Introduction to Time Series Modeling and Forecasting with Applications in R

David Ubilava

April 2022

Contents

Foreword

'Prediction is very difficult, especially about the future.' – a perfect quote about our inability to make a perfect forecast, often attributed to Niels Bohr, is very much on-brand for the study of forecasting. And so it goes.

I came across this quote as I was wrapping up my doctoral dissertation at Purdue University. My dissertation topic was 'Nonlinear Multivariate Modeling and Forecasting of Commodity Prices.' When I started my doctoral degree, I had not taken a designated time series econometrics or forecasting course. By the time I was completing my degree, I had learned enough to know how difficult it was to achieve accurate forecasts. A chapter of my dissertation, which pretty much was a forecasting exercise, yielded the so-called 'null results.' I felt uneasy about it. I had an urge to justify, somehow, these null results. And then I stumbled across this quote, which 'saved my day.' I stuck it as an epigraph of my dissertation. It captured my struggles as a forecaster, and provided the context to the results of my work.

Some years later, as I started teaching a course on forecasting, I spend a great deal of time explaining—indeed preparing students for the inevitable—that forecasting is difficult, and that more often than not we will not be able to give accurate forecasts. But I also stress that this should not discourage us from trying because most great achievements in this world have myriads of failed attempts as a foundation.

What is forecast if not a guess? An educated guess, nonetheless. A good guess doesn't have to be spot-on. It almost never is, only if by fluke. In fact, an inaccurate guess can be helpful too. That we were unable to exactly predict an event, tells us something about the underlying processes that result in an outcome different from what we predicted. Such conjecture can be useful. Forecasting, even if inaccurate, can be useful. George Box's 'all models are

wrong but some are useful' is certainly suitable to the study of forecasting. This book introduces time series models and presents a case of their usefulness in predicting events.

Preliminaries

Chapter 1

Introduction to Forecasting

Roots of forecasting extend very much to the beginning of human history. In their desire to predict the future, people have attempted to make forecasts of their own, or have used services of others. Fortunetellers, for example, have been forecast experts of some sort, basing their predictions on magic. They are less common in the current age. Astrologers, who rely on astronomical phenomena to foresee the future, maintain their relevance to this date. Over time, and particularly with the development of the study of econometrics, more rigorous forecasting methods have been introduced and developed. All methods – primitive or complex, spurious or scientifically substantiated – have one thing in common: they all rely (or, at least, pretend to rely) on *information*.

Information is key in forecasting. It comes in many forms, but once it is organized and stored, we end up with *data*. Data that is organized and stored a certain way – chronologically and at regular intervals – are referred to as the *time series*. A diverse set of forecasting methods typically rely on insights from econometric analysis of time series. In time series analysis, the implied assumption is that the past tends to repeat itself, at least to some extent. So, if we well study the past, we *may* be able to forecast an event with some degree of accuracy.

Accurate forecasting is difficult. Indeed, forecasting is difficult. So, as we forecast, we are bound to make mistakes, which manifest into the *forecast errors*. We have these errors because we don't know the true model, so we

make a guess about the most suitable model for time series analysis. We also have the errors because we don't know the true parameters of the model. Instead, using a subset of history we estimate the parameters, which are prone to a sampling error. Finally, even if our guess about the model is exactly correct, and even if our parameter estimates happen to be exactly equal to the true parameters of the 'right' model, our forecasts may still be off, because of all those factors that have never happened in past, and only characterize the future. And because of this, there is no such thing as precise forecast, even if by fluke we were to exactly predict an outcome of an event. But some forecasts are better than others. And in search of such forecasts the study of time series econometrics has evolved.

Chapter 2

Basics of Time Series Econometics

A time series is a realization of a sequence of random variables that are stored chronologically. This sequence is referred to as the *stochastic process*. Thus, a time series is a realization of the stochastic process. We index time periods as $1, 2, \ldots, T$, so that the stochastic process is given by $\{Y_t : t = 1, \ldots, T\}$. A time series is represented by a set of observations: $\{y_1, \ldots, y_T\}$. One can view a time series as a finite sample from an underlying doubly–infinite sequence: $\{\ldots, y_{-1}, y_0, y_1, y_2, \ldots, y_T, y_{T+1}, y_{T+2}, \ldots\}$. This is to say that the history extends beyond the starting and ending time periods of the sample at hand.

2.1 Stationarity

If all random variables, from where the time series are drawn, have the same distribution, then we refer to such data as *stationary* time series. Stationarity is an important feature, and the assumption, on which the time series analysis heavily relies.

Before diving any further into the concepts and methods of time series econometrics, consider the simplest kind of time series comprised of realizations from independent and identically distributed normal random variable with zero mean and constant variance: $\varepsilon_t \sim iid\,(0, \sigma^2)$. The following graph plots this time series against time.

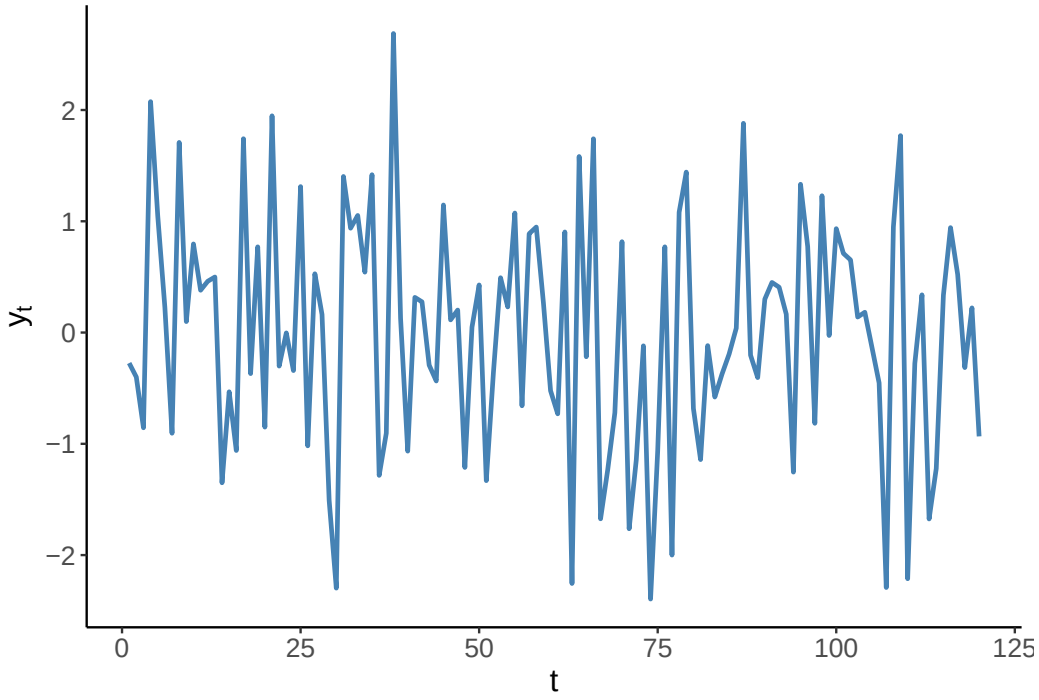


Figure 2.1: White noise: an illustration

Such time series are referred to as *white noise*. That is, $\{y_t : t = 1, \ldots, T\}$, is a white noise process if:

$$\begin{aligned} &E(y_t) = 0, \ \ \forall\ t \\ &Var(y_t) = \sigma^2, \ \ \forall\ t \\ &Cov(y_t, y_{t-k}) = 0, \ \ \forall\ k \neq 0 \end{aligned}$$

Because each observation is drawn from the same distribution, white noise is a stationary time series. Indeed, it is a special type of stationary time series insofar as its mean, variance, and covariance are time-invariant. Note, for a time series to be stationary, neither the mean nor the covariances need to be equal to zero. Thus, $\{y_t\}$ is stationary if the mean and variance are independent of t, and the autocovariances are independent of t for all k.

2.2 Serial Dependence

It is more of the norm rather than the exception for a time series to be correlated over time. Indeed, because of the sequential nature of time series, we commonly observe dependence among the temporally adjacent time series. That is, for most economic time series, we would expect y_t and y_{t-1} to be correlated. Such correlation, referred to as the first order *autocorrelations*, is given by: $\rho_1 = Cor(y_t, y_{t-1}) = \frac{Cov(y_t, y_{t-k})}{Var(y_t)}$. In general, the k^{th} order autocorrelation is given by:

$$\rho_k = Cor(y_t, y_{t-k}) = \frac{Cov(y_t, y_{t-k})}{Var(y_t)}, \quad k = 1, 2, \ldots$$

Autocorrelations are commonly illustrated via the so-called *autocorrelogram*, which plots the sequence of autocorrelation coefficients against the lags at which these coefficients are obtained. For example, an autocorrelogram of the previously illustrated white noise process is as follows:

For each k, the vertical line extending from zero represents the autocorrelation coefficient at that lag. The red dashed lines denote the 90% confidence interval, given by $\pm 1.96/\sqrt{T}$, where T is the length of the time series.

Another relevant measure of the time series dependence is partial autocorrelation, which is correlation between y_t and y_{t-k} net of any correlations between

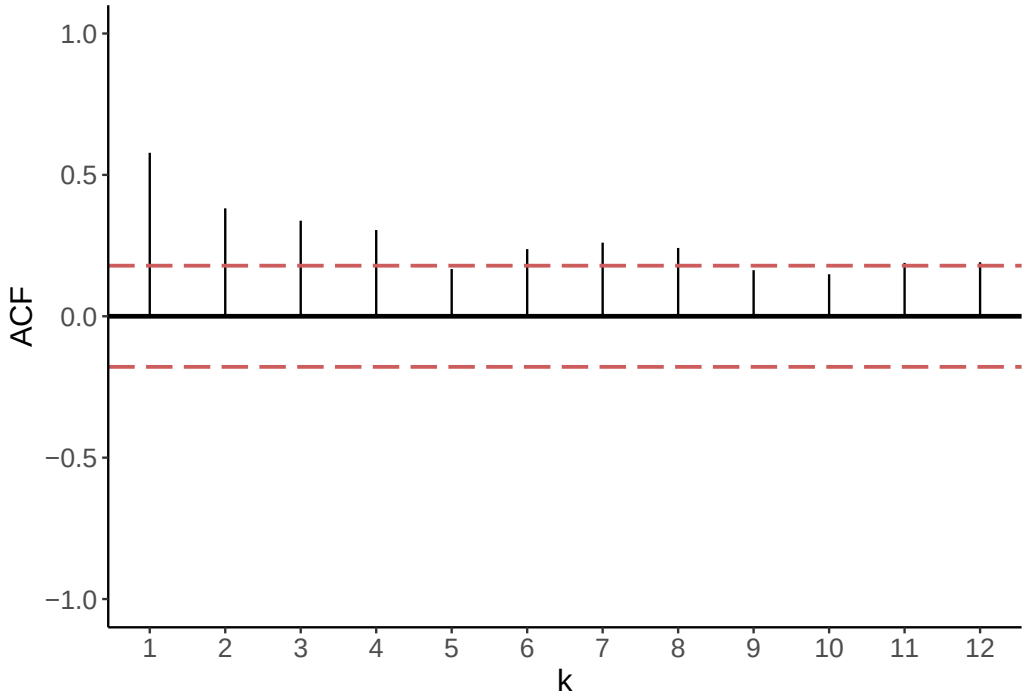


Figure 2.2: Autocorrelation

y_t and y_{t-k+j}, for all $j = 1, \ldots, k-1$. Similar to autocorrelations, partial autocorrelations can also be illustrated using autocorrelograms:

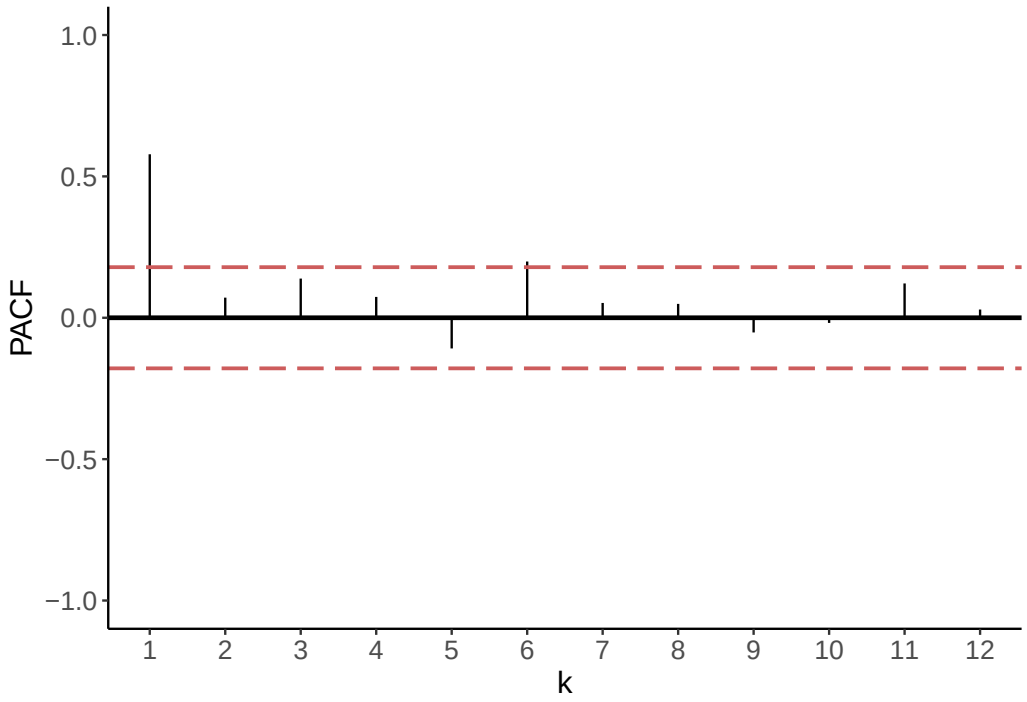


Figure 2.3: Partial Autocorrelation

2.3 Transformations

It is common to transform time series by taking logarithms, differences, or differences of logarithms (growth rates). Such transformations usually are done to work with the suitable variable for the desired econometric analysis. For example, if an economic time series is characterized by an apparent exponential growth (e.g., real GDP), by taking natural logarithms the time series "flatten'' and the fluctuations become proportionate. The difference operator is denoted by Δ, so that $\Delta y_t = y_t - y_{t-1}$. The following three graphs illustrate (i) a time series with an apparent exponential growth, (ii) the natural logarithm of this time series, and (iii) their differences (i.e., the log-differences of the original series).

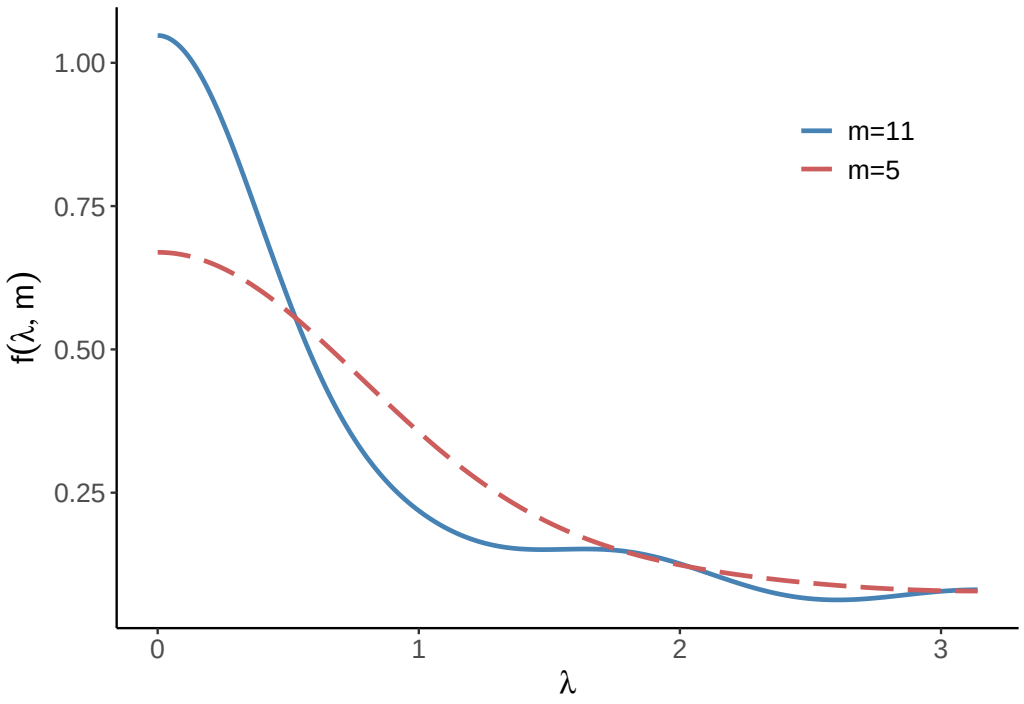


Figure 2.4: Spectral Density

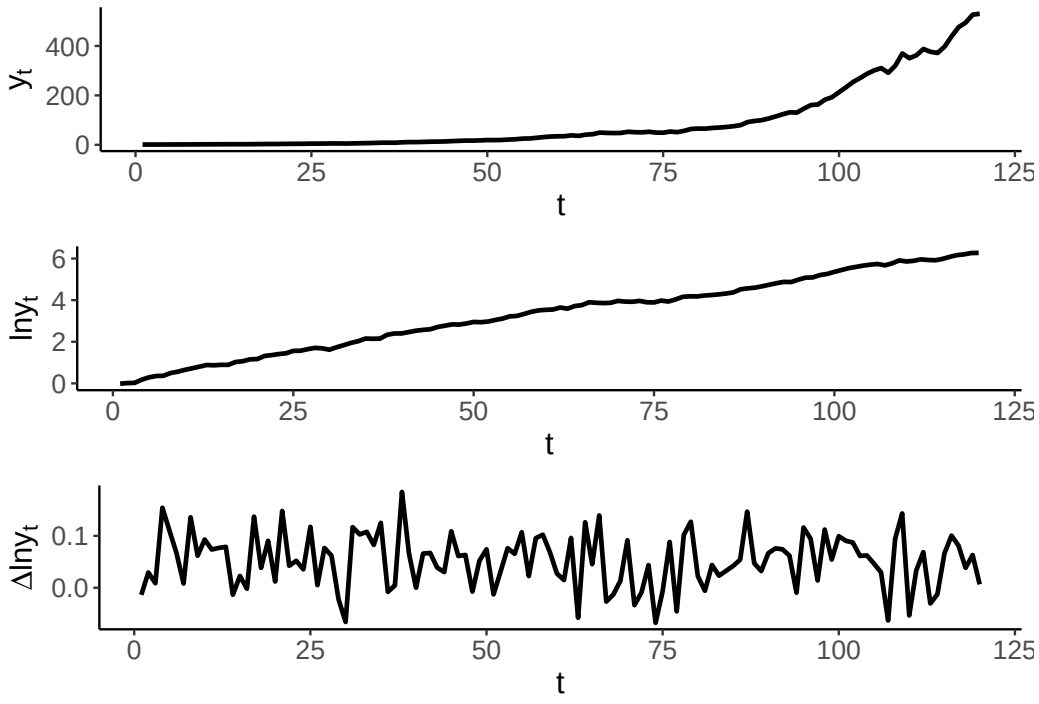


Figure 2.5: A time series and its transformations

Chapter 3

Forecasting Methods and Routines

3.1 Optimal Forecast

A forecast is a random variable which has some distribution and, thus, moments. The simplest form of a forecast is a point forecast (usually a mean of the distribution, but can be a median or, indeed, any quantile).

A point forecast made in period t for horizon h can be denoted as $y_{t+h|t}$; this is our 'best guess,' that is made in period t, about the actual realization of the random variable in period $t+h$, denoted by y_{t+h}. The difference between the two is the forecast error. That is,

$$e_{t+h|t} = y_{t+h} - y_{t+h|t}$$

The more accurate is the forecast the smaller is the forecast error.

Three types of uncertainty contribute to the forecast error:

$$\begin{aligned} e_{t+h|t} = &\Big[y_{t+h} - E(y_{t+h}|\Omega_t)\Big] \quad \text{(forecast uncertainty)} \\ &+ \Big[E(y_{t+h}|\Omega_t) - g(\Omega_t;\theta)\Big] \quad \text{(model uncertainty)} \\ &+ \Big[g(\Omega_t;\theta) - g(\Omega_t;\hat{\theta})\Big] \quad \text{(parameter uncertainty)} \end{aligned}$$

where Ω_t denotes the information set available at the time when the forecast is made; $g(\cdot)$ is a functional form of a model applied to the data; θ is a set of parameters of the model, and $\hat{\theta}$ are their estimates

Because uncertainty cannot be avoided, a forecaster is bound to commit forecast errors. The goal of the forecaster is to minimize the 'cost' associated with the forecast errors. This is achieved by minimizing the expected loss function.

A loss function, $L(e_{t+h|t})$, can take many different forms, but is should satisfy the following properties:

$$\begin{aligned} &L(e_{t+h|t}) = 0, \quad \forall\ e_{t+h|t} = 0 \\ &L(e_{t+h|t}) \geq 0, \quad \forall\ e_{t+h|t} \neq 0 \\ &L(e^{(i)}_{t+h|t}) > L(e^{(j)}_{t+h|t}), \quad \forall\ |e^{(i)}_{t+h|t}| > |e^{(j)}_{t+h|t}| \end{aligned}$$

Two commonly used symmetric loss functions are *absolute* and *quadratic* loss functions:

$$\begin{aligned} &L(e_{t+h|t}) = |e_{t+h|t}| \quad \text{(absolute loss function)} \\ &L(e_{t+h|t}) = (e_{t+h|t})^2 \quad \text{(quadratic loss function)} \end{aligned}$$

The quadratic loss function is popular, partly because we typically select models based on 'in-sample' quadratic loss (i.e. by minimizing the sum of squared residuals).

Optimal forecast is the forecast that minimizes the expected loss:

$$\min_{y_{t+h|t}} E\left[L\left(e_{t+h|t}\right)\right] = \min_{y_{t+h|t}} E\left[L\left(y_{t+h} - y_{t+h|t}\right)\right]$$

where the expected loss is given by:

$$E\left[L\left(y_{t+h} - y_{t+h|t}\right)\right] = \int L\left(y_{t+h} - y_{t+h|t}\right) f(y_{t+h}|\Omega_t)dy$$

We can assume that the conditional density is a normal density with mean $\mu_{t+h} \equiv E(y_{t+h})$, and variance $\sigma^2_{t+h} \equiv Var(y_{t+h})$.

Under the assumption of the quadratic loss function:

$$\begin{aligned} E\left[L(e_{t+h|t})\right] &= E(e^2_{t+h|t}) = E(y_{t+h} - \hat{y}_{t+h|t})^2 \\ &= E(y^2_{t+h}) - 2E(y_{t+h})\hat{y}_{t+h|t} + \hat{y}^2_{t+h|t} \end{aligned}$$

By solving the optimization problem it follows that:

$$\hat{y}_{t+h|t} = E(y_{t+h}) \equiv \mu_{t+h}$$

Thus, the optimal point forecast under the quadratic loss is the *mean* (for reference, the optimal point forecast under absolute loss is the *median*).

