis is that the coefficients for all of the lags of **i2p** are zero when modeling each of the other five outcome variables. However, our F-test is significant here, as the minuscule p-value shows. Therefore, we would conclude that Israeli actions directed towards Palestine do have a causal effect on the other political action variables in the system.

We can proceed to test whether each of the other five variables Granger-cause the other predictors in the system by considering each one with the `causality` command:

```
causality(levant.AIC, cause="a2i")$Granger
causality(levant.AIC, cause="a2p")$Granger
causality(levant.AIC, cause="i2a")$Granger
causality(levant.AIC, cause="p2i")$Granger
causality(levant.AIC, cause="p2a")$Granger
```

The results are not reprinted here to preserve space. At the 95 % confidence level, though, you will see that each variable significantly causes the others, except for American actions directed towards Israel (**a2i**).

Finally to get a sense of the substantive impact of each predictor, we turn to impulse response analysis. The logic here is somewhat similar to the intervention analysis we graphed back in Fig. 9.3. An impulse response function considers a one-unit increase in one of the endogenous variables and computes how such an exogenous shock would dynamically influence all of the endogenous variables, given autoregression and dependency patterns.

When interpreting an impulse response function, a key consideration is the fact that shocks to one endogenous variable are nearly always correlated with shocks to other variables. We therefore need to consider that a one-unit shock to one variable's residual term is likely to create contemporaneous movement in the residuals of the other variables. The off-diagonal terms of the variance–covariance matrix of the endogenous variables' residuals, $\hat{\boldsymbol{\Sigma}}$, tells us how much the shocks covary.

There are several ways to deal with this issue. One is to use theory to determine the *ordering* of the endogenous variables. In this approach, the researcher assumes that a shock to one endogenous variable is not affected by shocks to any other variable. Then a second variable's shocks are affected only by the first variable's, and no others. The researcher recursively repeats this process to identify a system in which all variables can be ordered in a causal chain. With a theoretically designed system like this, the researcher can determine how contemporaneous shocks affect each other with a structured Cholesky decomposition of $\hat{\boldsymbol{\Sigma}}$ (Enders 2009, p. 309). A second option, which is the default option in R's `irf` command, is to assume that there is no theoretical knowledge of the causal ordering of the contemporaneous shocks and apply the method of *orthogonalization of the residuals*. This involves another Cholesky decomposition, in which we find $\mathbf{A}_0^{-1}$ by solving $\mathbf{A}_0^{-1\prime}\mathbf{A}_0 = \hat{\boldsymbol{\Sigma}}$. For more details about response ordering or orthogonalizing residuals, see Brandt and Williams (2007, pp. 36–41 & 66–70), Enders (2009, pp. 307–315), or Hamilton (1994, pp. 318–324).

As an example of an impulse response function, we will graph the effect of one extra political event from Israel directed towards Palestine. R will compute our <u>i</u>mpulse <u>r</u>esponse <u>f</u>unction with the `irf` command, using the default orthogonalization of residuals. One of the key options in this command is `boot`, which determines whether to construct a confidence interval with <u>**boot**</u>straps. Generally, it is advisable to report uncertainty in predictions, but the process can take several minutes.[13] So if the reader wants a quick result, set `boot=FALSE`. To get the result with confidence intervals type:

```
levant.irf<-irf(levant.AIC,impulse="i2p",n.ahead=12,boot=TRUE)
```

We name our impulse response function `levant.irf`. The command requires us to state the name of our model (`levant.AIC`). The option `impulse` asks us to name the variable we want the effect of, the `n.ahead` option sets how far ahead we wish to forecast (we say 12 weeks, or 3 months), and lastly `boot` determines whether to create confidence intervals based on a bootstrap sample.

Once we have computed this, we can graph the impulse response function with a special call to `plot`:

```
plot(levant.irf)
```

The resulting graph is shown in Fig. 9.6. There are six panels, one for each of the six endogenous variables. The vertical axis on each panel lists the name of the endogenous variable and represents the expected change in that variable. The horizontal axis on each panel represents the number of months that have elapsed since the shock. Each panel shows a solid red line at zero, representing where a non-effect falls on the graph. The solid black line represents the expected effect in each month, and the red dashed lines represent each confidence interval. As the figure shows, the real impact of a shock in Israel-to-Palestine actions is dramatic to the Israel-to-Palestine series itself (**i2p**) on account of autoregression and feedback from effects to other series. We also see a significant jump in Palestine-to-Israel actions (**p2i**) over the ensuing 3 months. With the other four series, the effects pale in comparison. We easily could produce plots similar to Fig. 9.6 by computing the impulse response function for a shock in each of the six inputs. This is generally a good idea, but is omitted here for space.