3.2 Measuring Forecast Accuracy

Forecast accuracy should only be determined by considering how well a model performs on data not used in estimation. But to assess forecast accuracy we need access to the data, typically from future time periods, that was not used in estimation. This leads to the so-called 'pseudo forecasting' routine. This routine involves splitting the available data into two segments referred to as 'in-sample' and 'out-of-sample.' The in-sample segment of a series is also known as the 'estimation set' or the 'training set.' The out-of-sample segment of a series is also known as the 'hold-out set' or the 'test set.'

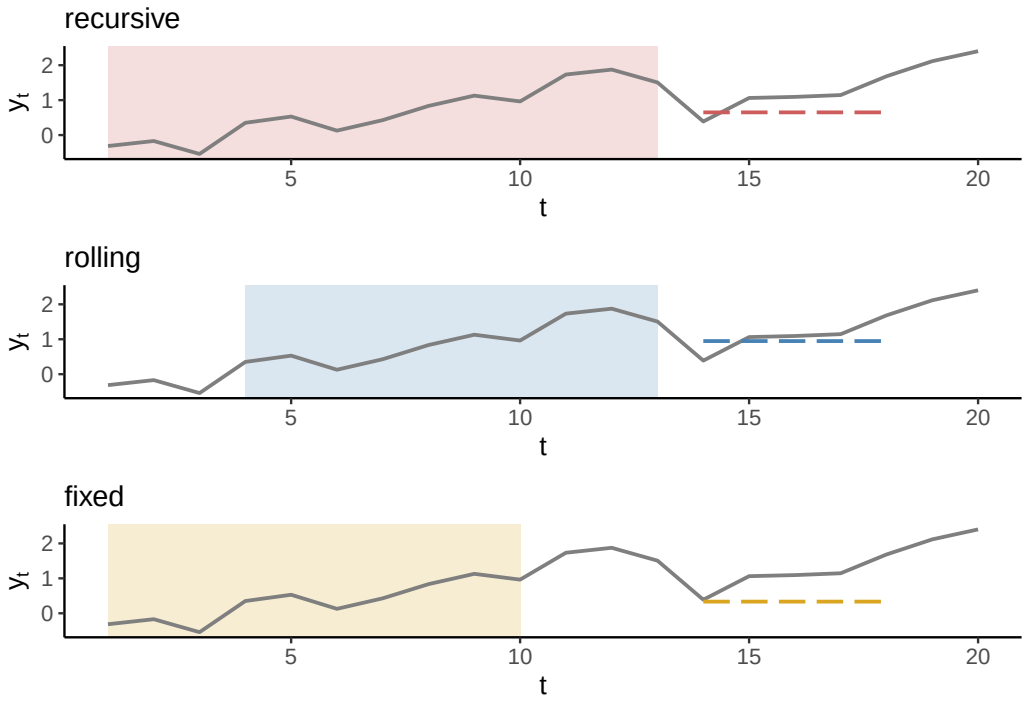
Thus, we make the so-called 'genuine' forecasts using only the information from the estimation set, and assess the accuracy of these forecasts in an out-of-sample setting.

Because forecasting is often performed in a time series context, the estimation set typically predates the hold-out set. In non-dynamic settings such chronological ordering may not be necessary, however.

There are different forecasting schemes for updating the information set in the pseudo-forecasting routine. These are: *recursive*, *rolling*, and *fixed*.

- The recursive forecasting environment uses a sequence of expanding windows to update model estimates and the information set.
- The rolling forecasting environment uses a sequence of rolling windows of the same size to update model estimates and the information set.
- The fixed forecasting environment uses one fixed window for model estimates, and only updates the information set.

The following animation illustrates the distinctive features of these three routines:



3.3 Evaluating Time Series Forecasts

To evaluate forecasts of a time series, $\{y_t\}$, with a total of T observations, we divide the sample into two parts, the in-sample set with a total of R observations, such that $R < T$ (typically, $R \approx 0.75T$), and the out-of-sample set.

For example, if we are interested in one-step-ahead forecast assessment, this way we will produce a sequence of forecasts: $\{y_{R+1|R}, y_{R+2|R+1}, \ldots, y_{T|T-1}\}$ for $\{Y_{R+1}, Y_{R+2}, \ldots, Y_T\}$.

Forecast errors, $e_{R+j} = y_{R+j} - y_{R+j|R+j-1}$, then can be computed for $j = 1, \ldots, T - R$.

The most commonly applied accuracy measures are the mean absolute forecast

error (MAFE) and the root mean squared forecast error (RMSFE):

$$\text{MAFE} = \frac{1}{P}\sum_{i=1}^{P}|e_i|$$

$$\text{RMSFE} = \sqrt{\frac{1}{P}\sum_{i=1}^{P}e_i^2}$$

where P is the total number of out-of-sample forecasts. The lower is the accuracy measure (of choice), the better a given model performs in generating accurate forecasts. As noted earlier, 'better' does not mean 'without errors.'

Forecast errors of a 'good' forecasting method will have the following properties:

- zero mean; otherwise, the forecasts are biased.
- no correlation with the forecasts; otherwise, there is information left that should be used in computing forecasts.
- no serial correlation among one-step-ahead forecast errors. Note that k-step-ahead forecasts, for $k > 1$, can be, and usually are, serially correlated.

Any forecasting method that does not satisfy these properties has a potential to be improved.

3.3.1 Unbiasedness

Testing $E(e_{t+h|t}) = 0$. Set up a regression:

$$e_{t+h|t} = \alpha + \upsilon_{t+h} \qquad t = R, \ldots, T-h,$$

where R is the estimation window size, T is the sample size, and h is the forecast horizon length. The null of zero-mean forecast error is equivalent of testing $H_0 : \alpha = 0$ in the OLS setting. For h-step-ahead forecast errors, when $h > 1$, autocorrelation consistent standard errors should be used.

3.3.2 Efficiency

Testing $Cov(e_{t+h|t}, y_{t+h|t}) = 0$. Set up a regression:

$$e_{t+h|t} = \alpha + \beta y_{t+h|t} + \upsilon_{t+h} \qquad t = R, \ldots, T-h.$$

The null of forecast error independence of the information set is equivalent of testing $H_0 : \beta = 0$ in the OLS setting. For h-step-ahead forecast errors, when $h > 1$, autocorrelation consistent standard errors should be used.

3.3.3 No Autocorrelation

Testing $Cov(e_{t+1|t}, e_{t|t-1}) = 0$. Set up a regression:

$$e_{t+1|t} = \alpha + \gamma e_{t|t-1} + \upsilon_{t+1} \qquad t = R+1, \ldots, T-1.$$

The null of no forecast error autocorrelation is equivalent of testing $H_0 : \gamma = 0$ in the OLS setting.

Deterministic Time Series Models

Chapter 4

Trends

Economic time series usually are characterized by trending behavior, and often present a seasonal pattern as well. Trend is a unidirectional change of time series over an extended period of time that arises from the accumulation of information over time. Seasonality is a repeating pattern *within a calendar year* that arises from the links of technologies, preferences, and institutions to the calendar. Modeling and forecasting these time series features is a fairly straightforward task. But before we get to it, let's discuss what may happen if we were to ignore the presence of trends and/or seasonality when analyzing the time series data.

4.1 Spurious Relationship

Nothing about trending time series necessarily violates the classical linear regression model assumptions. The issue may arise, however, if an unobserved trending variable is simultaneously correlated with the dependent variable as well as one of the independent variables in a time series regression. In such case, we may find a (statistically significant) relationship between two or more unrelated economic variables simply because they are all trending. Such relationship is referred to a *spurious relationship*.

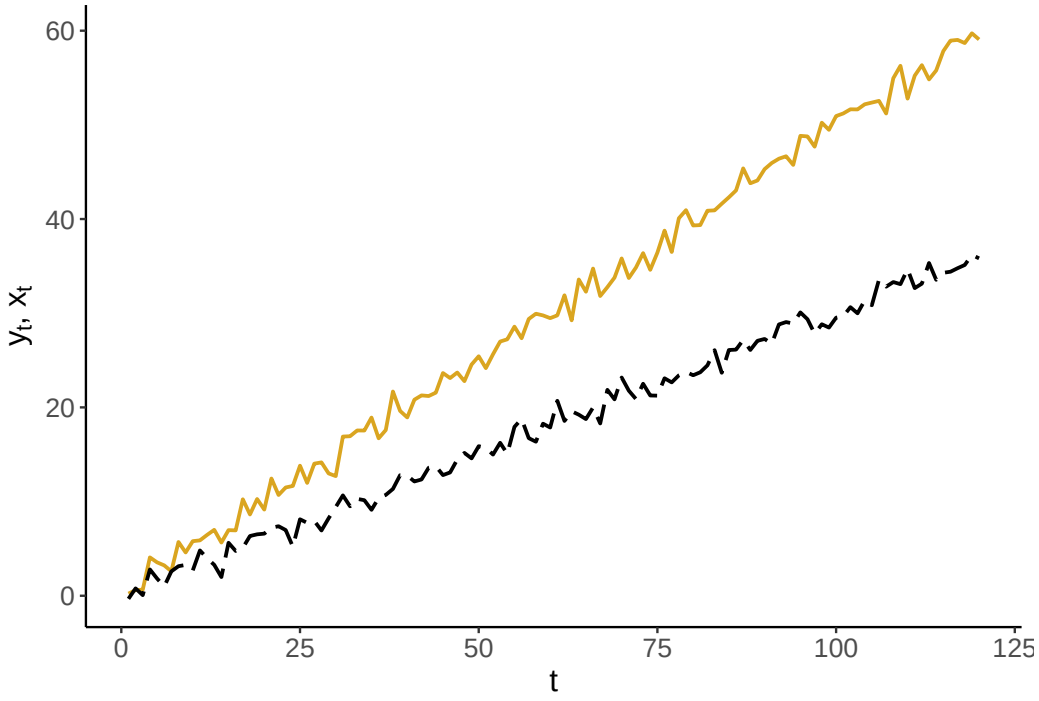
To illustrate, consider two trending variables:

$$y_t = \gamma t + \nu_t, \quad \nu \sim N(0, \sigma_\nu^2),$$

and

$$x_t = \delta t + \upsilon_t, \quad \upsilon \sim N(0, \sigma_\upsilon^2),$$

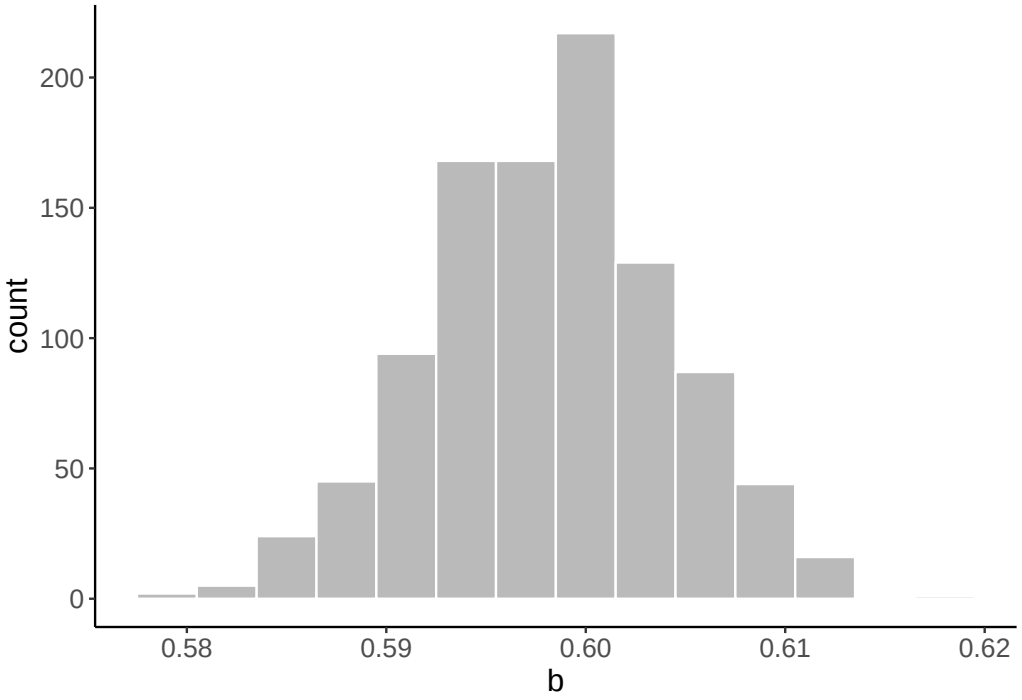
where $Cov(\nu_t, \upsilon_t) = 0$. For simplicity, we can assume $\sigma_\nu^2 = \sigma_\upsilon^2 = 1$. Suppose, γ and δ are some positive scalars, say, 0.3 and 0.5, respectively. That is, y and x are trending in the same direction. Below is an example of such time series:



If we were to estimate

$$y_t = \alpha + \beta x_t + \varepsilon_t,$$

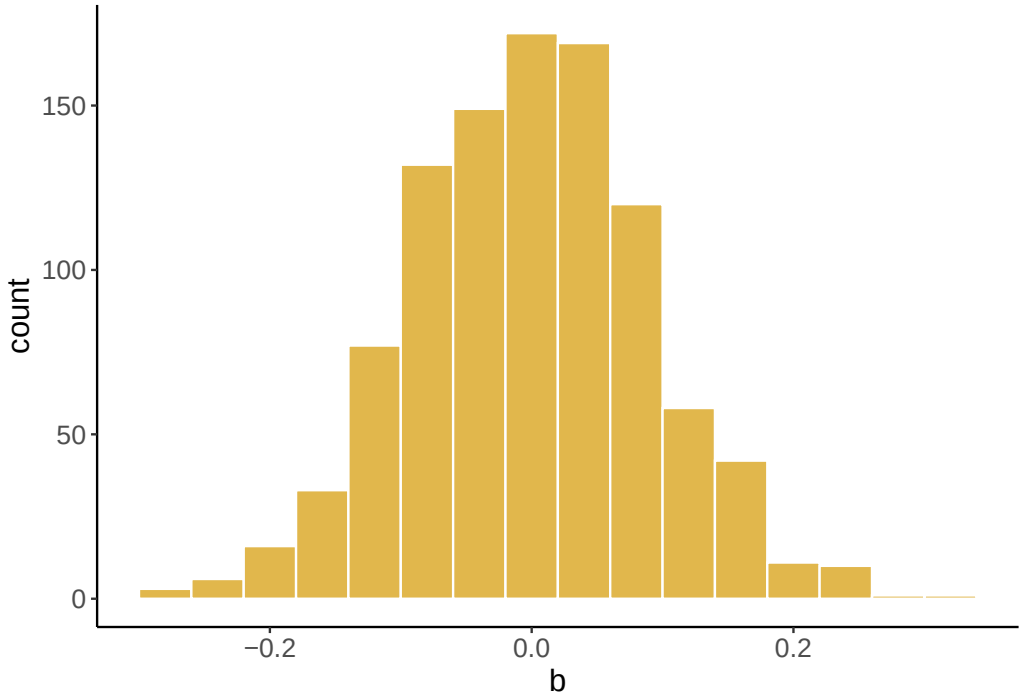
we are likely to find the relationship between the two – in this case $\beta > 0$ – even though, we know, the two are not related. To illustrate this, we will generate 1000 samples of size 120 for y and x, and in each case we will estimate the parameter β. The following graph illustrates the empirical distribution of these parameter estimates:



Luckily, we can easily "fix" the issue, by incorporating a trend in the regression:

$$y_t = \alpha + \beta x_t + \eta t + \varepsilon_t.$$

Once the trend is accounted for, the previously illustrated "bias" disappears. Using a similar simulation exercise as before, the following graph illustrates the empirical distribution of these parameter estimates:



In fact, this "fix" is equivalent to regressing a de-trended y on a de-trended x. To de-trend a variable, we first run a regression: $y_t = \gamma_0 + \gamma_1 t + \nu_t$, and then obtain the fitted values for some fixed trend (typically zero), that is: $\tilde{y}_t = \hat{\gamma}_0 + \hat{\nu}_t$, where $\hat{\gamma}_0$ and $\hat{\nu}_t$ are the parameter estimate and the residuals from the foregoing regression.

4.2 Modeling

As seen, accounting for trends in a time series can help us resolve some regression issues. But a trend in and of itself can be an inherent feature of a times series. To that end, we can apply deterministic trends to forecast time series.

The simplest (and perhaps most frequently applied) model to account for the trending time series is a *linear* trend model:

$$y_t = \alpha + \beta t$$

Other likely candidate trend specifications are *polynomial* (e.g. quadratic,

cubic, etc.), *exponential*, and *shifting* (or *switching*) trend models, respectively given by:

$$y_t = \alpha + \beta_1 t + \beta_2 t^2 + \ldots + \beta_p t^p$$
$$y_t = e^{\alpha+\beta t} \quad \text{or} \quad \ln y_t = \alpha + \beta t$$
$$y_t = \alpha + \beta_1 t + \beta_2 (t-\tau) I(t > \tau), \quad \tau \in \mathsf{T}$$

Of these, here we will primarily consider linear and quadratic trends. An exponential trend, from the standpoint of modeling and forecasting, is equivalent to a linear trend fitted to natural logarithm of the series. For a time series $\{y_t : t = 1, \ldots, T\}$, the natural logarithm is: $z_t = \ln y_t$. Some of the benefits of such a transformation are that:

- they are easier to interpret (relative/percentage change).
- they homogenizes the variance of the time series.
- they may result in improved forecasting accuracy.

Exponential trends are suitable when a time series is characterized by a stable relative change over time (e.g., when economic time series grow by 2% every year).

We will cover the shifting/switching trend models in another chapter.

Trends are (relatively) easy to model and forecast. Caution is needed, however, with (higher order) polynomial trends, as they may fit well in-sample, but cause major problems out-of-sample.

Consider a linear trend model with an additive error term:

$$y_t = \alpha + \beta t + \varepsilon_t$$

We estimate the model parameters, $\theta = \{\alpha, \beta\}$, by fitting the trend model to a time series using the least-squares regression:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \sum_{t=1}^{T} \Big(y_t - \alpha - \beta t \Big)^2.$$

Fitted values are then given by:

$$\hat{y}_t = \hat{\alpha} + \hat{\beta} t$$

4.3 Forecasting

If a linear trend model is fitted to the data, then any future realization of the stochastic process is assumed to follow the linear trend model:

$$y_{t+h} = \alpha + \beta(t+h) + \varepsilon_{t+h}.$$

An optimal forecast of y_{t+h}, therefore, is given by:

$$y_{t+h|t} = E(y_{t+h}|\Omega_t) = E[\alpha + \beta(t+h) + \varepsilon_{t+h}] = \alpha + \beta(t+h).$$

The forecast error is:

$$e_{t+h|t} = y_{t+h} - y_{t+h|t} = \varepsilon_{t+h}$$

The forecast variance, then, is:

$$\sigma^2_{t+h|t} = E(e^2_{t+h|t}) = E(\varepsilon^2_{t+h}) = \hat{\sigma}^2, \quad \forall\ h$$

From this, we can obtain interval forecast at any horizon, which is:

$$y_{t+h|t} \pm 1.96\hat{\sigma}.$$

A few features of trend forecasts to note:

- they tend to understate uncertainty (at long horizons as the forecast interval doesn not widen with the horizon);
- short-term trend forecasts can perform poorly; long-term trend forecasts typically perform poorly;
- sometimes it may be beneficial, from the standpoint of achieving better accuracy, to forecast growth rates, and then reconstruct level forecasts.

Chapter 5

Seasonality

Seasonality is typically modeled as monthly or quarterly pattern, but can also be modeled as a higher frequency pattern (e.g. weekly). Some examples of time series with apparent seasonal patterns are:

- Agricultural production.
- Sales of energy products.
- Airfare (in non-pandemic times).

One way to deal with the seasonality in data is to "remove" it prior to the use of the series (i.e., work with a seasonally adjusted time series). Indeed, some economic time series are only/also available in a seasonally-adjusted form.

Otherwise, and perhaps more interestingly, we can directly model seasonality in a regression setting by incorporating seasonal dummy variables, for example.

5.1 Modeling

A seasonal model is given by:

$$y_t = \sum_{i=1}^{s} \gamma_i d_{it} + \varepsilon_t,$$

where s denotes the frequency of the data, and d_{it} takes the value of 1 repeatedly after every s periods, and such that $\sum_i d_{it} = 1$, $\forall t$.

Alternatively the seasonal model can be rewritten as:

$$y_t = \alpha + \sum_{i=1}^{s-1} \delta_i d_{it} + \varepsilon_t,$$

in which case α is an intercept of an omitted season, and δ_i represents a deviation from it during the i^{th} season. This is a more typical form of a seasonal model.

Both variants of a seasonal model result in an identical fit and forecasts.

When dealing with weekly or daily data, the dummy variable approach of modeling seasonality may not be practical, nor efficient in most instances, as that will require estimating another 51 or 364 parameters. A way to model seasonality without giving up too many degrees of freedom is by using the so-called harmonic seasonal variables, which are a set of Fourier terms.

The Fourier terms can be applied to model seasonality at any frequency, indeed. Suppose, for example, we are working with monthly time series. A model with Fourier terms will have the following form:

$$y_t = \alpha + \sum_{k=1}^{K} \left[\beta_{1k} \sin\left(\frac{2\pi kt}{12}\right) + \beta_{2k} \cos\left(\frac{2\pi kt}{12}\right) \right] + \varepsilon_t,$$

where the value of K can be determined using an information criterion (e.g., AIC or SIC).

5.2 Forecasting

The predictors of the seasonal models are pre-determined, which means, after fitting the model, we can directly obtain the point and interval forecasts for any horizon h.

When using the dummy variable approach to model the seasonality, for example, a future realization of a random variable is:

$$y_{t+h} = \alpha + \sum_{i=1}^{s-1} \delta_i d_{i,t+h} + \varepsilon_{t+h}.$$

The optimal forecast of y_{t+h} is:

$$y_{t+h|t} = E(y_{t+h}|\Omega_t) = \alpha + \sum_{i=1}^{s-1} \delta_i d_{i,t+h}$$

The forecast error is:

$$e_{t+h|t} = y_{t+h} - y_{t+h|t} = \varepsilon_{t+h}$$

The forecast variance is:

$$\sigma^2_{t+h|t} = E(e^2_{t+h|t}) = E(\varepsilon^2_{t+h}) = \hat{\sigma}^2, \quad \forall\, h$$

The interval forecast is:

$$y_{t+h|t} \pm 1.96\hat{\sigma}.$$

Dynamic Time Series Models

Chapter 6

Linear Autoregression

Economic time series are often characterized by stochastic cycles. A cycle is a pattern of periodic fluctuations, not contained within a calendar year. A stochastic cycle is one generated by random variables. In general terms, the process is given by:

$$Y_t = f(Y_{t-1}, Y_{t-2}, \ldots; \theta) + \varepsilon_t. \quad t = 1, \ldots, T$$

An autoregressive process (or, simply, an autoregression) is a regression in which the dependent variable and the regressors belong to the same stochastic process.

An autoregression of order p, denoted as $AR(p)$, has the following functional form:

$$y_t = \alpha + \beta_1 y_{t-1} + \beta_2 y_{t-2} + \cdots + \beta_p y_{t-p} + \varepsilon_t$$

The sum of the autoregressive parameters, $\beta_1, \ldots, \beta_p$, depicts the persistence of the series. The larger is the persistence (i.e., closer it is to one), the longer it takes for the effect of a shock to dissolve. The effect will, eventually, dissolve so long as the series are covariance-stationary.

The autocorrelation, ρ, and partial autocorrelation, π, functions of the covariance-stationary $AR(p)$ process have the following distinctive features:

- $\rho_1 = \pi_1$, and $\pi_p = \beta_p$.

- The values of $\beta_1, \ldots, \beta_p$ determine the shape of the autocorrelation function (ACF); in any case, the smaller (in absolute terms) is the persistence measure, the faster the ACF decays toward zero.
- The partial autocorrelation function (PACF) is characterized by "statistically significant" first p spikes $\pi_1 \neq 0, \ldots, \pi_p \neq 0$, and the remaining $\pi_k = 0$, $\forall k > p$.

6.1 Modeling

6.1.1 AR(1)

The first-order autoregression is given by:

$$y_t = \alpha + \beta_1 y_{t-1} + \varepsilon_t,$$

where α is a constant term; β_1 is the *persistence* parameter; and ε_t is a white noise process.

A necessary and sufficient condition for an $AR(1)$ process to be covariance stationary is that $|\beta_1| < 1$. We can see this by substituting recursively the lagged equations into the lagged dependent variables:

$$\begin{aligned}
y_t &= \alpha + \beta_1 y_{t-1} + \varepsilon_t \\
y_t &= \alpha + \beta_1(\alpha + \beta_1 y_{t-2} + \varepsilon_{t-1}) + \varepsilon_t \\
&= \alpha(1 + \beta_1) + \beta_1^2(\alpha + \beta_1 y_{t-3} + \varepsilon_{t-2}) + \beta_1 \varepsilon_{t-1} + \varepsilon_t \\
&\vdots \\
&= \alpha \sum_{i=0}^{k-1} \beta_1^i + \beta_1^k y_{t-k} + \sum_{i=0}^{k-1} \beta_1^i \varepsilon_{t-i}
\end{aligned}$$

The end-result is a general linear process with geometrically declining coefficients. Here, $|\beta_1| < 1$ is required for convergence.

Assuming $|\beta_1| < 1$, as $k \to \infty$ the process converges to:

$$y_t = \frac{\alpha}{1 - \beta_1} + \sum_{i=0}^{\infty} \beta_1^i \varepsilon_{t-i}$$

The *unconditional mean* of this process is:

$$\mu = E(y_t) = E\left(\frac{\alpha}{1 - \beta_1} + \sum_{i=0}^{\infty} \beta_1^i \varepsilon_{t-i}\right) = \frac{\alpha}{1 - \beta_1}$$

The *unconditional variance* of this process is:

$$\gamma_0 = Var\left(y_t\right) = Var\left(\frac{\alpha}{1-\beta_1} + \sum_{i=0}^{\infty} \beta_1^i \varepsilon_{t-i}\right) = \frac{\sigma_\varepsilon^2}{1-\beta_1^2}$$

The *Autocovariance* is simply the covariance between y_t and y_{t-k}, that is:

$$\gamma_k = Cov(y_t, y_{t-k}) = E[(y_t - \mu)(y_{t-k} - \mu)] = E(y_t y_{t-k}) - \mu^2$$

Some algebraic manipulation can help us show that:

$$\gamma_k = \beta_1 \gamma_{k-1},$$

and that:

$$\rho_k = \beta_1 \rho_{k-1}$$

(recall, $\rho_k = \gamma_k/\gamma_0$ is the autocorrelation coefficient).

In fact, for AR(1), an autocorrelation coefficient of some lag can be represented as the autoregression parameter (which in this instance is equivalent to the persistence measure) to that power. That is:

$$\begin{aligned}
\rho_1 &= \beta_1 \rho_0 = \beta_1 \\
\rho_2 &= \beta_1 \rho_1 = \beta_1^2 \\
&\vdots \\
\rho_k &= \beta_1 \rho_{k-1} = \beta_1^k
\end{aligned}$$

It follows that the autocorrelation function of a covariance stationary AR(1) is a geometric decay; the smaller is $|\beta_1|$ the more rapid is the decay.

By imposing certain restrictions, the AR(1) will reduce to other already known models:

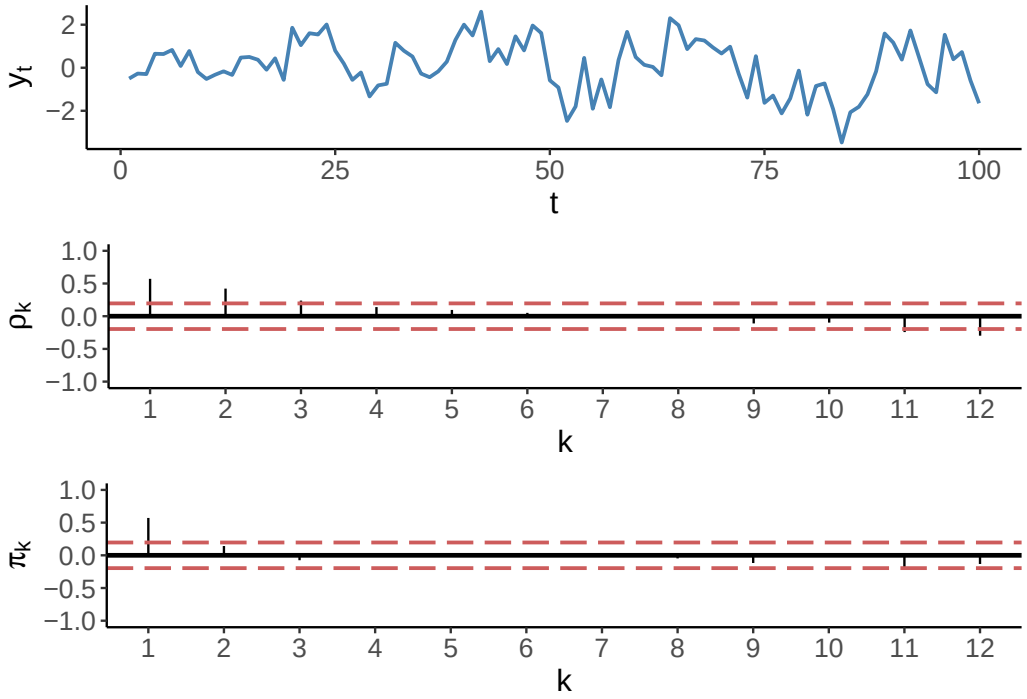
- If $\beta_1 = 0$, y_t is equivalent to a white noise.
- If $\beta_1 = 1$ and $\alpha = 0$, y_t is a random walk.
- If $\beta_1 = 1$ and $\alpha \neq 0$, y_t is a random walk with drift.

In general, a smaller persistence parameter results in a quicker adjustment to the *unconditional mean* of the process.

The autocorrelation and partial autocorrelation functions of the AR(1) process have three distinctive features:

- $\rho_1 = \pi_1 = \beta_1$. That is, the persistence parameter is also the autocorrelation and the partial autocorrelation coefficient.
- The autocorrelation function decreases exponentially toward zero, and the decay is faster when the persistence parameter is smaller.
- The partial autocorrelation function is characterized by only one spike $\pi_1 \neq 0$, and the remaining $\pi_k = 0$, $\forall k > 1$.

To illustrate the foregoing, let's generate a series of 100 observations that follow the process: $y_t = 0.8y_{t-1} + \varepsilon_t$, where $y_0 = 0$ and $\varepsilon \sim N(0, 1)$, and plot the ACF and PACF of this series.



6.1.2 AR(2)

Now consider the second-order autoregression:

$$y_t = \alpha + \beta_1 y_{t-1} + \beta_2 y_{t-2} + \varepsilon_t$$

where α is a constant term; $\beta_1 + \beta_2$ is the persistence measure; and ε_t is a white noise process.

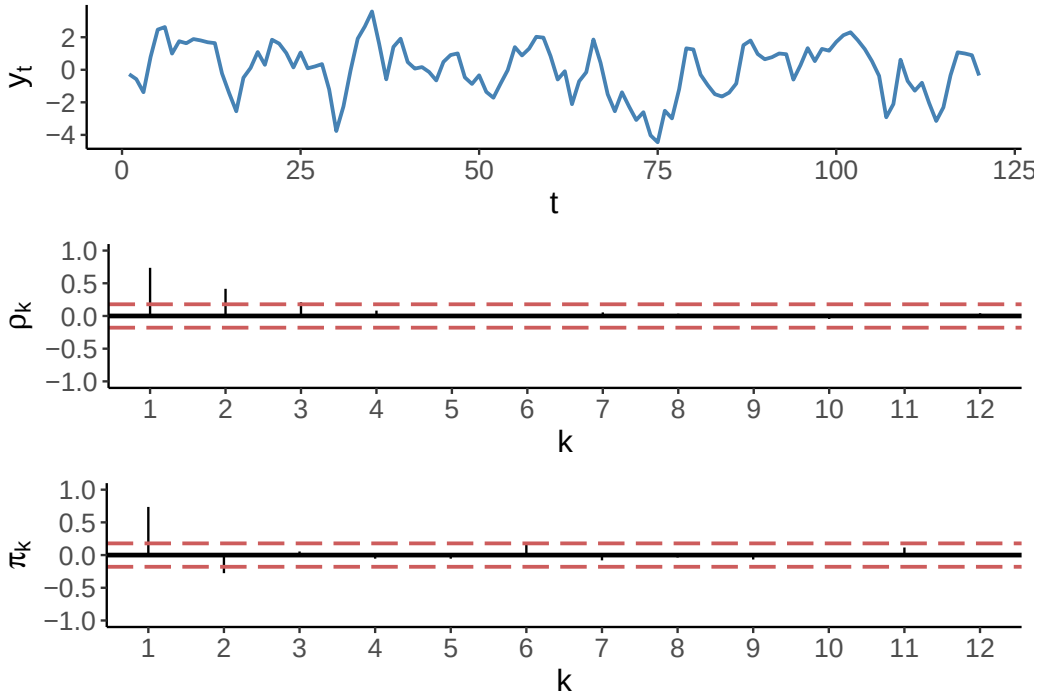
In what follows, the necessary (1 and 2) and sufficient (3 and 4) conditions for an $AR(2)$ process to be covariance stationary are:

1. $|\beta_2| < 1$
2. $|\beta_1| < 2$
3. $\beta_1 + \beta_2 < 1$
4. $\beta_2 - \beta_1 < 1$

The autocorrelation functions of the AR(2) process have the following distinctive features:

- $\rho_1 = \pi_1$ (which is true for any $AR(p)$ process), and $\pi_2 = \beta_2$.
- The autocorrelation function decreases toward zero. The path, however, varies depending on the values of β_1 and β_2. Nonetheless, the decay is faster when the persistence measure is smaller.
- The partial autocorrelation function is characterized by only two spikes $\pi_1 \neq 0$ and $\pi_2 \neq 0$, and the remaining $\pi_k = 0$, $\forall k > 2$.

Again, to illustrate, let's generate a series of 100 observations that follow the process: $y_t = 1.1y_{t-1} - 0.4y_{t-2} + \varepsilon_t$, where $y_{-1} = y_0 = 0$ and $\varepsilon \sim N(0, 1)$, and plot the ACF and PACF of this series.



6.2 Forecasting

Making forecasts for some future period, $t + h$, from an AR(p) model that has been fit to the data up to and including period $t + h - 1$ can be a straightforward exercise, so long as we have access to the relevant information set. For one-step-ahead forecasts, the information set is readily available. For multi-step-ahead forecasts, we need to 'come up' with the value of the variable that has not been realized yet. For example, when making a two-step-ahead forecast for period $t+2$, we need data from period $t+1$, which is not available at the time when the forecast is made. Instead, we need to use our forecast for period $t + 1$. The same applies to forecasts for any subsequent periods in the future. This approach is known as an *iterative* method of forecasting, wherein we make forecast for some period using the available data, then iterate forward by one period and use the most recent forecast to make the next period's forecast, and so on and so forth.

6.2.1 One-step-ahead forecast from AR(1)

The realization of the random variable in period $t+1$ is:

$$y_{t+1} = \alpha + \beta_1 y_t + \varepsilon_{t+1}$$

The optimal one-step-ahead forecast is:

$$y_{t+1|t} = E(y_{t+1}|\Omega_t) = E(\alpha + \beta_1 y_t + \varepsilon_{t+1}) = \alpha + \beta_1 y_t$$

The one-step-ahead forecast error is:

$$e_{t+1|t} = y_{t+1} - y_{t+1|t} = \alpha + \beta_1 y_t + \varepsilon_{t+1} - (\alpha + \beta_1 y_t) = \varepsilon_{t+1}$$

The one-step-ahead forecast variance:

$$\sigma^2_{t+1|t} = Var(y_{t+1}|\Omega_t) = E(e^2_{t+1|t}) = E(\varepsilon^2_{t+1}) = \sigma^2_\varepsilon$$

The one-step-ahead (95%) interval forecast:

$$y_{t+1|t} \pm z_{.025}\sigma_{t+1|t} = y_{t+1|t} \pm 1.96\sigma_\varepsilon$$

6.2.2 Two-step-ahead forecast from AR(1)

The realization of the random variable in period $t+2$ is:

$$y_{t+2} = \alpha + \beta_1 y_{t+1} + \varepsilon_{t+2}$$

The optimal two-step-ahead forecast is:

$$y_{t+2|t} = E(y_{t+2}|\Omega_t) = E(\alpha + \beta_1 y_{t+1} + \varepsilon_{t+2}) = \alpha(1+\beta_1) + \beta_1^2 y_t$$

Note, that here we substituted y_{t+1} with $\alpha + \beta_1 y_t + \varepsilon_{t+1}$.

The two-step-ahead forecast error is:

$$\begin{aligned} e_{t+2|t} &= y_{t+2} - y_{t+2|t} \\ &= \alpha(1+\beta_1) + \beta_1^2 y_t + \beta_1 \varepsilon_{t+1} + \varepsilon_{t+2} - [\alpha(1+\beta_1) + \beta_1^2 y_t] \\ &= \beta_1 \varepsilon_{t+1} + \varepsilon_{t+2} \end{aligned}$$

The two-step-ahead forecast variance is:

$$\begin{aligned}\sigma^2_{t+2|t} &= Var(y_{t+2}|\Omega_t) \\ &= E(e^2_{t+2|t}) = E(\beta_1\varepsilon_{t+1} + \varepsilon_{t+2})^2 = \sigma^2_\varepsilon(1+\beta_1^2)\end{aligned}$$

The two-step-ahead (95%) interval forecast:

$$y_{t+2|t} \pm z_{.025}\sigma_{t+2|t} = y_{t+2|t} \pm 1.96\sigma_\varepsilon\sqrt{1+\beta_1^2}$$

6.2.3 h-step-ahead forecast from AR(1)

The realization of the random variable in period $t+h$ is:

$$y_{t+h} = \alpha + \beta_1 y_{t+h-1} + \varepsilon_{t+h}$$

The optimal h-step-ahead forecast:

$$y_{t+h|t} = E(y_{t+h}|\Omega_t) = E(\alpha + \beta_1 y_{t+h-1} + \varepsilon_{t+1}) = \alpha\sum_{j=0}^{h-1}\beta_1^j + \beta_1^h y_t$$

The h-step-ahead forecast error:

$$e_{t+h|t} = y_{t+h} - y_{t+h|t} = \sum_{j=0}^{h-1}\beta_1^j\varepsilon_{t+h-j}$$

The h-step-ahead forecast variance:

$$\sigma^2_{t+h|t} = Var(y_{t+h}|\Omega_t) = E(e^2_{t+h|t}) = \sigma^2_\varepsilon\sum_{j=0}^{h-1}\beta_1^{2j}$$

The h-step-ahead (95%) interval forecast:

$$y_{t+h|t} \pm z_{.025}\sigma_{t+h|t} = y_{t+1|t} \pm 1.96\sigma_\varepsilon\sqrt{\sum_{j=0}^{h-1}\beta_1^{2j}}$$

If the series represent a covariance-stationary process, i.e. when $|\beta_1| < 1$, as $h \to \infty$:

The optimal point forecast converges to:

$$y_{t+h|t} = \frac{\alpha}{1-\beta_1}$$

The forecast variance converges to:

$$\sigma^2_{t+h|t} = \frac{\sigma^2_\varepsilon}{1-\beta_1^2}$$

The (95%) interval forecast converges to:

$$y_{t+h|t} \pm z_{.025}\sigma_{t+h|t} = \frac{\alpha}{1-\beta_1} \pm 1.96\frac{\sigma_\varepsilon}{\sqrt{1-\beta_1^2}}$$

6.2.4 One-step-ahead forecast from AR(2)

The realization of the random variable in period $t+1$ is:

$$y_{t+1} = \alpha + \beta_1 y_t + \beta_2 y_{t-1} + \varepsilon_{t+1}$$

The optimal one-step-ahead forecast:

$$\begin{aligned} y_{t+1|t} &= E(y_{t+1}|\Omega_t) \\ &= E(\alpha + \beta_1 y_t + \beta_2 y_{t-1} + \varepsilon_{t+1}) = \alpha + \beta_1 y_t + \beta_2 y_{t-1} \end{aligned}$$

The one-step-ahead forecast error:

$$\begin{aligned} e_{t+1|t} &= y_{t+1} - y_{t+1|t} \\ &= \alpha + \beta_1 y_t + \beta_2 y_{t-1} + \varepsilon_{t+1} - (\alpha + \beta_1 y_t + \beta_2 y_{t-1}) = \varepsilon_{t+1} \end{aligned}$$

The one-step-ahead forecast variance:

$$\sigma^2_{t+1|t} = Var(y_{t+1}|\Omega_t) = E(e^2_{t+1|t}) = E(\varepsilon^2_{t+1}) = \sigma^2_\varepsilon$$

The one-step-ahead (95%) interval forecast:

$$y_{t+1|t} \pm z_{.025}\sigma_{t+1|t} = y_{t+1|t} \pm 1.96\sigma_\varepsilon$$

6.2.5 Two-step-ahead forecast from AR(2)

The realization of the random variable in period $t+2$ is:

$$y_{t+2} = \alpha + \beta_1 y_{t+1} + \beta_2 y_t + \varepsilon_{t+2}$$

The optimal two-step-ahead forecast:

$$\begin{aligned} y_{t+2|t} = E(y_{t+2}|\Omega_t) &= E(\alpha + \beta_1 y_{t+1} + \beta_2 y_t + \varepsilon_{t+1}) \\ &= \alpha(1+\beta_1) + (\beta_1^2 + \beta_2) y_t + \beta_1 \beta_2 y_{t-1} \end{aligned}$$

The two-step-ahead forecast error:

$$\begin{aligned} e_{t+2|t} = y_{t+2} - y_{t+2|t} =& \alpha + \beta_1 y_{t+1} + \beta_2 y_t + \varepsilon_{t+2} \\ & - (\alpha + \beta_1 y_{t+1|t} + \beta_2 y_t) = \beta_1 \varepsilon_{t+1} + \varepsilon_{t+2} \end{aligned}$$

The two-step-ahead forecast variance:

$$\sigma^2_{t+2|t} = Var(y_{t+2}|\Omega_t) = E(e^2_{t+2|t}) = E(\beta_1 \varepsilon_{t+1} + \varepsilon_{t+2})^2 = \sigma^2_\varepsilon (1 + \beta_1^2)$$

The two-step-ahead (95%) interval forecast:

$$y_{t+2|t} \pm z_{.025} \sigma_{t+2|t} = y_{t+2|t} \pm 1.96 \sigma_\varepsilon \sqrt{1 + \beta_1^2}$$

6.2.6 One-step-ahead forecast from AR(2)

The realization of the random variable in period $t+h$ is:

$$y_{t+h} = \alpha + \beta_1 y_{t+h-1} + \beta_2 y_{t+h-2} + \varepsilon_{t+h}$$

The optimal h-step-ahead forecast (iterated method):

$$\begin{aligned} y_{t+1|t} &= \alpha + \beta_1 y_t + \beta_2 y_{t-1} \\ y_{t+2|t} &= \alpha + \beta_1 y_{t+1|t} + \beta_2 y_t \\ y_{t+3|t} &= \alpha + \beta_1 y_{t+2|t} + \beta_2 y_{t+1|t} \\ &\vdots \\ y_{t+h|t} &= \alpha + \beta_1 y_{t+h-1|t} + \beta_2 y_{t+h-2|t} \end{aligned}$$

The h-step-ahead forecast error:

$$e_{t+h|t} = y_{t+h} - y_{t+h|t} = \varepsilon_{t+h} + \beta_1 e_{t+h-1|t} + \beta_2 e_{t+h-2|t}$$

The h-step-ahead forecast variance:

$$\begin{aligned}\sigma^2_{t+h|t} &= Var(y_{t+h}|\Omega_t) = E(e^2_{t+h|t}) \\ &= \sigma^2_\varepsilon + \beta_1^2 Var(e_{t+h-1|t}) + \beta_2^2 Var(e_{t+h-2|t}) \\ &+ 2\beta_1\beta_2 Cov(e_{t+h-1|t}, e_{t+h-2|t})\end{aligned}$$

(Note: the formulas for $\sigma^2_{t+1|t}, \sigma^2_{t+2|t}, \ldots, \sigma^2_{t+h|t}$ are the same for any $AR(p)$, $p \geq h-1$).

The h-step-ahead (95%) interval forecast:

$$y_{t+h|t} \pm z_{.025}\sigma_{t+h|t} = y_{t+1|t} \pm 1.96\sigma_{t+h|t}$$

The optimal h-step-ahead forecast:

$$y_{t+h|t} = E(y_{t+h}|\Omega_t) = \alpha + \beta_1 y_{t+h-1|t} + \beta_2 y_{t+h-2|t} + \cdots + \beta_p y_{t+h-p|t}$$

The h-step-ahead forecast error:

$$e_{t+h|t} = \varepsilon_{t+h} + \beta_1 e_{t+h-1|t} + \beta_2 e_{t+h-2|t} + \cdots + \beta_p e_{t+h-p|t}$$

The h-step-ahead forecast variance:

$$\begin{aligned}\sigma^2_{t+h|t} &= Var(y_{t+h}|\Omega_t) = E(e^2_{t+2|t}) \\ &= \sigma^2_\varepsilon + \sum_{i=1}^{p} \beta_i^2 Var(e_{t+h-i|t}) + 2\sum_{i\neq j} \beta_i\beta_j Cov(e_{t+h-i|t}, e_{t+h-j|t})\end{aligned}$$

The h-step-ahead (95%) interval forecast:

$$y_{t+h|t} \pm z_{.025}\sigma_{t+h|t} = y_{t+h|t} \pm 1.96\sigma_{t+h|t}$$

Chapter 7

Vector Autoregression

Many economic variables are interrelated. For example, changes to household income impact their consumption levels; changes to interest rates impact investments in the economy. Often (albeit not always) the relationship between the variables goes in both directions. For example, higher wages (and, therefore, income) result in higher prices (inflation), which in turn puts an upward pressure on wages.