9.4 Further Reading About Time Series Analysis

With these tools in hand, the reader should have some sense of how to estimate and interpret time series models in R using three approaches—Box–Jenkins modeling, econometric modeling, and vector autoregression. Be aware that, while this chapter uses simple examples, time series analysis is generally challenging. It can be difficult to find a good model that properly accounts for all of the trend and error processes, but the analyst must carefully work through all of these issues or the

[13] Beware that bootstrap-based confidence intervals do not always give the correct coverages because they confound information about how well the model fits with uncertainty of parameters. For this reason, Bayesian approaches are often the best way to represent uncertainty (Brandt and Freeman 2006; Sims and Zha 1999).

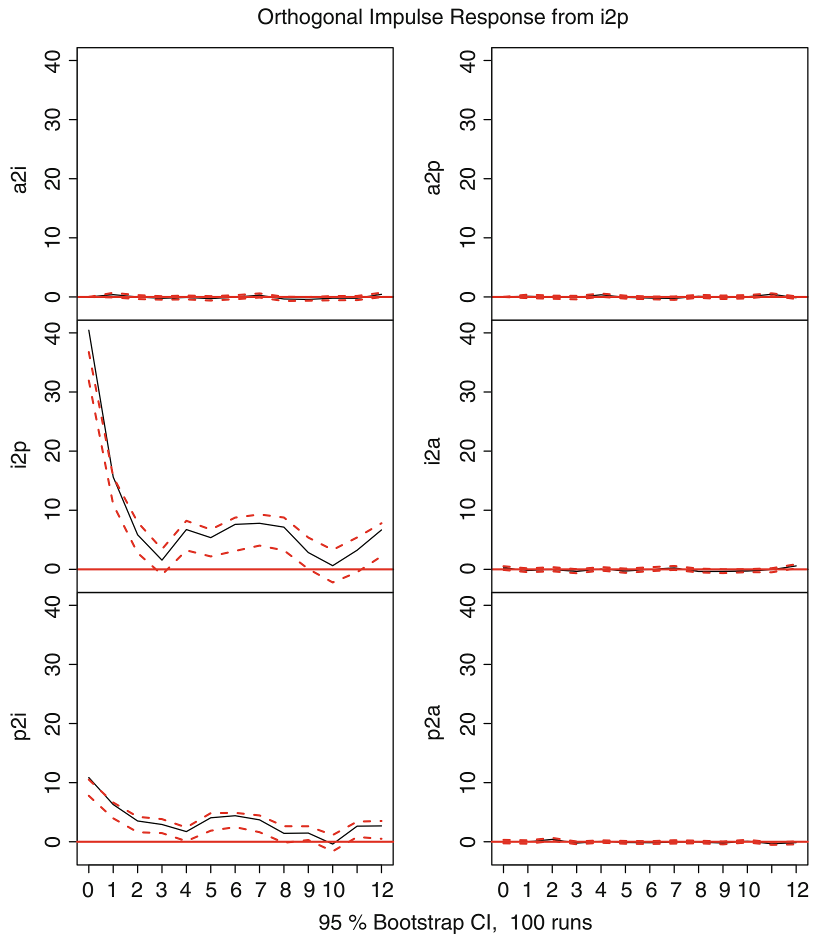

Fig. 9.6 Impulse response function for a one-unit shock in the Israel-to-Palestine series in a six-variable vector autoregression model

inferences will be biased. (See again, Granger and Newbold 1974.) So be sure to recognize that it often takes several tries to find a model that properly accounts for all issues.

It is also worth bearing in mind that this chapter cannot comprehensively address any of the three approaches we consider, much less touch on all types of time series analysis. (Spectral analysis, wavelet analysis, state space models, and error-correction models are just a handful of topics not addressed here.) Therefore, the interested reader is encouraged to consult other resources for further important details on various topics in time series modeling. Good books that cover a range of time series topics and include R code are: Cowpertwait and Metcalfe (2009), Cryer and Chan (2008), and Shumway and Stoffer (2006). For more depth on the theory behind time series, good volumes by Political Scientists include Box-Steffensmeier et al. (2014) and Brandt and Williams (2007). Other good books that cover time series theory include: Box et al. 2008, Enders (2009), Wei (2006), Lütkepohl (2005), and for an advanced take see Hamilton (1994). Several books focus on specific topics and applying the methods in R: For instance, Petris et al. (2009) covers dynamic linear models, also called state space models, and explains how to use the corresponding `dlm` package in R to apply these methods. Pfaff (2008) discusses cointegrated data models such as error-correction models and vector error-correction models, using the R package `vars`. Additionally, readers who are interested in time series cross section models, or panel data, should consult the `plm` package, which facilitates the estimation of models appropriate for those methods. Meanwhile, Mátyás and Sevestre (2008) offers a theoretical background on panel data methods.