The foregoing implies that a shock to a variable may propagate a dynamic response not only of that variable, but also of related variables. The dynamic linkages between two (or more) economic variables can be modeled as a *system of equations*, represented by a vector autoregressive (VAR) process.

7.1 Modeling

To begin, consider a bivariate (two-dimensional) VAR of order one, VAR(1).

Let $\{X_{1,t}\}$ and $\{X_{2,t}\}$ be the stationary stochastic processes. A bivariate VAR(1), is then given by:

$$x_{1,t} = \alpha_1 + \pi_{11}x_{1,t-1} + \pi_{12}x_{2,t-1} + \varepsilon_{1,t}$$
$$x_{2,t} = \alpha_2 + \pi_{21}x_{1,t-1} + \pi_{22}x_{2,t-1} + \varepsilon_{2,t}$$

where $\varepsilon_{1,t} \sim iid(0, \sigma_1^2)$ and $\varepsilon_{2,t} \sim iid(0, \sigma_2^2)$, and the two can be correlated, i.e., $Cov(\varepsilon_{1,t}, \varepsilon_{2,t}) \neq 0$.

To generalize, consider a multivariate (n-dimensional) VAR of order p, VAR(p), presented in matrix notation:

$$\mathbf{x}_t = \alpha + \Pi_1 \mathbf{x}_{t-1} + \ldots + \Pi_p \mathbf{x}_{t-p} + \varepsilon_t,$$

where $\mathbf{x}_t = (x_{1,t}, \ldots, x_{n,t})'$ is a vector of n (potentially) related variables; $\varepsilon_t = (\varepsilon_{1,t}, \ldots, \varepsilon_{n,t})'$ is a vector of error terms, such that $E(\varepsilon_t) = \mathbf{0}$, $E(\varepsilon_t \varepsilon_t') = \Sigma$, and $E(\varepsilon_t \varepsilon_{s \neq t}') = 0$. $\Pi_1, \ldots, \Pi_p$ are n-dimensional parameter matrices:

$$\Pi_j = \begin{bmatrix} \pi_{11j} & \pi_{12j} & \cdots & \pi_{1nj} \\ \pi_{21j} & \pi_{22j} & \cdots & \pi_{2nj} \\ \vdots & \vdots & \ddots & \vdots \\ \pi_{n1j} & \pi_{n2j} & \cdots & \pi_{nnj} \end{bmatrix}, \quad j = 1, \ldots, p$$

When two or more variables are modeled in this way, the implies assumption is that these variables are endogenous to each other; that is, each of the variables affects and is affected by other variables.

There are three forms of vector autoregressions: structural, recursive, and reduced-form. The structural VAR uses economic theory to impose the 'structure' on correlations of the error terms in the system, thus, facilitate their 'causal' interpretation. The recursive VAR also introduces a structure of some sort, which primarily involves ordering the equations in the system in a specific way so that the error terms in each equation are uncorrelated with those in the preceding equations. To the extent that the 'identifying assumptions' are satisfied, some contemporaneous values (of other variables) appear in the equation of a given variable. The reduced-form VAR makes no claims of causality, instead it only includes lagged values of all the variables in each equation of the system. To the extent that the variables entering the system are, indeed, correlated among each other, the error terms of the reduced-form VAR (typically) are contemporaneously correlated. Here, unless otherwise specified, VAR refers to the reduced-form VAR.

Some of the features of the VAR are that:

- only lagged values of the dependent variables are considered as the right-hand-side variables (although, trend and seasonal variables might also be included in higher-frequency data analysis);
- each equation has the same set of right-hand-side variables (however, it is possible to impose a different lag structure across the equations,

especially when p is relatively large, to preserve the degrees of freedom, particularly when the sample size is relatively small and when there are several variables in the system).
- The autregressive order of the VAR, p, is determined by the maximum lag of a variable across all equations.

The order of a VAR, p, can be determined using system-wide information criteria:

$$AIC = \ln|\Sigma_\varepsilon| + \frac{2}{T}(pn^2 + n)$$
$$SIC = \ln|\Sigma_\varepsilon| + \frac{\ln T}{T}(pn^2 + n)$$

where $|\Sigma_\varepsilon|$ is the determinant of the residual covariance matrix; n is the number of equations, and T is the effective sample size.

We can estimate each equation of the VAR individually using OLS.

7.1.1 In-Sample Granger Causality

A test of joint significance of parameters associated with all the lags of a variable entering the equation of another variable is known as the 'Granger causality' test. The use of the term 'causality' in this context has been criticized. That one variable can help explain the movement of another variable does not necessarily mean that the former causes the latter. To that end the use of the term is misleading, indeed. Rather, it simply means that the former helps predict the latter, and that is, in effect, the meaning of causality in Granger sense.

To illustrate the testing framework, consider a bivariate VAR(p):

$$\begin{aligned} x_{1,t} &= \alpha_1 + \pi_{111}x_{1,t-1} + \cdots + \pi_{11p}x_{1,t-p} \\ &\quad + \pi_{121}x_{2,t-1} + \cdots + \pi_{12p}x_{2,t-p} + \varepsilon_{1,t} \\ x_{2,t} &= \alpha_1 + \pi_{211}x_{1,t-1} + \cdots + \pi_{21p}x_{1,t-p} \\ &\quad + \pi_{221}x_{2,t-1} + \cdots + \pi_{22p}x_{2,t-p} + \varepsilon_{2,t} \end{aligned}$$

It is said that: - $\{X_2\}$ does not Granger cause $\{X_1\}$ if $\pi_{121} = \cdots = \pi_{12p} = 0$ - $\{X_1\}$ does not Granger cause $\{X_2\}$ if $\pi_{211} = \cdots = \pi_{21p} = 0$

So long as the variables of the system are covariance-stationarity, we can apply a F test for the hypotheses testing. If $p = 1$, a t test is equivalently applicable for hypotheses testing.

7.2 Forecasting

Generating forecasts for each of the variable comprising the VAR(p) model can be a straightforward exercise, so long as we have access to the relevant information set. As it was the case with autoregressive models, we make one-step-ahead forecasts based on the readily available data; and we make multi-step-ahead forecasts iteratively, using the forecast in periods for which the data are not present.

7.2.1 One-step-ahead forecast from bivariate VAR(1)

The realizations of the variables in period $t + 1$ are:

$$x_{1,t+1} = \alpha_1 + \pi_{11}x_{1,t} + \pi_{12}x_{2,t} + \varepsilon_{1,t+1}$$
$$x_{2,t+1} = \alpha_2 + \pi_{21}x_{1,t} + \pi_{22}x_{2,t} + \varepsilon_{2,t+1}$$

The optimal one-step-ahead forecasts are:

$$x_{1,t+1|t} = E(x_{1,t+1}|\Omega_t) = \alpha_1 + \pi_{11}x_{1,t} + \pi_{12}x_{2,t}$$
$$x_{2,t+1|t} = E(x_{2,t+1}|\Omega_t) = \alpha_2 + \pi_{21}x_{1,t} + \pi_{22}x_{2,t}$$

The one-step-ahead forecast errors are:

$$e_{1,t+1|t} = x_{1,t+1} - x_{1,t+1|t} = \varepsilon_{1,t+1}$$
$$e_{2,t+1|t} = x_{2,t+1} - x_{2,t+1|t} = \varepsilon_{2,t+1}$$

The one-step-ahead forecast variances are:

$$\sigma^2_{1,t+1|t} = E(x_{1,t+1} - x_{1,t+1|t}|\Omega_t)^2 = E(\varepsilon^2_{1,t+1}) = \sigma^2_1$$
$$\sigma^2_{2,t+1|t} = E(x_{2,t+1} - x_{2,t+1|t}|\Omega_t)^2 = E(\varepsilon^2_{2,t+1}) = \sigma^2_2$$

The one-step-ahead (95%) interval forecasts are:

$$x_{1,t+1|t} \pm z_{.025}\sigma_{1,t+1|t} = x_{1,t+1|t} \pm 1.96\sigma_{\varepsilon_1}$$
$$x_{2,t+1|t} \pm z_{.025}\sigma_{2,t+1|t} = x_{2,t+1|t} \pm 1.96\sigma_{\varepsilon_2}$$

7.2.2 Two-step-ahead forecast from bivariate VAR(1)

The realizations of the variables in period $t+2$ are:

$$x_{1,t+2} = \alpha_1 + \pi_{11}x_{1,t+1} + \pi_{12}x_{2,t+1} + \varepsilon_{1,t+2}$$
$$x_{2,t+2} = \alpha_2 + \pi_{21}x_{1,t+1} + \pi_{22}x_{2,t+1} + \varepsilon_{2,t+2}$$

The optimal two-step-ahead forecasts are:

$$x_{1,t+2|t} = E(x_{1,t+2}|\Omega_t) = \alpha_1 + \pi_{11}x_{1,t+1|t} + \pi_{12}x_{2,t+1|t}$$
$$x_{2,t+2|t} = E(x_{2,t+2}|\Omega_t) = \alpha_2 + \pi_{21}x_{1,t+1|t} + \pi_{22}x_{2,t+1|t}$$

The two-step-ahead forecast errors are:

$$e_{1,t+2|t} = x_{1,t+2} - x_{1,t+2|t} = \pi_{11}e_{1,t+1|t} + \pi_{12}e_{2,t+1|t} + \varepsilon_{1,t+2}$$
$$e_{2,t+2|t} = x_{2,t+2} - x_{2,t+2|t} = \pi_{21}e_{1,t+1|t} + \pi_{22}e_{2,t+1|t} + \varepsilon_{2,t+2}$$

The two-step-ahead forecast variances are:

$$\begin{aligned}\sigma^2_{1,t+2|t} &= E(x_{1,t+2} - x_{1,t+2|t}|\Omega_t)^2 \\ &= \sigma_1^2(1+\pi_{11}^2) + \sigma_2^2\pi_{12}^2 + 2\pi_{11}\pi_{12}Cov(\varepsilon_1,\varepsilon_2) \\ \sigma^2_{2,t+2|t} &= E(x_{2,t+2} - x_{2,t+2|t}|\Omega_t)^2 \\ &= \sigma_2^2(1+\pi_{22}^2) + \sigma_1^2\pi_{21}^2 + 2\pi_{21}\pi_{22}Cov(\varepsilon_1,\varepsilon_2)\end{aligned}$$

The two-step-ahead (95%) interval forecasts are:

$$x_{1,t+2|t} \pm z_{.025}\sigma_{1,t+2|t}$$
$$x_{2,t+2|t} \pm z_{.025}\sigma_{2,t+2|t}$$

7.2.3 Out-of-Sample Granger Causality

The previously discussed (in sample) test of causality in Granger sense is frequently performed in practice. But as noted above, the term 'causality' may be misleading somewhat. Indeed, the 'true spirit' of such test is to assess the ability of a variable to help predict another variable in an out-of-sample setting.

Consider the restricted and unrestricted information sets, where the former only contains information on variable X_1, while the latter contains the same information as well as information on variable X_2:

$$\Omega_{t,r} \equiv \Omega_t(X_1) = \{x_{1,t}, x_{1,t-1}, \ldots\}$$
$$\Omega_{t,u} \equiv \Omega_t(X_1, X_2) = \{x_{1,t}, x_{1,t-1}, \ldots, x_{2,t}, x_{2,t-1}, \ldots\}$$

Following Granger's definition of causality: $\{X_2\}$ is said to cause $\{X_1\}$ if $\sigma^2_{x_1}(\Omega_{t,u}) < \sigma^2_{x_1}(\Omega_{t,r})$, meaning that we can better predict X_1 using all available information on X_1 and X_2, rather than that on X_1 only.

Let the forecasts based on each of the information sets be:

$$x_{1,t+h|t,r} = E(x_{1,t+h}|\Omega_{t,r})$$
$$x_{1,t+h|t,u} = E(x_{1,t+h}|\Omega_{t,u})$$

For these forecasts, the corresponding forecast errors are:

$$e_{1,t+h|t,r} = x_{1,t+h} - x_{1,t+h|t,r}$$
$$e_{1,t+h|t,u} = x_{1,t+h} - x_{1,t+h|t,u}$$

The out-of-sample forecast errors are then evaluated by comparing the loss functions based on these forecasts errors. For example, assuming quadratic loss, and P out-of-sample forecasts:

$$RMSFE_r = \sqrt{\frac{1}{P}\sum_{s=1}^{P}\left(e_{1,R+s|R-1+s,r}\right)^2}$$
$$RMSFE_u = \sqrt{\frac{1}{P}\sum_{s=1}^{P}\left(e_{1,R+s|R-1+s,u}\right)^2}$$

where R is the size of the (first) estimation window.

$\{X_2\}$ is said to cause *in Granger sense* $\{X_1\}$ if $RMSFE_u < RMSFE_r$.

Chapter 8

Threshold Autoregression

While a stochastic process can be approximated by a linear model, it is possible that a nonlinear model offers a better fit to the data. Nonlinear models come in many flavours. Here we will consider one type of nonlinear models, which belongs to the family of regime-dependent models.

A regime-dependent model can be seen as a combination of linear specifications that are linked to each other in some (nonlinear) way. To that end, such nonlinear models are also referred to as the piecewise linear models - each piece in and of itself is linear, but when taken together we have a nonlinear model at hand.

8.1 Nonlinear Models

In what follows, we will consider two representative regime-dependent models: the time-varying threshold autoregression and the self-exciting threshold autoregression. In both instances we will assume that the switch between the regimes happens based on some threshold variable, and that it instantaneously.

Consider an AR(p) process with a deterministic trend:

$$y_t = \alpha_0 + \alpha_1 t + \sum_{i=1}^{p} \beta_i y_{t-i} + \varepsilon_t,$$

where $\alpha_0 + \alpha_1 t$ is the time-specific deterministic component.

This specification implies a linear trend, but that doesn't need to be the case. We can have quadratic or cubic trends, for example, or we can have no trend component at all.

A simple augmentation of the foregoing model is an autoregressive model with a switching trend component:

$$y_t = \delta_0 + \delta_1 t + \delta_2 (t - \tau) I(t > \tau) + \beta y_{t-1} + \varepsilon_t,$$

where τ is the threshold parameter.

Such switch can be extended to the whole autoregressive process. For example, a two-regime AR(p) with drift can be given by:

$$y_t = \delta_0 + \delta_1 t + \sum_{i=1}^{p} \beta_{1i} y_{t-i} + \left[\delta_2 (t - \tau) + \sum_{i=1}^{p} \beta_{2i} y_{t-i} \right] I(t > \tau) + \varepsilon_t.$$

This equation implies that not only the trend, but also the autoregressive process changes around the threshold parameter τ.

The foregoing nonlinear specifications assumed that the switch in the model occurs at some point in time, i.e. the regime-switching variable is a function of time. But the regime-switching variable can also be a function of the dependent variable, as well as other (potentially) related variables:

$$y_t = \alpha_0 + \sum_{i=1}^{p} \beta_{0i} y_{t-i} + \left(\alpha_1 + \sum_{i=1}^{p} \beta_{1i} y_{t-i} \right) I(s_t > c) + \varepsilon_t,$$

where s_t is the regime-switching variable, and c is the threshold, such that $\underline{s}_t < c < \overline{s}_t$, where $\underline{s}_t$ and $\overline{s}_t$ are lower and upper quantiles of the regime-switching variable.

This equation is referred as threshold autoregression, or TAR(p). More specifically, if in a TAR(p), $s_t = y_{t-d}$, $d = 1, \ldots, m$, then it is a self-exciting threshold autoregression, or SETAR(p); alternatively, if $s_t = \Delta y_{t-d}$, $d = 1, \ldots, m$, then the model is referred to as a momentum threshold autoregression, or momentum-TAR(p). The latter is typically preferred when y_t is an I(1) process.

A TAR (any version of it) can take a multiple-regime form:

$$y_t = \alpha_1 + \sum_{i=1}^{p} \beta_{1i} y_{t-i} + \sum_{j=2}^{K} \left(\alpha_j + \sum_{i=1}^{p} \beta_{ji} y_{t-i} \right) I(s_t > c_j) + \varepsilon_t,$$

where K depicts the number of regimes in the equation.

8.2 Modeling

When estimating TAR-type models, we have no *a priori* knowledge on the number of regimes, the autoregressive order in each regime, the regime-switching (or threshold) variable, and the threshold values.

When threshold values are unknown (and need to be estimated), standard statistical inference is no longer applicable. Otherwise, and given that the process is stationary, standard statistical inference applies.

Joint estimation of model parameters would require some nonlinear optimization routine. Alternatively, we can approximate such optimization using a grid-search procedure. The procedure relies on the fact that once the threshold parameter is known, the model reduces to a linear model, and the least squares estimator can be applied then. That is,

$$\hat{\tau} = \arg\min_{\tau} \hat{\sigma}^2(\tau),$$

where

$$\hat{\sigma}^2(\tau) = \frac{1}{T-k} \sum_{t=1}^{T} \hat{\epsilon}_t^2(\tau)$$

for all candidate values of τ. The candidate values of τ typically belong to a range between 10th and 90th percentiles of the transition variable, which is simply the trend in the case of the time-varying TAR, and the lagged dependent variable in the case of the self-exciting TAR.

8.3 Forecasting

In the case of time-varying shifting trend (mean) models, the most recent trend component is used to obtain forecasts. To that end, the forecasting routine is similar to that of linear trend models.

In the case of regime-switching models (e.g., TAR), obtaining one-step-ahead forecasts can be a rather straightforward exercise:

$$\begin{aligned} y_{t+1|t} = {} & \alpha_0 + \beta_{01} y_t + \beta_{02} y_{t-1} + \dots \\ & + (\alpha_1 + \beta_{11} y_t + \beta_{12} y_{t-1} + \dots) I(s_t > c) \end{aligned}$$

Obtaining h-step-ahead forecasts (where $h \geq 2$) is less trivial, however. Of the available options:

- The iterated method (or, the so-called skeleton extrapolation) is an easy but an inefficient option.
- The analytical method can be unbearably tedious.
- A numerical method is applicable and, moreover, it resolves the caveats of the previous two options.

8.3.1 Skeleton Extrapolation

One-step-ahead forecast:

$$y_{t+1|t} = E(y_{t+1}|\Omega_t) = g(y_t, y_{t-1}, \ldots, y_{t+1-p}; \theta)$$

Two-step-ahead forecast:

$$y_{t+2|t} = E(y_{t+2}|\Omega_t) = g(y_{t+1|t}, y_t, \ldots, y_{t+2-p}; \theta)$$

h-step-ahead forecast:

$$y_{t+h|t} = E(y_{t+h}|\Omega_t) = g(y_{t+h-1|t}, y_{t+h-2|t}, \ldots, y_{t+h-p|t}; \theta)$$

This is fine for linear models; but not okay for nonlinear models.

8.3.2 Analytical Method

One-step-ahead forecast is the same as before (no uncertainty about the observed data).

Two-step-ahead forecast is:

$$\tilde{y}_{t+2|t} = \int_{-\infty}^{\infty} g(y_{t+1|t} + \varepsilon_{t+1}, y_t, \ldots, y_{t+2-p}; \theta) f(\varepsilon_{t+1}) d\varepsilon_{t+1}$$

Unless the model is linear, $\tilde{y}_{t+2|t} \neq y_{t+2|t}$.

Longer horizon forecasts require multiple integrals.

8.3.3 Numerical Method: Bootstrap Resampling

Bootstrap resampling helps approximate the optimal forecast from nonlinear models and circumvents the complexity of integration.

As an additional benefit, the procedure generates forecast distribution, from which empirical confidence intervals (along with the point forecast) can be obtained.

Algorithm:

1. Estimate the time series model and store the residuals.
2. From this set of residuals, sample (with replacement) a vector of shocks for a bootstrap iteration, $\varepsilon^b = (\varepsilon^b_{t+1}, \varepsilon^b_{t+2}, \ldots, \varepsilon^b_{t+h})'$.
3. Use this sample of shocks, along with the estimated parameters and historical observations, to generate a forecast path for the given bootstrap iteration.
4. Repeat steps 2-3 many times to generate an empirical distribution of bootstrap forecasts.

One-step-ahead bootstrap iteration:

$$y_{t+1|t,\varepsilon^b_{t+1}} \equiv y^b_{t+1|t} = g(y_t, y_{t-1}, \ldots, y_{t+1-p}; \theta) + \varepsilon^b_{t+1}$$

Two-step-ahead bootstrap iteration:

$$y_{t+2|t,\varepsilon^b_{t+1},\varepsilon^b_{t+2}} \equiv y^b_{t+2|t} = g(y_{t+1|t,\varepsilon^b_{t+1}}, y_t, \ldots, y_{t+2-p}; \theta) + \varepsilon^b_{t+2}$$

For example, consider a linear AR(p) model.

One-step-ahead bootstrap iteration:

$$y^b_{t+1|t} = \alpha + \beta_1 y_t + \ldots + \beta_p y_{t+1-p} + \varepsilon^b_{t+1}$$

Two-step-ahead bootstrap iteration:

$$y^b_{t+2|t} = \alpha + \beta_1 \hat{y}^b_{t+1} + \ldots + \beta_p y_{t+2-p} + \varepsilon^b_{t+2}$$

Now consider a nonlinear SETAR(p, y_{t-1}) model:

One-step-ahead bootstrap iteration:

$$\begin{aligned} y^b_{t+1|t} = &(\alpha_1 + \beta_{11} y_t + \ldots + \beta_{1p} y_{t+1-p}) I(y_t \leq c) \\ &+ (\alpha_2 + \beta_{21} y_t + \ldots + \beta_{2p} y_{t+1-p}) I(y_t > c) + \varepsilon^b_{t+1} \end{aligned}$$

Two-step-ahead bootstrap iteration:

$$\begin{aligned} y^b_{t+2|t} = &(\alpha_1 + \beta_{11} y^b_{t+1|t} + \ldots + \beta_{1p} y_{t+2-p}) I(y^b_{t+1|t} \leq c) \\ &+ (\alpha_2 + \beta_{21} y^b_{t+1|t} + \ldots + \beta_{2p} y_{t+2-p}) I(y^b_{t+1|t} > c) + \varepsilon^b_{t+2} \end{aligned}$$

One-step-ahead bootstrap forecast:

$$\bar{y}_{t+1|t} = B^{-1} \sum_{b=1}^{B} y^b_{t+1|t}$$

Two-step-ahead bootstrap forecast:

$$\bar{y}_{t+2|t} = B^{-1} \sum_{b=1}^{B} y^b_{t+2|t}$$

Here, B is the total number of bootstrap iterations (usually many thousand iterations).

Forecast Assessment

Chapter 9

Forecast Evaluation

9.1 The Need for the Forecast Evaluation

We typically have several candidate econometric models or methods to forecast an economic variable of interest. Among these, we tend to select the most adequate using in-sample goodness of fit measures (e.g., AIC or SIC).

A more sensible approach, at least from the standpoint of a forecaster, would be to assess goodness of fit in an out-of-sample setting. Recall that models that offer the superior in-sample fit don't necessarily produce the most accurate out-of-sample forecasts. That is, while a better in-sample fit can be achieved by incorporating additional parameters in the model, such more complex models extrapolate the estimated parameter uncertainty into the forecasts, thus sabotaging their accuracy.

Thus far we have applied the following algorithm to identify 'the best' among the competing forecasts:

- Decide on a loss function (e.g., quadratic loss).
- Obtain forecasts, the forecast errors, and the corresponding sample expected loss (e.g., root mean squared forecast error) for each model in consideration.
- Rank the models according to their sample expected loss values.
- Select the model with the lowest sample expected loss.

But the loss function is a function of a random variable, and in practice we

deal with sample information, so sampling variation needs to be taken into the account. Statistical methods of evaluation are, therefore, desirable (at the very least).

9.2 Relative Forecast Accuracy Tests

Here we will cover two tests for the hypothesis that two forecasts are equivalent, in the sense that the associated loss differential is not statistically significantly different from zero.

Consider a time series of length T. Suppose h-step-ahead forecasts for periods $R+h$ through T have been generated from two competing models i and j: $y_{t+h|t,i}$ and $y_{t+h|t,j}$, with corresponding forecast errors: $e_{t+h,i}$ and $e_{t+h,j}$. The null hypothesis of equal predictive ability can be given in terms of the unconditional expectation of the loss differential:

$$H_0 : E\left[d(e_{t+h,ij})\right] = 0,$$

where $d(e_{t+h,ij}) = L(e_{t+h,i}) - L(e_{t+h,j})$.

9.2.1 The Morgan-Granger-Newbold Test

The Morgan-Granger-Newbold (MGN) test is based on auxiliary variables: $u_{1,t+h} = e_{t+h,i} - e_{t+h,j}$ and $u_{2,t+h} = e_{t+h,i} + e_{t+h,j}$. It follows that:

$$E(u_{1,t+h}, u_{2,t+h}) = MSFE(i, t+h) - MSFE(j, t+h).$$

Thus, the hypothesis of interest is equivalent to testing whether the two auxiliary variables are correlated.

The MGN test statistic is:

$$\frac{r}{\sqrt{(P-1)^{-1}(1-r^2)}} \sim t_{P-1},$$

where t_{P-1} is a Student t distribution with $P-1$ degrees of freedom, P is the number of out-of-sample forecasts, and

$$r = \frac{\sum_{t=R}^{T-h} u_{1,t+h} u_{2,t+h}}{\sqrt{\sum_{t=R}^{T-h} u_{1,t+h}^2 \sum_{t=R}^{T-h} u_{2,t+h}^2}}$$

The MGN test relies on the assumption that forecast errors (of the forecasts to be compared) are unbiased, normally distributed, and uncorrelated (with each other). These are rather strict assumptions that are, often, violated in empirical applications.

9.2.2 The Diebold-Mariano Test

The Diebold-Mariano (DM) test relaxes the aforementioned requirements on the forecast errors. The DM test statistic is:

$$\frac{\bar{d}_h}{\sqrt{\sigma_d^2/P}} \sim N(0,1),$$

where $\bar{d}_h = P^{-1} \sum_{t=R}^{T-h} d(e_{t+h,ij})$, and where $P = T - R - h + 1$ is the total number of forecasts generated using a rolling or recursive widnow scheme, for example.

A modified version of the DM statistic, due to Harvey, Leybourne, and Newbold (1998), addresses the finite sample properties of the test, so that:

$$\sqrt{\frac{P + 1 - 2h + P^{-1}h(h-1)}{P}} DM \sim t_{P-1},$$

where t_{P-1} is a Student t distribution with $P - 1$ degrees of freedom.

In practice, the test of equal predictive ability can be applied within the framework of a regression model as follows:

$$d_{t+h} = \delta + \upsilon_{t+h} \quad t = R, \ldots, T - h,$$

where $d_{t+h} \equiv d(e_{t+h,ij})$. The null of equal predictive ability is equivalent of testing $H_0 : \delta = 0$ in the OLS setting. Moreover, because d_{t+h} may be serially correlated, autocorrelation consistent standard errors should be used for inference.

Chapter 10

Forecast Combination

Tutorial 1: Introduction to R

R is a programming language for data analysis and visualisation. Here I introduce basic commands that should facilitate your understanding of R. You can further enhance your skillset using numerous online resources, as well as your own trial–and–error. To the extent that new features are added to R on a daily basis, there are virtually no limits to how far you can advance your knowledge of this programming language.

We will work in RStudio—the go-to interface for R (as R itself is not an overly user-friendly platform). Thus, you will need to have installed both, R and RStudio on your devise (the latter will 'find' and connect with the former on its own). R is available from CRAN, and RStudio is available from RStudio.

Data Management

There are a number of ways in which we can work with data in R. Let's begin with matrices.

Consider a string of observations:

```
a <- c(1,0,4,3,2,6)
a
```

```
## [1] 1 0 4 3 2 6
```

A *string*, unlike a *vector*, has no dimensions. But we can transform it to a $n \times 1$ vector using the as.matrix() function:

```
b <- as.matrix(a)
b
```

```
##      [,1]
## [1,]    1
## [2,]    0
## [3,]    4
## [4,]    3
## [5,]    2
## [6,]    6
```

The result is a 6×1 vector, or a column matrix. To obtain a 1×6 vector, or a row matrix, we *transpose* the foregoing vector using the t() function:

```
bt <- t(b)
bt
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    0    4    3    2    6
```

```
dim(bt)
```

```
## [1] 1 6
```

We can create any $n \times k$ matrix, using the matrix() function. For example, consider a 3×2 matrix:

```
B <- matrix(a,nrow=3,ncol=2)
B
```

```
##      [,1] [,2]
## [1,]    1    3
## [2,]    0    2
## [3,]    4    6
```

We can add column names and row names to this matrix:

```
colnames(B) <- c("c1","c2")
rownames(B) <- c("r1","r2","r3")
B
```

```
##    c1 c2
## r1  1  3
## r2  0  2
## r3  4  6
```

If, at this point, we would like to only work with, say, the first column of the matrix, we can call it using its column number, (1), or the column name, ("c1"), as follows:

```
B[,"c1"]
```

```
## r1 r2 r3
##  1  0  4
```

Similarly, if we want to refer to a matrix element, say $b_{3,2}$, we can do this as follows:

```
B[3,2]
```

```
## [1] 6
```

Matrix multiplication is done using %*% command, granted that the two matrices are compatible. For example, we obtain a product of matrix B and a new 2×1 vector, d, as follows:

```
d <- as.matrix(c(5,-2))
Bd <- B%*%d
Bd
```

```
##    [,1]
## r1   -1
## r2   -4
## r3    8
```

We can add columns (and rows) to the existing matrix using a cbind() function:

```
c3 <- c(0,1,0)
D <- cbind(B,c3)
D
```

```
##    c1 c2 c3
## r1  1  3  0
## r2  0  2  1
## r3  4  6  0
```

We can invert a(n invertible) matrix using the solve() function:

```
Di <- solve(D)
Di
```

```
##            r1 r2         r3
## c1 -1.0000000  0  0.5000000
## c2  0.6666667  0 -0.1666667
## c3 -1.3333333  1  0.3333333
```

Data Visualisation

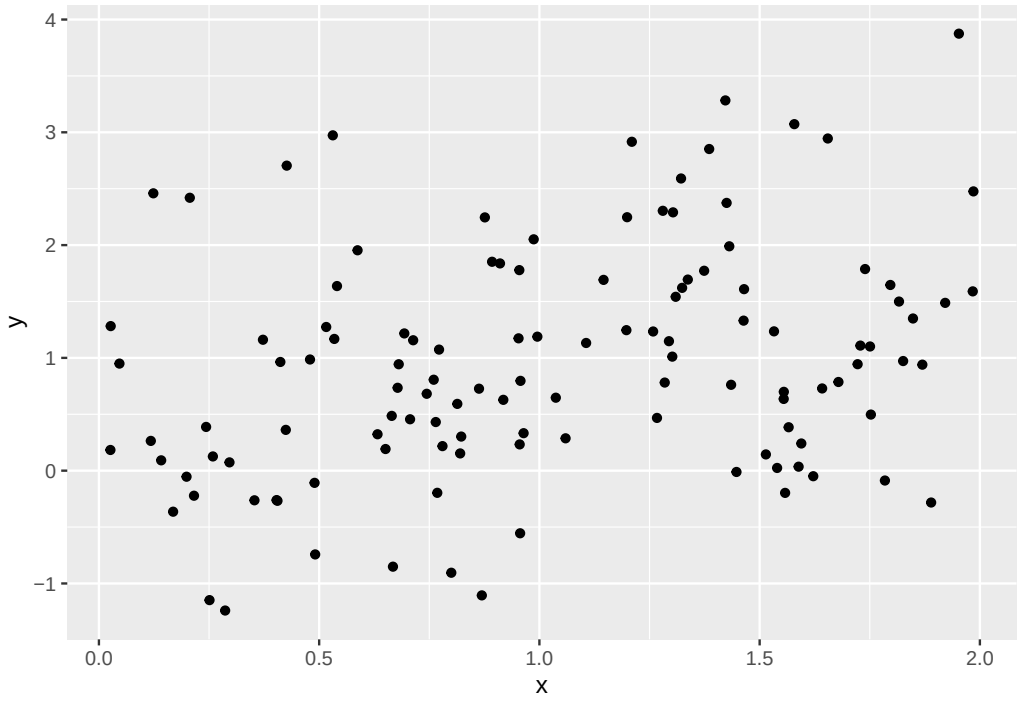
One of the comparative advantages of R is in its graphing aesthetics. Currently, the best graphs are plotted via the **ggplot2** package. Notably, this package requires that the data are maintained in the data.frame or the data.table format (for the latter, you need to load the **data.table** package). Let's create a data.table object and observe its few lines:

```
set.seed(1)
x <- runif(120,0,2)
y <- 0.2+0.7*x+rnorm(120)

library(data.table)

dt <- data.table(y=y,x=x)
dt
```

```
##                y         x
##   1:  2.9733299 0.53101733
##   2:  0.6817335 0.74424780
##   3:  1.6917341 1.14570673
##   4:  1.4994931 1.81641558
##   5: -0.2609185 0.40336386
##  ---
## 116:  0.1835826 0.02615515
## 117:  1.9894321 1.43113213
## 118:  2.4197029 0.20636847
## 119:  1.8521905 0.89256870
## 120:  2.3040499 1.28020209
```

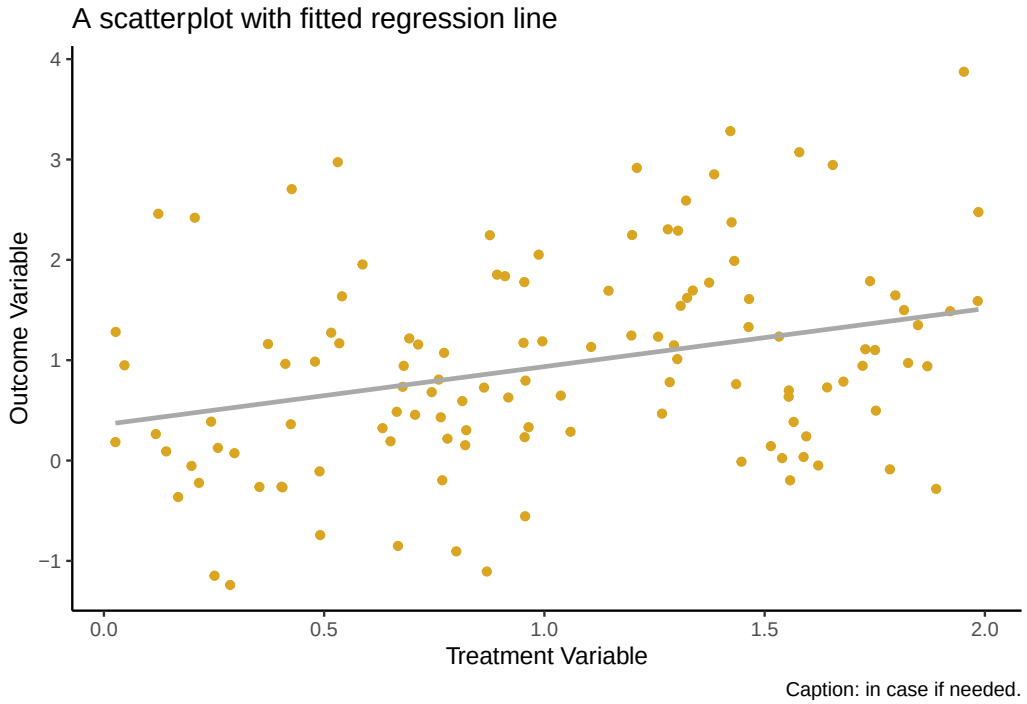
Now, let's load **ggplot2** and generate a simple scatter plot:

```
library(ggplot2)

ggplot(dt,aes(x=x,y=y))+
  geom_point()
```



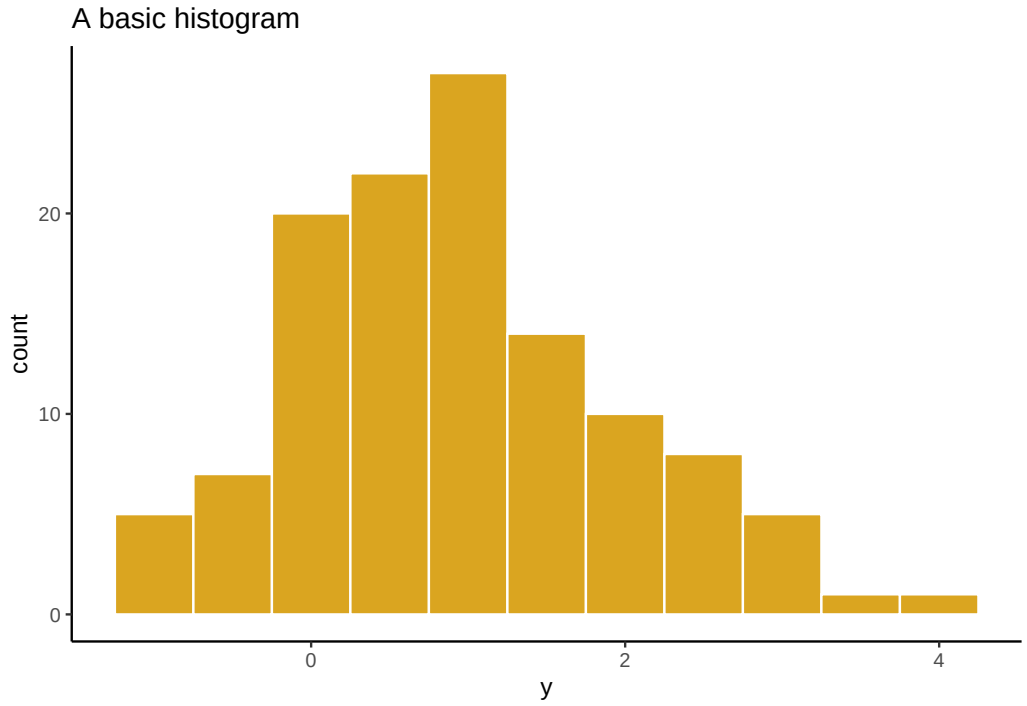
We can augment this plot in a number of different ways. Here we change the point color to red, add the fitted regression line to the plot, add labels to the figure, and apply a 'classic' background theme:

```
ggplot(dt,aes(x=x,y=y))+
  geom_point(color="goldenrod")+
  geom_smooth(method="lm",formula=y~x,se=F,color="darkgray")+
  labs(title="A scatterplot with fitted regression line",
       x="Treatment Variable",
       y="Outcome Variable",
       caption="Caption: in case if needed.")+
  theme_classic()
```



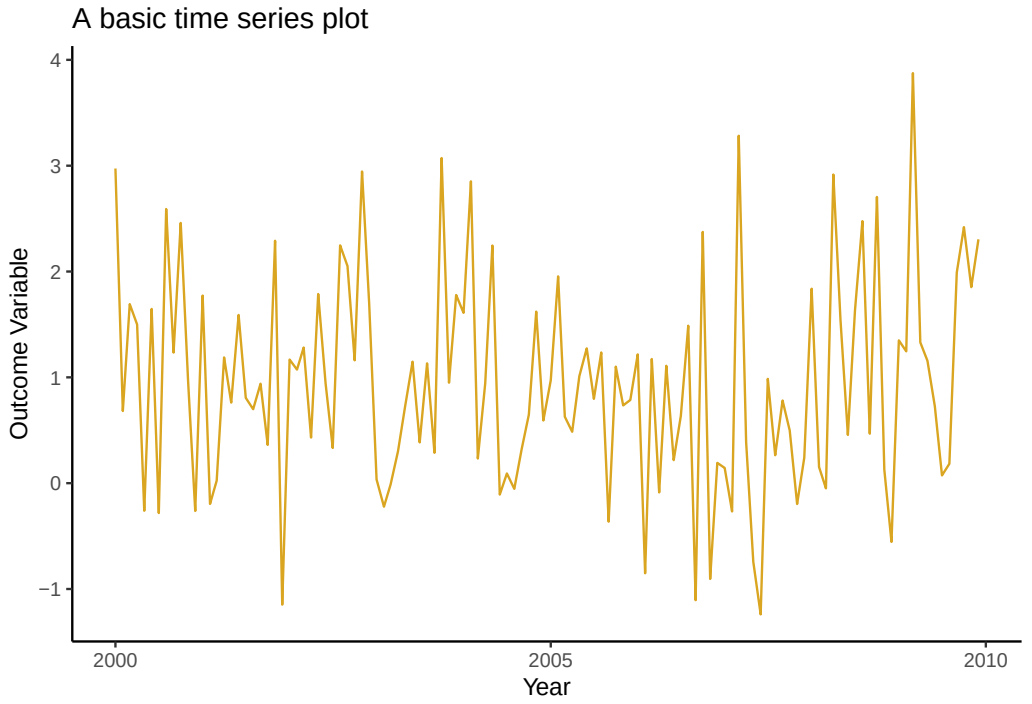
As another example, let's generate a histogram (of the dependent variable):

```
ggplot(dt,aes(x=y))+
  geom_histogram(color="white",fill="goldenrod",binwidth=.5)+
  labs(title="A basic histogram")+
  theme_classic()
```



We typically apply a line plot to illustrate a time series (that are ordered by date). In what follows, we add a date column to our data frame and then plot the dependent variable in the chronological order:

```
dt$date <- seq(from=as.Date("2000-01-01"),by="month",along.with=y)

ggplot(dt,aes(x=date,y=y))+
  geom_line(color="goldenrod")+
  labs(title="A basic time series plot",
       x="Year",
       y="Outcome Variable")+
  theme_classic()
```



Regression Analysis

To illustrate the OLS regression in R, we apply the previously generated x and y as independent and dependent variables. To begin, we obtain the least squares estimator "by hand" as follows:

```
X <- cbind(1,x)
b <- solve(t(X)%*%X)%*%t(X)%*%y
b
```

```
##        [,1]
##   0.3577680
## x 0.5781188
```

This can be easily done using the lm() function:

```
ols <- lm(y~x)
ols
```

```
##
```

```
## Call:
## lm(formula = y ~ x)
##
## Coefficients:
## (Intercept)            x
##      0.3578       0.5781
```

We can apply the summary() function to see the complete set of regression results:

```
summary(ols)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.9662 -0.5983 -0.1127  0.5639  2.3882
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   0.3578     0.1904   1.879 0.062717 .
## x             0.5781     0.1641   3.522 0.000609 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9716 on 118 degrees of freedom
## Multiple R-squared:  0.09514,    Adjusted R-squared:  0.08748
## F-statistic: 12.41 on 1 and 118 DF,  p-value: 0.0006091
```

Tutorial 2: Some R Functions

In this tutorial, we will introduce several simple R functions, and will perform a basic forecasting exercise.

Let's generate a sequence of 200 iid random variables with mean zero and variance 4, call it e.

```
set.seed(1)
e <- rnorm(200,0,2)
```

Notice that prior to sampling we set seed to some value (to one in this instance). We do so to ensure that we can exactly replicate the sample in the future.

Nett, generate a sequence of 200 binary variables, call it x.

```
set.seed(2)
x <- sample(c(0,1),200,replace=T)
```

Construct a dependent variable, y, using the following formula: $y = 2+0.5x+e$.

```
y <- 2+0.5*x+e
```

Regress y on x, using the `lm()` function, to obtain estimates of the intercept and slope parameters.

```
ols <- lm(y~x)
ols
```

```
##
## Call:
## lm(formula = y ~ x)
##
```

```
## Coefficients:
## (Intercept)            x
##      2.1244       0.3854
```

Generate some "future" realizations (100 observations) of y.

```
set.seed(3)
e <- rnorm(100,0,2)

set.seed(4)
x <- sample(c(0,1),100,replace=T)

y <- 2+0.5*x+e
```

Note that these represent actual realizations of the variable; these not forecasts.

Suppose we think that in the considered forecast period, x only takes on 1 (below we will refer to this as the Model 1). Based on this, and using parameter estimates from above, let's generate forecasts for this period.

```
y_f1 <- ols$coefficients[1]+ols$coefficients[2]*rep(1,100)
```

At this point, we have actual realisations of y and its forecasts. Thus, we can obtain forecast errors, mean absolute forecast errors, and root mean square forecast errors.

```
e_f1 <- y-y_f1

mafe1 <- mean(abs(e_f1))
rmsfe1 <- sqrt(mean(e_f1^2))

mafe1
```

```
## [1] 1.43523
```

```
rmsfe1
```

```
## [1] 1.739508
```

Suppose, instead, we think that in the considered forecast period x only takes on 0 (below we will refer to this as the Model 2). Based on this, and using

parameter estimates from above, let's generate forecasts for this period.

```
y_f0 <- ols$coefficients[1]+ols$coefficients[2]*rep(0,100)
```

Using these forecasts, obtain forecast errors, mean absolute forecast errors, and root mean square forecast errors.

```
e_f0 <- y-y_f0

mafe0 <- mean(abs(e_f0))
rmsfe0 <- sqrt(mean(e_f0^2))

mafe0
```

```
## [1] 1.455768
```

```
rmsfe0
```

```
## [1] 1.736182
```

By comparing the two sets of forecasts, we can observe a somewhat rare and yet not an unlikely scenario: MAFE points to the Model 1 as more accurate of the two models, while RMSFE suggests the Model 2 as the more accurate one. More often than not, however, these two accuracy measures tend to agree.

Tutorial 3: Forecasting Methods and Routines

In this tutorial, we will introduce 'for loop,' and use it to generate time series as well as to obtain one-step-ahead forecasts using a rolling window procedure; we will also perform forecast error diagnostics.