9.5 Alternative Time Series Code

As mentioned in Sect. 9.1, we will now show alternative syntax for producing Figs. 9.3 and 9.4. This code is a little longer, but is more generalizable.[14] First off, if you do not have all of the packages, data objects, and models loaded from before, be sure to reload a few of them so that we can draw the figure:

```
library(TSA)
pres.energy<-read.csv("PESenergy.csv")
predictors<-as.matrix(subset(pres.energy,select=c(rmn1173,
     grf0175,grf575,jec477,jec1177,jec479,embargo,hostages,
     oilc,Approval,Unemploy)))
```

All three of the previous lines of code were run earlier in the chapter. By way of reminder, the first line loads the `TSA` package, the second loads our energy policy coverage data, and the third creates a matrix of predictors.

Now, to redraw either figure, we need to engage in a bit of data management:

```
months <- 1:180
static.predictors <- predictors[,-1]
dynamic.predictors <- predictors[,1, drop=FALSE]
y <- ts(pres.energy$Energy, frequency=12, start=c(1972, 1))
```

First, we define a month index, as before. Second, we subset our matrix `predictors` to those that have a static effect. Third, we isolate our dynamic predictor of the Nixon's speech. Fourth, we use the `ts` command to declare energy policy coverage to be a time series object. On this last line, we use the `frequency` option to specify that these are monthly data (hence the value 12) and the `start` option to note that these data begin in the first month of 1972.

[14]My thanks to Dave Armstrong for writing and suggesting this alternative code.

Next, we need to actually estimate our transfer function. Once we have done this, we can save several outputs from the model:

```
dynamic.mod<-arimax(y,order=c(1,0,0),xreg=static.predictors,
     xtransf=dynamic.predictors,transfer=list(c(1,0)))
b <- coef(dynamic.mod)
static.coefs <- b[match(colnames(static.predictors), names(b))]
ma.coefs <- b[grep("MA0", names(b))]
ar.coefs <- b[grep("AR1", names(b))]
```

The first line refits our transfer function. The second uses the `coef` command to extract the coefficients from the model and save them in a vector named `b`. The last three lines separate our coefficients into static effects (`static.coefs`), initial dynamic effects (`ma.coefs`), and decay terms (`ar.coefs`). In each line, we carefully reference the names of our coefficient vector, using the `match` command to find coefficients for the static predictors, and then the `grep` command to search for terms that contain `MA0` and `AR1`, respectively, just as output terms of a transfer function do.

With all of these elements extracted, we now turn specifically to redrawing Fig. 9.3, which shows the effect of the Nixon's speech intervention against the real data. Our intervention effect consists of two parts, the expected value from holding all of the static predictors at their values for the 58th month, and the dynamic effect of the transfer function. We create this as follows:

```
xreg.pred<-b["intercept"]+static.coefs%*%static.predictors[58,]
transf.pred <- as.numeric(dynamic.predictors%*%ma.coefs+
     ma.coefs*(ar.coefs^(months-59))*(months>59))
y.pred<-ts(xreg.pred+transf.pred,frequency=12,start=c(1972,1))
```

The first line simply makes the static prediction from a linear equation. The second uses our initial effects and decay terms to predict the dynamic effect of the intervention. Third, we add the two pieces together and save them as a time series with the same frequency and start date as the original series. With both `y` and `y.pred` now coded as time series of the same frequency over the same time span, it is now easy to recreate Fig. 9.3:

```
plot(y,xlab="Month", ylab="Energy Policy Stories",type="l")
lines(y.pred, lty=2,col='blue',lwd=2)
```

The first line simply plots the original time series, and the second line adds the intervention effect itself.

With all of the setup work we have done, reproducing Fig. 9.4 now only requires three lines of code:

```
full.pred<-fitted(dynamic.mod)
plot(full.pred,ylab="Energy Policy Stories",type="l",
     ylim=c(0,225))
points(y, pch=20)
```

The first line simply uses the `fitted` command to extract fitted values from the transfer function model. The second line plots these fitted values, and the third adds the points that represent the original series.