Let's generate a random walk process, such that $y_t = y_{t-1} + e_t$, where e_t $N(0, 1)$, and where $y_0 = 0$, for $t = 1, \ldots, 120$.

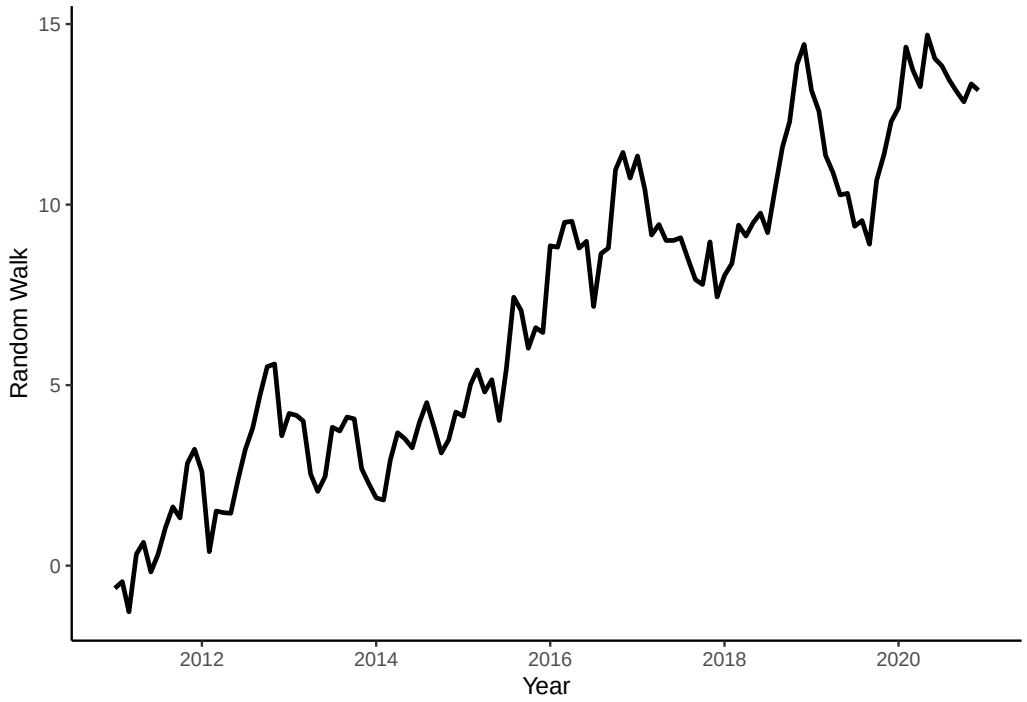
```
n <- 120

set.seed(1)
e <- rnorm(n)

y <- rep(NA,n)

y[1] <- e[1]

for(i in 2:n){
  y[i] <- y[i-1] + e[i]
}
```

Store y and e in a **data.table**, call it 'dt.' Add some arbitrary dates to the data (e.g., suppose we deal with the monthly series beginning from January 2011).

```
dt <- data.table(y,e)

dt$date <- seq(as.Date("2011-01-01"),as.Date("2020-12-01"),by="month")
```

Plot the realized time series using **ggplot** function.

```
ggplot(dt,aes(x=date,y=y))+
  geom_line(size=1)+
  labs(x="Year",y="Random Walk")+
  theme_classic()
```



Generate a sequence of one-step-ahead forecasts from naive and average methods, using the rolling window scheme, where the first rolling window ranges from period 1 to period 80.

```
dt$average <- NA
dt$naive <- NA

R <- 80
P <- n-R
for(i in 1:P){
  w <- y[i:(R-1+i)]
  dt$average[R+i] <- mean(w)
  dt$naive[R+i] <- w[length(w)]
```

```
}
```

Calculate the RMSFE measures for each of the two forecasting methods.

```
dt[,`:=`(e_average=y-average,e_naive=y-naive)]

rmsfe_average <- sqrt(mean(dt$e_average^2,na.rm=T))
rmsfe_naive <- sqrt(mean(dt$e_naive^2,na.rm=T))

rmsfe_average
```

```
## [1] 4.672947
```

```
rmsfe_naive
```

```
## [1] 0.850081
```

Perform the forecast error diagnostics for the two considered methods.

Zero mean of the forecast errors: $E(e_{t+1|t}) = 0$. We perform this test by regressing the forecast error on the constant, and checking whether the coefficient is statistically significantly different from zero.

```
summary(lm(e_average~1,data=dt))$coefficients
```

```
##             Estimate Std. Error  t value     Pr(>|t|)
## (Intercept) 4.434858  0.2358002 18.80769 3.682273e-21
```

```
summary(lm(e_naive~1,data=dt))$coefficients
```

```
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.1168396    0.13483 0.86657 0.391478
```

No correlation of the forecast errors with the forecasts: $Cov(e_{t+1|t}, y_{t+1|t}) = 0$. We perform this test by regressing the forecast error on the forecast, and checking whether the slope coefficient is statistically significantly different from zero.

```
summary(lm(e_average~average,data=dt))$coefficients
```

```
##              Estimate Std. Error  t value   Pr(>|t|)
## (Intercept) 2.4180785  1.2929708 1.870172 0.06917788
## average     0.2942557  0.1856048 1.585389 0.12116580
```

```
summary(lm(e_naive~naive,data=dt))$coefficients
```

```
##                Estimate Std. Error   t value  Pr(>|t|)
## (Intercept)  1.06489905 0.69740557  1.526944 0.1350565
## naive       -0.08486143 0.06127484 -1.384931 0.1741512
```

No serial correlation in one-step-ahead forecast errors: $Cov(e_{t+1|t}, y_{t|t-1}) = 0$. We perform this test by regressing the forecast error on its lag, and checking whether the slope coefficient is statistically significantly different from zero. (Note: first we need to generate lagged forecast errors)

```
dt[,`:=`(e_average.l1=shift(e_average),e_naive.l1=shift(e_naive))]

summary(lm(e_average~e_average.l1,data=dt))$coefficients
```

```
##                Estimate Std. Error  t value     Pr(>|t|)
## (Intercept)  0.7898068 0.42027705 1.879253 6.810403e-02
## e_average.l1 0.8275396 0.08966026 9.229726 3.892504e-11
```

```
summary(lm(e_naive~e_naive.l1,data=dt))$coefficients
```

```
##                Estimate Std. Error   t value  Pr(>|t|)
## (Intercept) 0.12971406  0.1403668 0.9241081 0.3614178
## e_naive.l1  0.03780853  0.1631333 0.2317647 0.8179979
```

Tutorial 4: Deterministic Trends

In this tutorial, we will generate trending series, we will apply an information criterion to select the most suitable trend model, and we will obtain and compare one-step-ahead forecasts using a rolling window procedure.

Let's generate a time series that follows a quadratic trend: $y_t = 10 + 0.01t + 0.002t^2 + e_t$, where $e_t \sim N(0, 16)$, for $t = 1, \ldots, 180$.

```
n <- 180

set.seed(7)
e <- rnorm(n,0,4)

trend <- c(1:n)

y <- 10+0.01*trend+0.002*trend^2+e
```

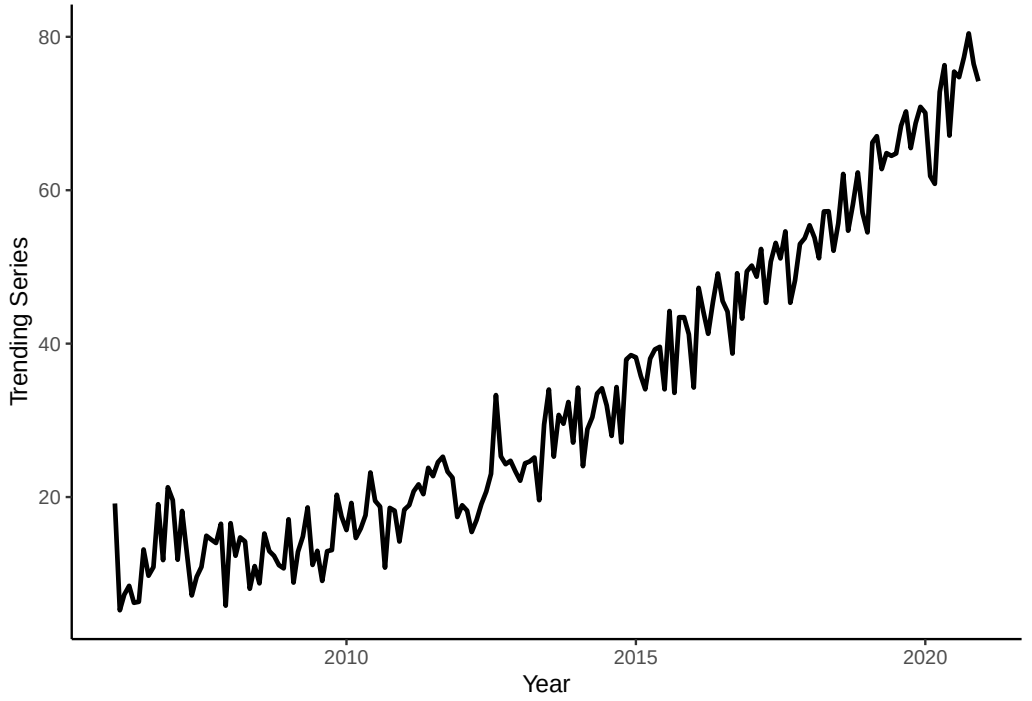
Store y and *trend* in a **data.table**, call it 'dt.' Add some arbitrary dates to the data (e.g., suppose we deal with the monthly series beginning from January 2006).

```
dt <- data.table(y,trend)

dt$date <- seq(as.Date("2006-01-01"),by="month",along.with=y)
```

Plot the realized time series using **ggplot** function.

```
ggplot(dt,aes(x=date,y=y))+
  geom_line(size=1)+
```

```
  labs(x="Year",y="Trending Series")+
  theme_classic()
```



Calculate Akaike Information Criteria for linear, quadratic, cubic, and exponential trend models, using all observations in the series.

```
AIC_vec <- matrix(ncol=4,nrow=1)
for(i in 1:4){
  if(i < 4){
    reg <- lm(y~poly(trend,degree=i,raw=T),data=dt)
    AIC_vec[i] <- log(crossprod(reg$residuals))+2*length(reg$coefficients)/n
  }else{
    reg <- lm(log(y)~trend,data=dt)
    yhat <- reg$fitted.values
    sig <- sd(reg$residuals)
    ystar <- exp(yhat+sig^2/2)
    res <- dt$y-ystar
    AIC_vec[i] <- log(crossprod(res))+2*length(reg$coefficients)/n
  }
```

```
}

AIC_vec
```

```
##          [,1]     [,2]     [,3]     [,4]
## [1,] 8.879646 7.841645 7.849521 7.975999
```

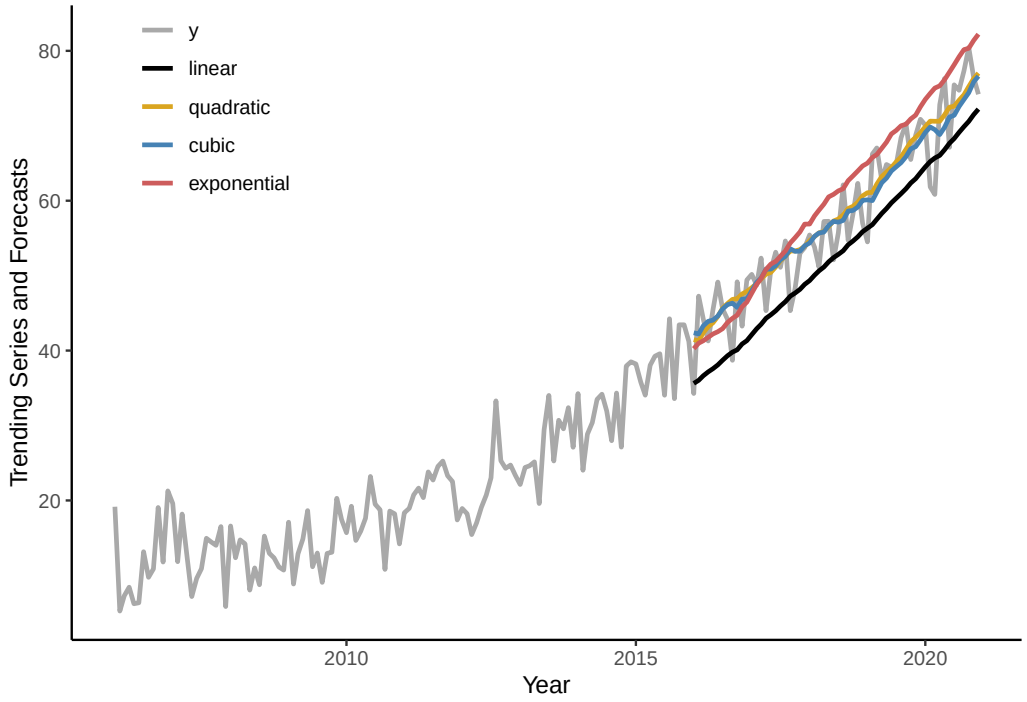
Generate a sequence of one-step-ahead forecasts from linear, quadratic, cubic, and exponential trend models, using the rolling window scheme, where the first rolling window ranges from period 1 to period 120.

```
dt$t1 <- NA
dt$t2 <- NA
dt$t3 <- NA
dt$te <- NA

R <- 120
P <- n-R
for(i in 1:P){
  reg1 <- lm(y~trend,data=dt[i:(R-1+i)])
  dt$t1[R+i] <- reg1$coef[1]+reg1$coef[2]*(R+i)

  reg2 <- lm(y~poly(trend,degree=2,raw=T),data=dt[i:(R-1+i)])
  dt$t2[R+i] <- reg2$coef[1]+reg2$coef[2]*(R+i)+reg2$coef[3]*((R+i)^2)

  reg3 <- lm(y~poly(trend,degree=3,raw=T),data=dt[i:(R-1+i)])
  dt$t3[R+i] <- reg3$coef[1]+reg3$coef[2]*(R+i)+reg3$coef[3]*((R+i)^2)+reg3$coef[4

  rege <- lm(log(y)~trend,data=dt[i:(R-1+i)])
  sig <- sd(rege$residuals)
  dt$te[R+i] <- exp(rege$coef[1]+rege$coef[2]*(R+i)+sig^2/2)
}
```

Plot the original series overlay by the one-step-ahead forecasts from the four considered trend models. Note, for convenience we will first 'melt' the data.table in to the 'long' format.

```
dt_long <- melt(dt[,.(date,y,linear=t1,quadratic=t2,cubic=t3,exponential=te)],id.v
```

```
ggplot(dt_long,aes(x=date,y=value,color=variable))+
  geom_line(size=1,na.rm=T)+
  scale_color_manual(values=c("darkgray","black","goldenrod","steelblue","indianre
  labs(x="Year",y="Trending Series and Forecasts")+
  theme_classic()+
  theme(legend.title=element_blank(),legend.position=c(.15,.85))
```



Calculate the RMSFE measures for each of the two forecasting methods.

```
dt[,`:=`(e_t1=y-t1,e_t2=y-t2,e_t3=y-t3,e_te=y-te)]

rmsfe_t1 <- sqrt(mean(dt$e_t1^2,na.rm=T))
rmsfe_t2 <- sqrt(mean(dt$e_t2^2,na.rm=T))
rmsfe_t3 <- sqrt(mean(dt$e_t3^2,na.rm=T))
rmsfe_te <- sqrt(mean(dt$e_te^2,na.rm=T))

rmsfe_t1
```

```
## [1] 6.151904
```

```
rmsfe_t2
```

```
## [1] 3.796861
```

```
rmsfe_t3
```

```
## [1] 3.906023
```

```
rmsfe_te
```

```
## [1] 5.132312
```

Tutorial 5: Seasonality

(this is a AR stuff, will need to move back)

In this tutorial, we will generate autocorrelated series, we will apply an information criterion to select a suitable autoregressive model, we will obtain and compare one-step-ahead forecasts from competing models using a rolling window procedure, and we will generate one set of multi-step forecasts to illustrate the convergence to unconditional mean of the series.

Let's generate a time series that follows an AR(2) process: $y_t = 1.2y_{t-1} - 0.3y_{t-2} + e_t$, where $e_t \sim N(0,1)$, for $t = 1, \ldots, 180$.

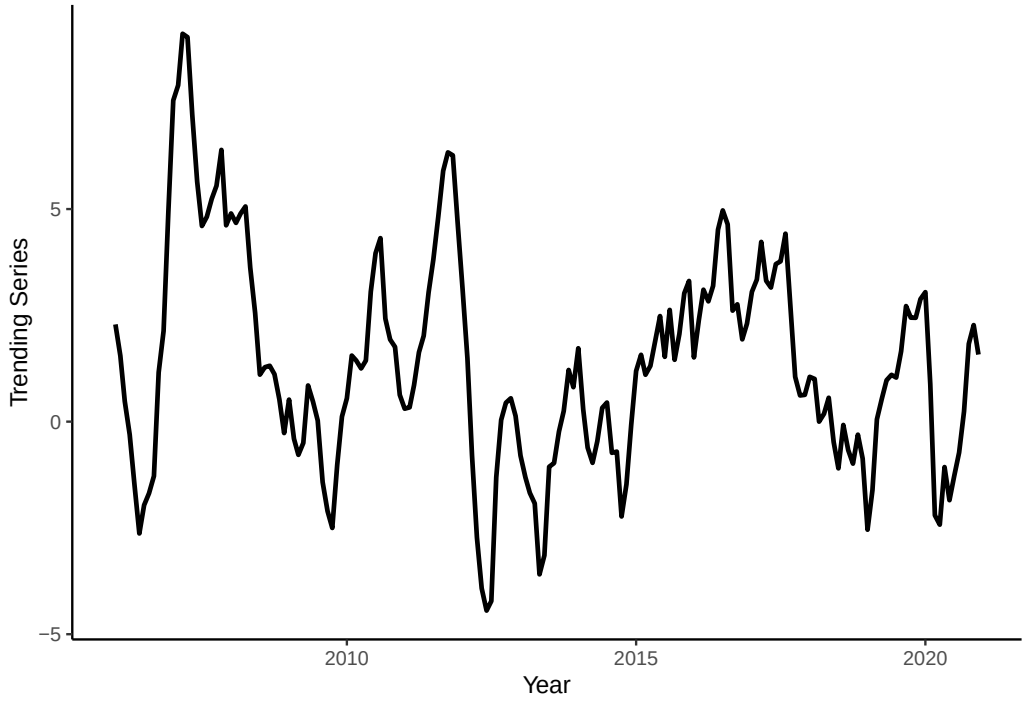
```
n <- 180

set.seed(7)
e <- rnorm(n,0,1)

y <- rep(NA,n)
y[1] <- e[1]
y[2] <- 1.2*y[1]+e[2]
for(i in 3:n){
  y[i] <- 1.2*y[i-1]-0.3*y[i-2]+e[i]
}
```

Generate a vector of some arbitrary dates (e.g., suppose we deal with the monthly series beginning from January 2006), and store these along with y in a **data.table**, call it 'dt.'

```
date <- seq(as.Date("2006-01-01"),by="month",along.with=y)

dt <- data.table(date,y)
```

Plot the realized time series using **ggplot** function.

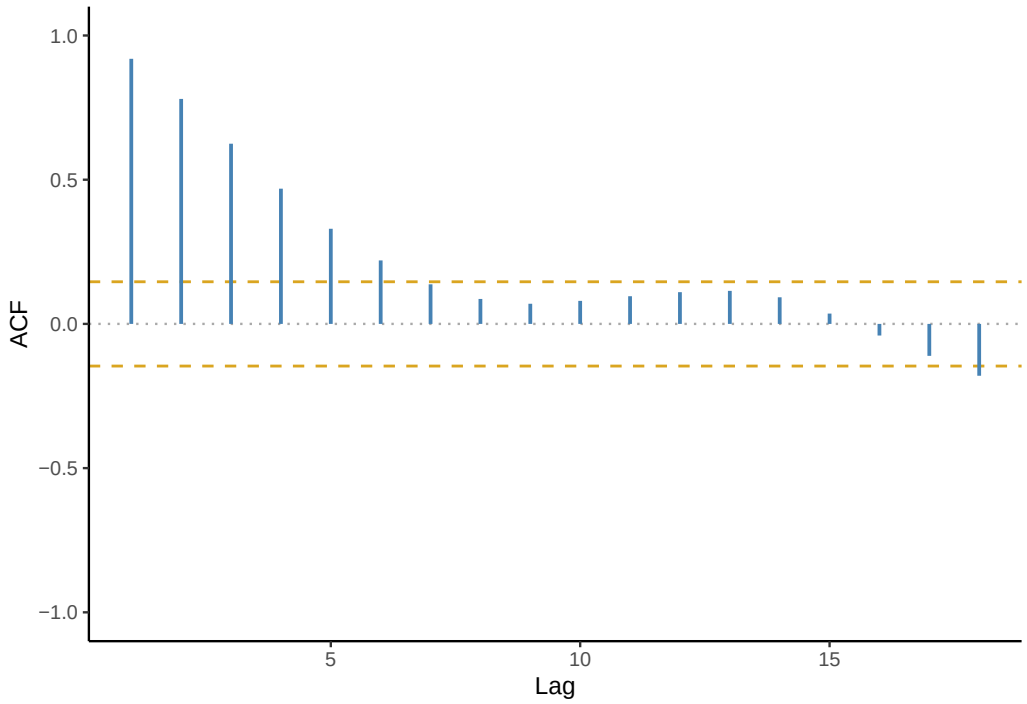
```
ggplot(dt,aes(x=date,y=y))+
  geom_line(size=1)+
  labs(x="Year",y="Trending Series")+
  theme_classic()
```



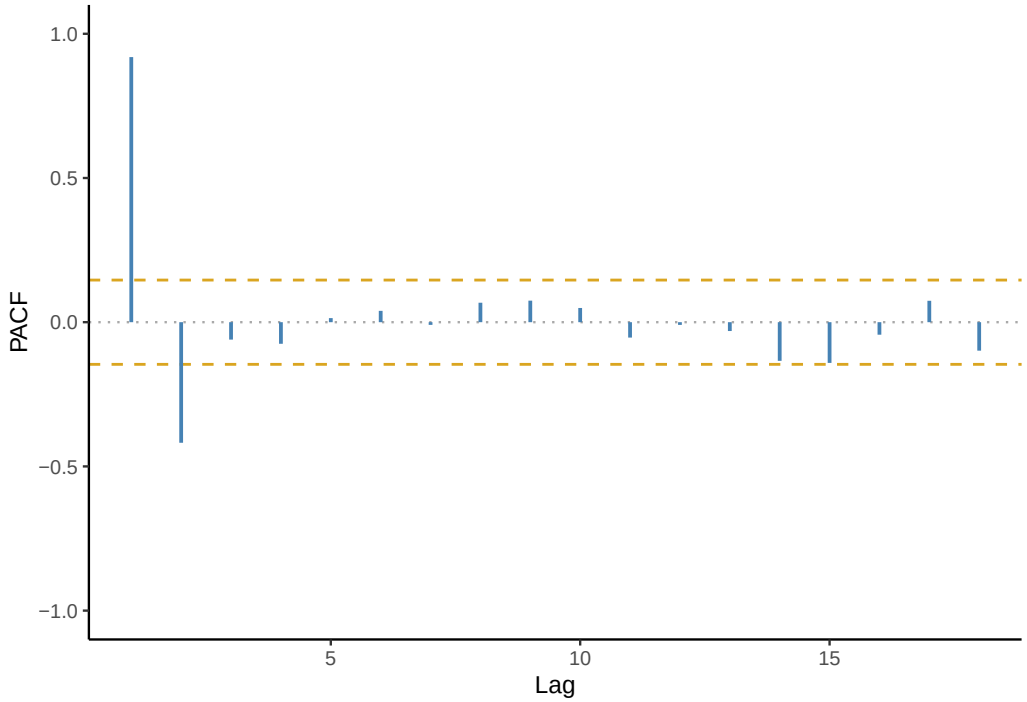
Generate and plot the autocorrelation function and the partial autocorrelation function for lags 1 through 18.

```
acf_vec <- c(acf(dt$y,lag.max=18,plot=F)$acf[-1])
pacf_vec <- c(pacf(dt$y,lag.max=18,plot=F)$acf)

sd_rho <- sqrt(1/n)

acf_dt <- data.table(lags=1:18,acf=acf_vec,pacf=pacf_vec)

ggplot(acf_dt,aes(x=lags,y=acf)) +
```

```
  geom_hline(yintercept=0,color="darkgray",linetype=3,size=.5) +
  geom_hline(yintercept=c(-1.96*sd_rho,1.96*sd_rho),color="goldenrod",linetype=2,s
  geom_segment(aes(xend=lags,yend=0),color="steelblue",size=.8)+
  labs(x="Lag",y="ACF")+
  coord_cartesian(ylim=c(-1,1))+
  theme_classic()
```




```
ggplot(acf_dt,aes(x=lags,y=pacf)) +
  geom_hline(yintercept=0,color="darkgray",linetype=3,size=.5) +
  geom_hline(yintercept=c(-1.96*sd_rho,1.96*sd_rho),color="goldenrod",linetype=2,s
  geom_segment(aes(xend=lags,yend=0),color="steelblue",size=.8)+
  labs(x="Lag",y="PACF")+
  coord_cartesian(ylim=c(-1,1))+
  theme_classic()
```



Calculate Akaike Information Criteria (AIC) and Schwarz Information Criteria (SIC) for AR(1), AR(2), AR(3), and AR(4) models, using all observations in the series, to decide on the optimal lag length.

```
dt[,`:=`(y_l1=shift(y),y_l2=shift(y,2),y_l3=shift(y,3),y_l4=shift(y,4))]

# get rid of the rows with NAs
dt <- dt[complete.cases(dt)]

IC_dt <- data.table(lag=c(1:4),AIC=NA,SIC=NA)

for(i in 1:nrow(IC_dt)){

  fmla <- as.formula(paste("y",paste0("y_l",c(1:i),collapse="+"),sep="~"))
  reg.ar <- lm(fmla,data=dt)

  IC_dt$AIC[i] <- log(crossprod(reg.ar$residuals))+2*(i+1)/nrow(dt)
  IC_dt$SIC[i] <- log(crossprod(reg.ar$residuals))+log(nrow(dt))*(i+1)/nrow(dt)
```

```
}

IC_dt
```

```
##    lag      AIC      SIC
## 1:   1 5.192156 5.228184
## 2:   2 5.011351 5.065393
## 3:   3 5.018687 5.090743
## 4:   4 5.024127 5.114198
```

Generate a sequence of one-step-ahead forecasts from random walk, as well as AR(1), AR(2), and AR(3), using the rolling window scheme, where the first rolling window ranges from period 1 to period 120.

```
R <- 120
P <- nrow(dt)-R

dt$rw <- NA
dt$ar1 <- NA
dt$ar2 <- NA

for(i in 1:P){
  dt$rw[R+i] <- dt$y[R-1+i]

  ar1 <- lm(y~y_l1,data=dt[i:(R-1+i)])
  ar2 <- lm(y~y_l1+y_l2,data=dt[i:(R-1+i)])

  dt$ar1[R+i] <- ar1$coefficients[1]+ar1$coefficients[2]*dt$y[R-1+i]
  dt$ar2[R+i] <- ar2$coefficients[1]+ar2$coefficients[2]*dt$y[R-1+i]+ar2$coefficie
}
```

Calculate the RMSFE measures for all considered models.

```
dt[,`:=`(rw_e=y-rw,ar1_e=y-ar1,ar2_e=y-ar2)]

rmsfe_rw <- sqrt(mean(dt$rw_e^2,na.rm=T))
rmsfe_ar1 <- sqrt(mean(dt$ar1_e^2,na.rm=T))
rmsfe_ar2 <- sqrt(mean(dt$ar2_e^2,na.rm=T))
```

```
rmsfe_rw
```

```
## [1] 0.9569683
```

```
rmsfe_ar1
```

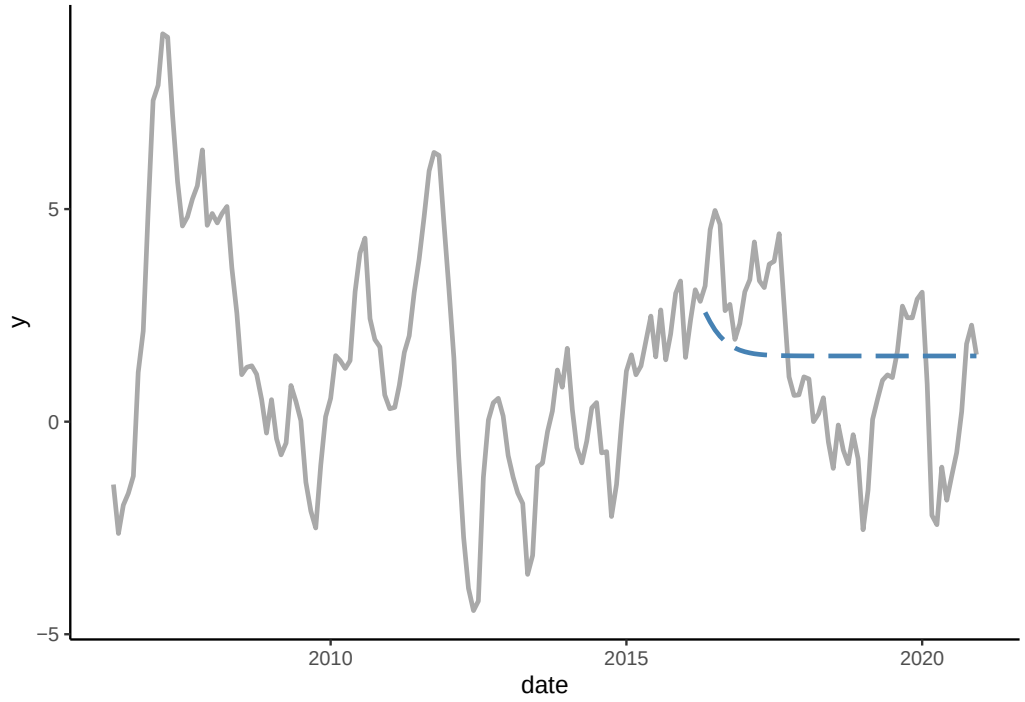
```
## [1] 0.9298374
```

```
rmsfe_ar2
```

```
## [1] 0.8901728
```

Using the first rolling window as the information set, generate the multi-step-ahead forecast for the hold-out period.

```
dt[,`:=`(ar2_multi=y)]

ar2 <- lm(y~y_l1+y_l2,data=dt[1:R])

for(i in 1:P){

  dt$ar2_multi[R+i] <- ar2$coefficients[1]+ar2$coefficients[2]*dt$ar2_multi[R-1+i]

}

dt[1:R]$ar2_multi <- NA

ggplot(dt,aes(x=date))+
  geom_line(aes(y=y),color="darkgray",size=1)+
  geom_line(aes(y=ar2_multi),color="steelblue",na.rm=T,size=1,linetype=5)+
  theme_classic()
```


5
0
−5
y
2010
2015
2020
date

Tutorial 6: Linear Autoregression

(this is a VAR stuff, will need to move back)

In this tutorial, we will generate bivariate series, we will apply a system-wide information criterion to select a suitable vector autoregressive model, we will perform an in-sample test of Granger causality, we will obtain and compare one-step-ahead forecasts from competing models using a rolling window procedure, and in so doing we will investigate the evidence of Granger causality in an out-of-sample setting. To run the code, the `data.table` and `MASS` packages need to be installed and loaded.

Let's generate a two-dimensional vector of time series that follow a VAR(1) process of the following form:

$$x_{1,t} = 0.3 + 0.7x_{1,t-1} + 0.1x_{2,t-1} + \varepsilon_{1,t}$$
$$x_{2,t} = -0.2 + 0.9x_{1,t-1} + \varepsilon_{2,t}$$

where $\mathbf{e}_t \sim N(\mathbf{0}, \Sigma)$, and where Σ is the covariance matrix of the residuals such that $Cov(\varepsilon_{1,t}, \varepsilon_{2,t}) = 0.3$ for all $t = 1, \ldots, 180$. (Note: in the code, x_1 is denoted by y and x_2 is denoted by x).

```
n <- 180

R <- matrix(c(1,0.3,0.3,1),nrow=2,ncol=2)
set.seed(1)
e <- mvrnorm(n,mu=c(0,0),Sigma=R)

e_y <- e[,1]
```

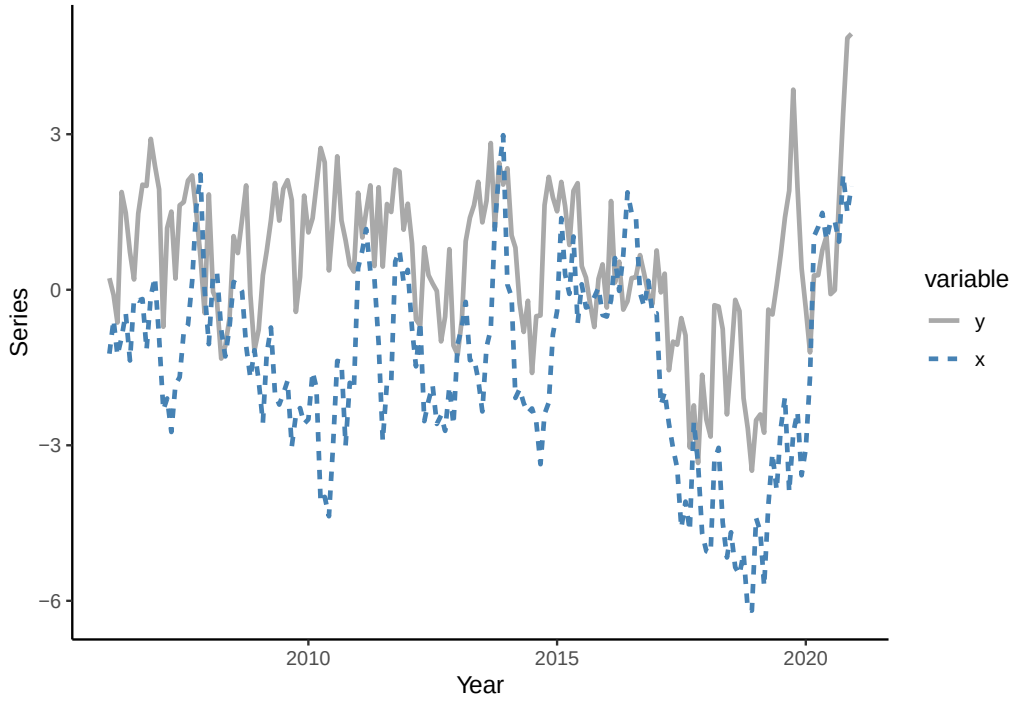
```
e_x <- e[,2]

y <- rep(NA,n)
x <- rep(NA,n)

y[1] <- e_y[1]
x[1] <- e_x[1]

for(i in 2:n){
  y[i] <- 0.3+0.7*y[i-1]+0.1*x[i-1]+e_y[i]
  x[i] <- -0.2+0.9*x[i-1]+e_x[i]
}
```

Generate a vector of some arbitrary dates (e.g., suppose we deal with the monthly series beginning from January 2006), and store these along with y in a **data.table**, call it 'dt.'

```
date <- seq(as.Date("2006-01-01"),by="month",along.with=y)

dt <- data.table(date,y,x)
```

Plot the realized time series using **ggplot** function.

```
dt_long <- melt(dt,id.vars="date")

ggplot(dt_long,aes(x=date,y=value,color=variable,linetype=variable))+
  geom_line(size=1)+
  scale_color_manual(values=c("darkgray","steelblue"))+
  labs(x="Year",y="Series")+
  theme_classic()
```



Estimate VAR(1) and VAR(2) by running regressions on each equation separately. Collect residuals and obtain system-wide AIC for each of the two models.

```
dt[,`:=`(y_l1=shift(y,1),y_l2=shift(y,2),x_l1=shift(x,1),x_l2=shift(x,2))]

# VAR(1)
p <- 1
k <- 2

var1y <- lm(y~y_l1+x_l1,data=dt)
var1x <- lm(x~y_l1+x_l1,data=dt)

var1r <- cbind(var1y$residuals,var1x$residuals)
cov1r <- crossprod(var1r)/(nrow(dt)-(p*k^2+k))

AIC1 <- log(det(cov1r))+2*(p*k^2+k)/nrow(dt)

# VAR(2)
```

```
p <- 2
k <- 2

var2y <- lm(y~y_l1+y_l2+x_l1+x_l2,data=dt)
var2x <- lm(x~y_l1+y_l2+x_l1+x_l2,data=dt)

var2r <- cbind(var2y$residuals,var2x$residuals)
cov2r <- crossprod(var2r)/(nrow(dt)-(p*k^2+k))

AIC2 <- log(det(cov2r))+2*(p*k^2+k)/nrow(dt)

AIC1
```

```
## [1] -0.1270596
```

```
AIC2
```

```
## [1] -0.047212
```

Perfrom tests of (in-sample) Granger causality in each of the two models. Note, in the case of VAR(1), both t tests and F tests are applicable and they both provide identical inference. In the case of VAR(p), where $p > 1$, the only appropriate test is an F test for joint significance of the parameters associated with the lags of the potentially causal variable.

```
# VAR(1)

## t test
summary(var1y)
```

```
##
## Call:
## lm(formula = y ~ y_l1 + x_l1, data = dt)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.32279 -0.63680 -0.00953  0.68826  2.63468
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)  0.34926    0.11146   3.134  0.00202 **
## y_l1         0.68903    0.05877  11.725  < 2e-16 ***
## x_l1         0.11304    0.04509   2.507  0.01308 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.976 on 176 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.5689, Adjusted R-squared:  0.564
## F-statistic: 116.1 on 2 and 176 DF,  p-value: < 2.2e-16
```

```
summary(var1x)
```

```
##
## Call:
## lm(formula = x ~ y_l1 + x_l1, data = dt)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.38491 -0.68087  0.03216  0.66109  2.74762
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.22353    0.10805  -2.069    0.040 *
## y_l1         0.05814    0.05697   1.021    0.309
## x_l1         0.85285    0.04371  19.511   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9461 on 176 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.7536, Adjusted R-squared:  0.7508
## F-statistic: 269.2 on 2 and 176 DF,  p-value: < 2.2e-16
```

```
## F test
ar1y <- lm(y~y_l1,data=dt)
ar1x <- lm(x~x_l1,data=dt)

anova(var1y,ar1y)
```

```
## Analysis of Variance Table
##
## Model 1: y ~ y_l1 + x_l1
## Model 2: y ~ y_l1
##   Res.Df    RSS Df Sum of Sq      F  Pr(>F)
## 1    176 167.64
## 2    177 173.62 -1   -5.9866 6.2852 0.01308 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(var1x,ar1x)
```

```
## Analysis of Variance Table
##
## Model 1: x ~ y_l1 + x_l1
## Model 2: x ~ x_l1
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    176 157.54
## 2    177 158.47 -1  -0.93231 1.0416 0.3089
```

```
## VAR(2)

### t test (no longer applicable to test GC)
summary(var2y)
```

```
##
## Call:
## lm(formula = y ~ y_l1 + y_l2 + x_l1 + x_l2, data = dt)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.38088 -0.71387 -0.01504  0.72538  2.70511
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.32203    0.11775   2.735  0.00689 **
## y_l1         0.65954    0.07822   8.431  1.3e-14 ***
## y_l2         0.04964    0.07965   0.623  0.53397
## x_l1         0.15919    0.08052   1.977  0.04962 *
```

```
## x_l2        -0.05991    0.08104  -0.739  0.46080
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9816 on 173 degrees of freedom
##   (2 observations deleted due to missingness)
## Multiple R-squared:  0.5709, Adjusted R-squared:  0.561
## F-statistic: 57.55 on 4 and 173 DF,  p-value: < 2.2e-16
```

```
summary(var2x)
```

```
##
## Call:
## lm(formula = x ~ y_l1 + y_l2 + x_l1 + x_l2, data = dt)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.37790 -0.64364  0.05401  0.67542  2.71827
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.24577    0.11419  -2.152   0.0328 *
## y_l1         0.03070    0.07586   0.405   0.6862
## y_l2         0.04486    0.07724   0.581   0.5621
## x_l1         0.86196    0.07809  11.039   <2e-16 ***
## x_l2        -0.01627    0.07859  -0.207   0.8362
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9519 on 173 degrees of freedom
##   (2 observations deleted due to missingness)
## Multiple R-squared:  0.7546, Adjusted R-squared:  0.7489
## F-statistic:   133 on 4 and 173 DF,  p-value: < 2.2e-16
```

```
### F test
ar2y <- lm(y~y_l1+y_l2,data=dt)
ar2x <- lm(x~x_l1+x_l2,data=dt)

anova(var2y,ar2y)
```

```
## Analysis of Variance Table
##
## Model 1: y ~ y_l1 + y_l2 + x_l1 + x_l2
## Model 2: y ~ y_l1 + y_l2
##   Res.Df    RSS Df Sum of Sq      F  Pr(>F)
## 1    173 166.68
## 2    175 172.78 -2   -6.0971 3.1641 0.04471 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(var2x,ar2x)
```

```
## Analysis of Variance Table
##
## Model 1: x ~ y_l1 + y_l2 + x_l1 + x_l2
## Model 2: x ~ x_l1 + x_l2
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    173 156.76
## 2    175 158.03 -2   -1.2729 0.7024 0.4968
```

Generate a sequence of one-step-ahead forecasts from VAR(1) using the rolling window scheme, where the first rolling window ranges from period 1 to period 120.

```
R <- 120
P <- nrow(dt)-R

dt$ar1y <- NA
dt$ar1x <- NA
dt$var1y <- NA
dt$var1x <- NA

for(i in 1:P){

  ar1y <- lm(y~y_l1,data=dt[i:(R-1+i)])
  ar1x <- lm(x~x_l1,data=dt[i:(R-1+i)])

  var1y <- lm(y~y_l1+x_l1,data=dt[i:(R-1+i)])
  var1x <- lm(x~y_l1+x_l1,data=dt[i:(R-1+i)])
```

```
  dt$ar1y[R+i] <- ar1y$coefficients[1]+ar1y$coefficients[2]*dt$y[R-1+i]
  dt$ar1x[R+i] <- ar1x$coefficients[1]+ar1x$coefficients[2]*dt$x[R-1+i]

  dt$var1y[R+i] <- var1y$coefficients[1]+var1y$coefficients[2]*dt$y[R-1+i]+var1y$c
  dt$var1x[R+i] <- var1x$coefficients[1]+var1x$coefficients[2]*dt$y[R-1+i]+var1x$c

}
```

Calculate the RMSFE measures for restricted and unrestricted models, and compare those to each other to make a suggestion about out-of-sample Granger causality.

```
dt[,`:=`(ar1y_e=y-ar1y,ar1x_e=x-ar1x,var1y_e=y-var1y,var1x_e=x-var1x)]

# calculate RMSFE for restructed and unrestricted models
rmsfe_yr <- sqrt(mean(dt$ar1y_e^2,na.rm=T))
rmsfe_yu <- sqrt(mean(dt$var1y_e^2,na.rm=T))

rmsfe_xr <- sqrt(mean(dt$ar1x_e^2,na.rm=T))
rmsfe_xu <- sqrt(mean(dt$var1x_e^2,na.rm=T))

rmsfe_yr
```

```
## [1] 1.134867
```

```
rmsfe_yu
```

```
## [1] 1.104268
```

```
rmsfe_xr
```

```
## [1] 1.00908
```

```
rmsfe_xu
```

```
## [1] 1.011653
```

Tutorial 7: Vector Autoregression

(this is a threshold stuff, will need to move back)

In this tutorial, we will generate regime-dependent series, we will apply a grid-search method to obtain the threshold parameter, we will obtain and compare one-step-ahead forecasts from competing models using a rolling window procedure, and we will apply bootstrap resampling method to generate multi-step-ahead forecasts from a threshold regression. To run the code, the `data.table` and `ggplot2` packages need to be installed and loaded.