9.6 Practice Problems

This set of practice problems reviews each of the three approaches to time series modeling introduced in the chapter, and then poses a bonus question about the Peake and Eshbaugh-Soha energy data that asks you to learn about a new method. Questions #1–3 relate to single-equation models, so all of these questions use a dataset about electricity consumption in Japan. Meanwhile, question #4 uses US economic data for a multiple equation model.

1. Time series visualization: Wakiyama et al. (2014) study electricity consumption in Japan, assessing whether the March 11, 2011, Fukushima nuclear accident affected electricity consumption in various sectors. They do this by conducting an intervention analysis on monthly measures of electricity consumption in megawatts (MW), from January 2008 to December 2012. Load the `foreign` package and open these data in Stata format from the file `comprehensiveJapanEnergy.dta`. This data file is available from the Dataverse (see page vii) or this chapter's online content (see page 155). We will focus on household electricity consumption (variable name: **house**). Take the logarithm of this variable and draw a line plot of logged household electricity consumption from month-to-month. What patterns are apparent in these data?
2. Box–Jenkins modeling:
   a. Plot the autocorrelation function and partial autocorrelation function for logged household electricity consumption in Japan. What are the most apparent features from these figures?
   b. Wakiyama et al. (2014) argue that an ARIMA(1,0,1), with a Seasonal ARIMA(1,0,0) component fit this series. Estimate this model and report your results. (*Hint:* For this model, you will want to include the option `seasonal=list(order=c(1,0,0), period=12)` in the `arima` command.)
   c. How well does this ARIMA model fit? What do the ACF and PACF look like for the residuals from this model? What is the result of a Ljung–Box Q-test?
   d. Use the `arimax` command from the `TSA` package. Estimate a model that uses the ARIMA error process from before, the static predictors of temperature (**temp**) and squared temperature (**temp2**), and a transfer function for the Fukushima intervention (**dummy**).
   e. Bonus: The Fukushima indicator is actually a *step* intervention, rather than a *pulse*. This means that the effect *cumulates* rather than *decays*. Footnote 8 describes how these effects cumulate. Draw a picture of the cumulating effect of the Fukushima intervention on logged household electricity consumption.
3. Econometric modeling:
   a. Fit a static linear model using OLS for logged household electricity consumption in Japan. Use temperature (**temp**), squared temperature (**temp2**), and the Fukushima indicator (**dummy**) as predictors. Load the `lmtest` package,

and compute both a Durbin–Watson and Breusch–Godfrey test for this linear model. What conclusions would you draw from each? Why do you think you get this result?

b. Reestimate the static linear model of logged household electricity consumption using FGLS with the Cochrane–Orcutt algorithm. How similar or different are your results from the OLS results? Why do you think this is?

c. Load the `dyn` package, and add a lagged dependent variable to this model. Which of the three econometric models do you think is the most appropriate and why? Do you think your preferred econometric model or the Box–Jenkins intervention analysis is more appropriate? Why?

4. Vector autoregression:

   a. Enders (2009, p. 315) presents quarterly data on the US economy, which runs from the second quarter of 1959 to the first quarter of 2001. Load the `vars` and `foreign` packages, and then open the data in Stata format from the file `moneyDem.dta`. This file is available from the Dataverse (see page vii) or this chapter's online content (see page 155). Subset the data to only include three variables: change in logged real GDP (**dlrgdp**), change in the real M2 money supply (**dlrm2**), and change in the 3-month interest rate on US Treasury bills (**drs**). Using the `VARselect` command, determine the best-fitting lag length for a VAR model of these three variables, according to the AIC.
   b. Estimate the model you determined to be the best fit according to the AIC. Examine the diagnostic plots. Do you believe these series are clear of serial correlation and that the functional form is correct?
   c. For each of the three variables, test whether the variable Granger-causes the other two.
   d. In monetary policy, the interest rate is an important policy tool for the Federal Reserve Bank. Compute an impulse response function for a percentage point increase in the interest rate (**drs**). Draw a plot of the expected changes in logged money supply (**dlrm2**) and logged real GDP (**dlrgdp**). (*Hint:* Include the option `response=c("dlrgdp","dlrm2")` in the `irf` function.) Be clear about whether you are orthogonalizing the residuals or making a theoretical assumption about response ordering.

5. Bonus: You may have noticed that Peake and Eshbaugh-Soha's (2008) data on monthly television coverage of the energy issue was used both as an example for count regression in Chap. 7 and as an example time series in this chapter. Brandt and Williams (2001) develop a Poisson autoregressive (PAR) model for time series count data, and Fogarty and Monogan (2014) apply this model to these energy policy data. Replicate this PAR model on these data. For replication information see: http://hdl.handle.net/1902.1/16677.