Let's generate a time series that follow a TAR(2) process of the following form:

$$y_t = \begin{cases} y_{t-1} + \varepsilon_t & \text{if } \ y_{t-1} \geq 0 \\ 1.2y_{t-1} - 0.3y_{t-2} + \varepsilon_t & \text{if } \ y_{t-1} < 0 \end{cases}$$

where $e_t \sim N(0, \sigma^2)$. This suggests that the time series follow the unit root process if the lagged dependent variable is non-negative, otherwise the time series is a mean-reverting AR(2) process.

```
n <- 360

set.seed(6)
e <- rnorm(n,0,1)

y <- rep(NA,n)
y[1] <- e[1]
if(y[1]>=0){
  y[2] <-  1.0*y[1]+e[2]
```

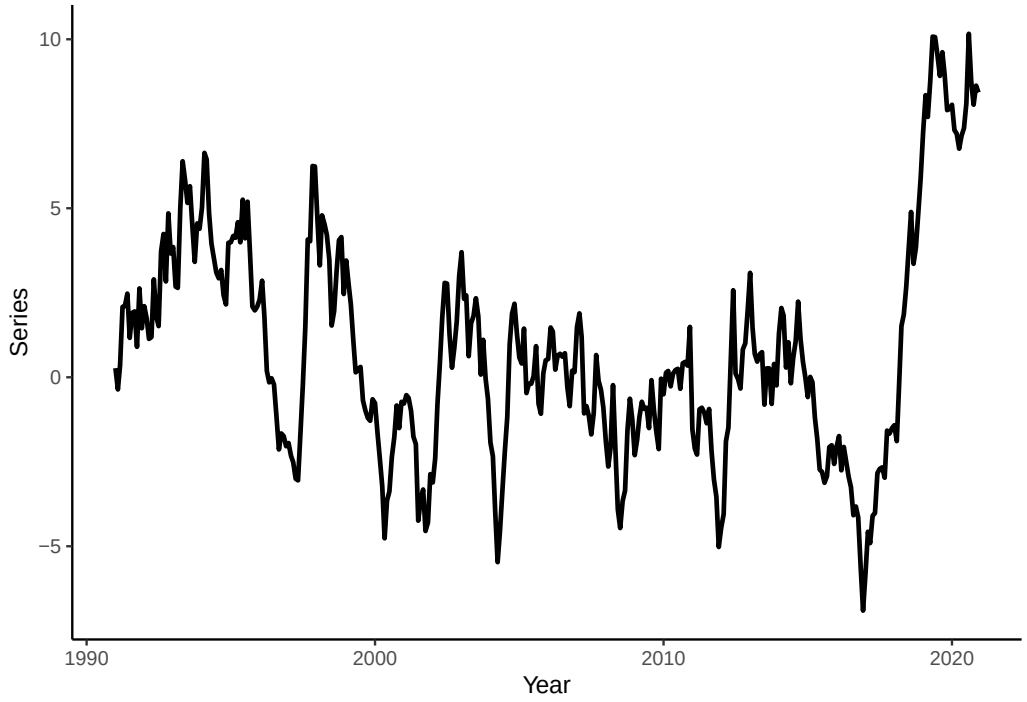
```
}else{
  y[2] <- 1.2*y[1]+e[2]
}

for(i in 3:n){
  if(y[i-1]>=0){
    y[i] <-  1.0*y[i-1]+e[i]
  }else{
    y[i] <- 1.2*y[i-1]-0.3*y[i-2]+e[i]
  }
}
```

Generate a vector of some arbitrary dates (e.g., suppose we deal with the monthly series beginning from January 1991), and store these along with y in a **data.table**, call it 'dt.'

```
date <- seq(as.Date("1991-01-01"),by="month",along.with=y)

dt <- data.table(date,y)
```

Plot the realized time series using **ggplot** function.

```
ggplot(dt,aes(x=date,y=y))+
  geom_line(size=1)+
  labs(x="Year",y="Series")+
  theme_classic()
```



Decide on the optimal lag length based on AIC.

```
dt[,`:=`(y_l1=shift(y),y_l2=shift(y,2),y_l3=shift(y,3),y_l4=shift(y,4))]
dt <- dt[complete.cases(dt)]

IC_dt <- data.table(lag=c(1:4),AIC=as.numeric(NA),SIC=as.numeric(NA))

for(i in 1:nrow(IC_dt)){

  fmla <- as.formula(paste("y",paste0("y_l",c(1:i),collapse="+"),sep="~"))
  reg.ar <- lm(fmla,data=dt)

  IC_dt$AIC[i] <- log(crossprod(reg.ar$residuals))+2*(i+1)/nrow(dt)
  IC_dt$SIC[i] <- log(crossprod(reg.ar$residuals))+log(nrow(dt))*(i+1)/nrow(dt)

}

IC_dt
```

```
##    lag      AIC      SIC
## 1:   1 5.822589 5.844358
## 2:   2 5.821342 5.853996
## 3:   3 5.826916 5.870454
## 4:   4 5.832294 5.886717
```

We now need to get an estimate of the threshold parameter (i.e., the value at which the switch between the regimes happens). For that, we perform a grid-search routine. We will consider a range of candidate thresholds that are within 10th and 90th percentile of the lagged dependent variable. For each candidate threshold, we will run an OLS and calculate the residual sums of squares. A threshold that yields the lowest residual sum of squares will be the estimate.

```
qy <- round(quantile(dt$y,c(.1,.9)),1)

tr <- seq(qy[1],qy[2],by=.1)

grid_dt <- data.table(tr,ssr=NA)
grid_dt[,`:=`(ssr=as.numeric(ssr))]

for(i in tr){

  dt[,`:=`(d=ifelse(y_l1>=i,1,0))]

  tar <- lm(y~(y_l1+y_l2):I(d)+(y_l1+y_l2):I(1-d),data=dt)

  grid_dt[tr==i]$ssr <- crossprod(tar$residuals)

}

tr_hat <- grid_dt[ssr==min(ssr)]$tr
tr_hat
```

```
## [1] -1
```

Estimate the threshold autoregression and compare the parameter estimates with the true parameters of the model.

```
dt[,`:=`(d=ifelse(y_l1>=tr_hat,1,0))]

tar <- lm(y~(y_l1+y_l2):I(d)+(y_l1+y_l2):I(1-d),data=dt)
summary(tar)
```

```
##
## Call:
## lm(formula = y ~ (y_l1 + y_l2):I(d) + (y_l1 + y_l2):I(1 - d),
##     data = dt)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.97704 -0.60528 -0.02506  0.61754  2.66733
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -0.01380    0.07519  -0.183 0.854535
## y_l1:I(d)      0.97060    0.06283  15.449  < 2e-16 ***
## y_l2:I(d)      0.00998    0.06160   0.162 0.871403
## y_l1:I(1 - d)  1.24182    0.10229  12.140  < 2e-16 ***
## y_l2:I(1 - d) -0.34188    0.10158  -3.366 0.000848 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9586 on 351 degrees of freedom
## Multiple R-squared:  0.916,  Adjusted R-squared:  0.9151
## F-statistic: 957.5 on 4 and 351 DF,  p-value: < 2.2e-16
```

Generate a sequence of one-step-ahead forecasts from TAR(2) using the rolling window scheme, where the first rolling window ranges from period 1 to period 240. For comparison, also generate the one-step-ahead forecasts from the AR(2) and the random walk models. Calculate the RMSFE measures for the three models.

```
R <- 240
P <- nrow(dt)-R

dt[,`:=`(rw=as.numeric(NA),ar=as.numeric(NA),tar=as.numeric(NA))]
```

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

  ar <- lm(y~y_l1+y_l2,data=dt[i:(R-1+i)])

  tar <- lm(y~(y_l1+y_l2):I(d)+(y_l1+y_l2):I(1-d),data=dt[i:(R-1+i)])

  dt$rw[R+i] <- dt$y[R-1+i]

  dt$ar[R+i] <- ar$coefficients[1]+ar$coefficients[2]*dt$y[R-1+i]+ar$coefficients[

  if(dt$y[R-1+i]>=0){
    dt$tar[R+i] <- tar$coefficients[1]+tar$coefficients[2]*dt$y[R-1+i]+tar$coeffic
  }else{
    dt$tar[R+i] <- tar$coefficients[1]+tar$coefficients[4]*dt$y[R-1+i]+tar$coeffic
  }

}

dt[,`:=`(rw_e=y-rw,ar_e=y-ar,tar_e=y-tar)]

rmsfe_rw <- sqrt(mean(dt$rw_e^2,na.rm=T))
rmsfe_ar <- sqrt(mean(dt$ar_e^2,na.rm=T))
rmsfe_tar <- sqrt(mean(dt$tar_e^2,na.rm=T))

rmsfe_rw
```

```
## [1] 0.9258102
```

```
rmsfe_ar
```

```
## [1] 0.9811523
```

```
rmsfe_tar
```

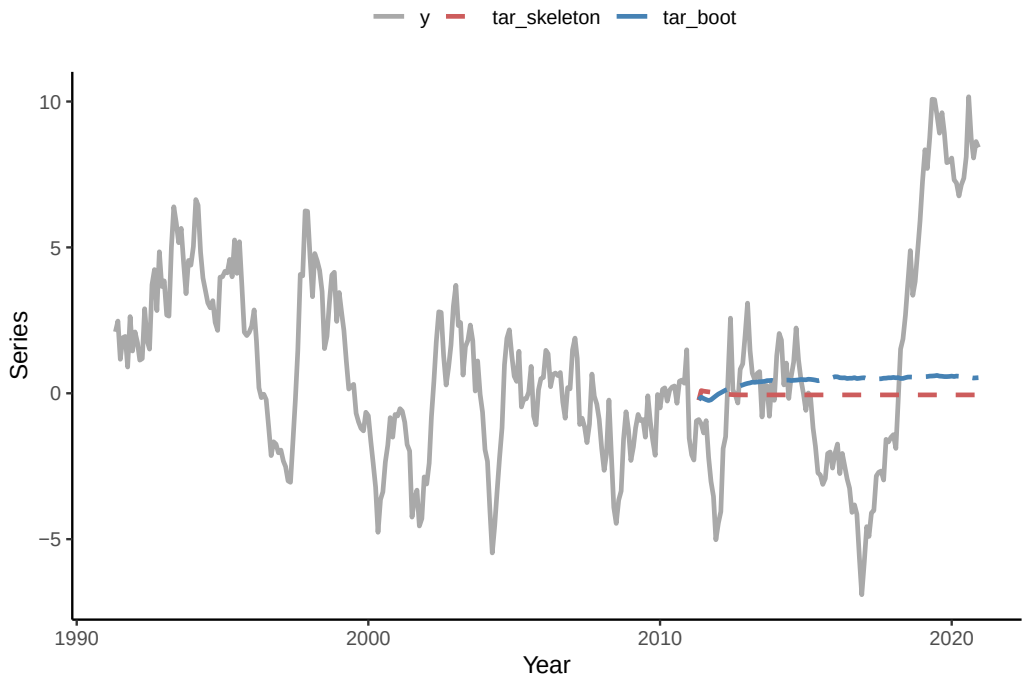
```
## [1] 0.9996084
```

Obtain the multi-step-ahead forecasts from period 241 onward using the so-called 'skeleton extrapolation' method (which yields biased forecasts) and the bootstrap resampling method (a numerical method that yields valid multi-

step-ahead forecasts from nonlinear models). Plot the two forecasts along with the time series.

```
dt[,`:=`(tar_skeleton=y)]

tar <- lm(y~(y_l1+y_l2):I(d)+(y_l1+y_l2):I(1-d),data=dt[1:R])

for(i in 1:P){

  if(dt$tar_skeleton[R-1+i]>=0){
    dt$tar_skeleton[R+i] <- tar$coefficients[1]+tar$coefficients[2]*dt$tar_skeleto
  }else{
    dt$tar_skeleton[R+i] <- tar$coefficients[1]+tar$coefficients[4]*dt$tar_skeleto
  }

}

dt[1:R]$tar_skeleton <- NA


B <- 5000 # the number of bootstrap simulations

boot_mat <- replicate(B,dt$y)

for(b in 1:B){
  eps <- sample(tar$residuals,P,replace=T)

  for(i in 1:P){

    if(boot_mat[R-1+i,b]>=0){
      boot_mat[R+i,b] <- tar$coefficients[1]+tar$coefficients[2]*boot_mat[R-1+i,b]
    }else{
      boot_mat[R+i,b] <- tar$coefficients[1]+tar$coefficients[4]*boot_mat[R-1+i,b]
    }

  }

}
```

```
dt$tar_boot <- rowMeans(boot_mat)
dt[1:R]$tar_boot <- NA

sub_lg <- melt(dt[,.(date,y,tar_skeleton,tar_boot)],id.vars="date")

ggplot(sub_lg,aes(x=date,y=value,color=variable,linetype=variable))+
  geom_line(size=1,na.rm=T)+
  scale_color_manual(values=c("darkgray","indianred","steelblue"))+
  scale_linetype_manual(values=c(1,2,5))+
  labs(x="Year",y="Series")+
  theme_classic()+
  theme(legend.position="top",legend.title=element_blank())
```



Tutorial 8: Threshold Autoregression

(this is a evaluation stuff, will need to move back)

In this tutorial, we will generate a time series, we will obtain one-step-ahead forecasts from competing models using a rolling window procedure, and we will perform the Diebold-Mariano type regression-based test for equal predictive ability of the competing models. To run the code, the `data.table`, `ggplot2`, `lmtest`, and `sandwich` packages need to be installed and loaded.

Let's generate a time series that follow an AR(2) process of the following form:

$$y_t = 0.2 + 1.1y_{t-1} - 0.3y_{t-2} + \varepsilon_t$$

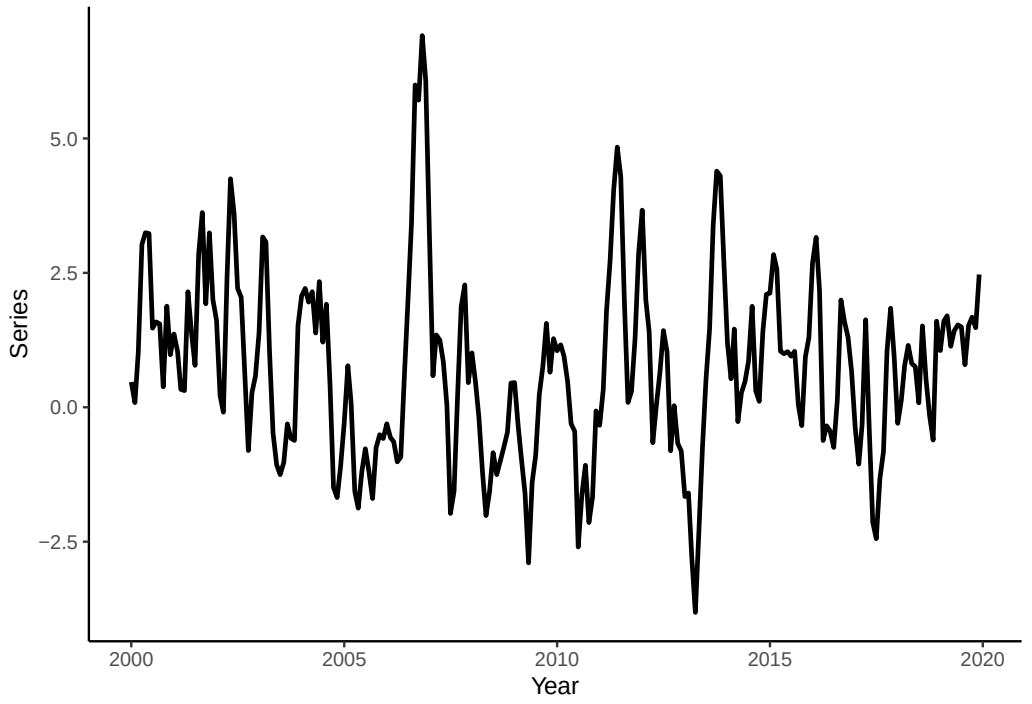
where $e_t \sim N(0, \sigma^2)$.

```
n <- 240

set.seed(6)
e <- rnorm(n,0,1)

y <- rep(NA,n)
y[1] <- 0.2+e[1]
y[2] <- 0.2+1.1*y[1]+e[2]
for(i in 3:n){
  y[i] <- 0.2+1.1*y[i-1]-0.3*y[i-2]+e[i]
}
```

Generate a vector of some arbitrary dates (e.g., suppose we deal with the monthly series beginning from January 2000), store these along with y in

a **data.table**, call it 'dt,' and plot the realized time series using **ggplot** function.

```
date <- seq(as.Date("2000-01-01"),by="month",along.with=y)

dt <- data.table(date,y)

ggplot(dt,aes(x=date,y=y))+
  geom_line(size=1)+
  labs(x="Year",y="Series")+
  theme_classic()
```



Suppose the candidate models are AR(1), AR(2), and AR(3), and that we want to compare forecasts obtained from these models to those from a random walk process. Generate a sequence of one-step-ahead forecasts using the rolling window scheme, where the first rolling window ranges from period 1 to period 180. Calculate the RMSFE measures for the canddidate models.

```
dt[,`:=`(y1=shift(y,1),y2=shift(y,2),y3=shift(y,3))]

R <- 180
P <- nrow(dt)-R

dt[,`:=`(rw=as.numeric(NA),a1=as.numeric(NA),a2=as.numeric(NA),a3=as.numeric(NA))]

for(i in 1:P){

  dt$rw[R+i] <- dt$y[R+i-1]

  ar1 <- lm(y~y1,data=dt[i:(R+i-1)])
  ar2 <- lm(y~y1+y2,data=dt[i:(R+i-1)])
  ar3 <- lm(y~y1+y2+y3,data=dt[i:(R+i-1)])

  dt$a1[R+i] <- ar1$coefficients%*%as.numeric(c(1,dt[R+i,c("y1")]))
  dt$a2[R+i] <- ar2$coefficients%*%as.numeric(c(1,dt[R+i,c("y1","y2")]))
  dt$a3[R+i] <- ar3$coefficients%*%as.numeric(c(1,dt[R+i,c("y1","y2","y3")]))

}

dt$rw_e <- dt$y-dt$rw
dt$a1_e <- dt$y-dt$a1
dt$a2_e <- dt$y-dt$a2
dt$a3_e <- dt$y-dt$a3

# RMSFEs
sqrt(mean(dt$rw_e^2,na.rm=T))
```

```
## [1] 0.9653331
```

```
sqrt(mean(dt$a1_e^2,na.rm=T))
```

```
## [1] 0.9053279
```

```
sqrt(mean(dt$a2_e^2,na.rm=T))
```

```
## [1] 0.8842908
```

```
sqrt(mean(dt$a3_e^2,na.rm=T))
```

```
## [1] 0.8877883
```

Do the autoregressive models generate 'statistically significantly' more accurate forecasts than the random walk model? We will answer this question by performing the regression-based Diebold-Mariano tests. First we will generate the loss differentials; then we will run three separate regressions to assess predictive accuracy of AR(1), AR(2), and AR(3) relative to the random walk; and finally we will base our decision on the heteroskedasticity and autocorrelation consistent standard errors.

```
dt$ld1 <- dt$rw_e^2-dt$a1_e^2
dt$ld2 <- dt$rw_e^2-dt$a2_e^2
dt$ld3 <- dt$rw_e^2-dt$a3_e^2

reg.ld1 <- lm(ld1~1,data=dt)
reg.ld2 <- lm(ld2~1,data=dt)
reg.ld3 <- lm(ld3~1,data=dt)

coeftest(reg.ld1,vcov.=vcovHAC(reg.ld1))
```

```
##
## t test of coefficients:
##
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.112249   0.047525  2.3619   0.0215 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
coeftest(reg.ld2,vcov.=vcovHAC(reg.ld2))
```

```
##
## t test of coefficients:
##
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.149898   0.083086  1.8041  0.07632 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
coeftest(reg.ld3,vcov.=vcovHAC(reg.ld3))
```

```
##
## t test of coefficients:
##
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.143700   0.081309  1.7673  0.08235 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Tutorial 9: Forecast Evaluation

(this is a combination stuff, will need to move back)

In this tutorial, we will generate a time series, we will obtain one-step-ahead forecasts from a set of models using a rolling window procedure, we will combine these forecasts and assess the accuracy of the combined forecast. To run the code, the `data.table`, `ggplot2`, `lmtest`, and `sandwich` packages need to be installed and loaded.

Let's generate a time series that follow an AR(2) process with the quadratic trend component as follows:

$$y_t = 0.03t - 0.0001t^2 + 0.6y_{t-1} + 0.2y_{t-2} + \varepsilon_t$$

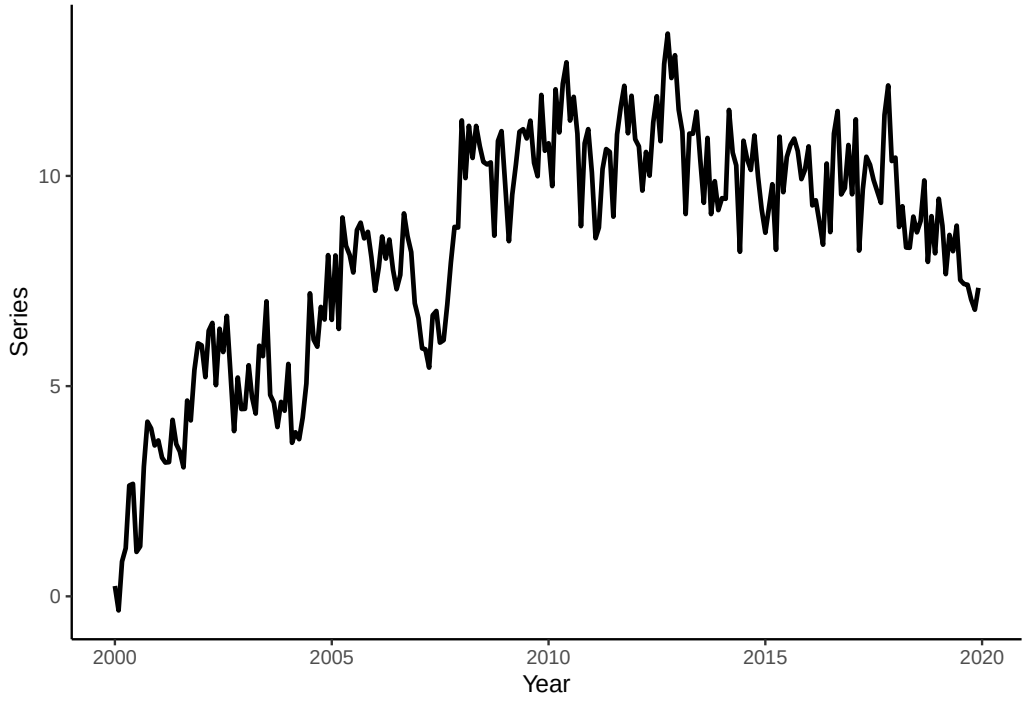
where $e_t \sim N(0, \sigma^2)$.

```
n <- 240

set.seed(4)
e <- rnorm(n,0,1)

y <- rep(NA,n)
y[1] <- 0.03*1-0.0001*(1^2)+e[1]
y[2] <- 0.03*2-0.0001*(2^2)+0.6*y[1]+e[2]
for(i in 3:n){
  y[i] <- 0.03*i-0.0001*(i^2)+0.6*y[i-1]+0.2*y[i-2]+e[i]
}
```

Generate a vector of some arbitrary dates (e.g., suppose we deal with the monthly series beginning from January 2000), store these along with y in

a **data.table**, call it 'dt,' and plot the realized time series using **ggplot** function.

```
date <- seq(as.Date("2000-01-01"),by="month",along.with=y)

dt <- data.table(date,y)

ggplot(dt,aes(x=date,y=y))+
  geom_line(size=1)+
  labs(x="Year",y="Series")+
  theme_classic()
```



Suppose the candidate models are AR(1), AR(2), and a linear trend model, and that we want to compare forecasts obtained from these models to those from a random walk process. Generate a sequence of one-step-ahead forecasts using the rolling window scheme, where the first rolling window ranges from period 1 to period 180.

```
dt[,`:=`(y1=shift(y,1),y2=shift(y,2),y3=shift(y,3),trend=c(1:nrow(dt)))]

R <- 180
P <- nrow(dt)-R

dt[,`:=`(rw=as.numeric(NA),a1=as.numeric(NA),a2=as.numeric(NA),tr=as.numeric(NA))]

for(i in 1:P){

  dt$rw[R+i] <- dt$y[R+i-1]

  a1 <- lm(y~y1,data=dt[i:(R+i-1)])
  a2 <- lm(y~y1+y2,data=dt[i:(R+i-1)])
  tr <- lm(y~y1+y2+trend,data=dt[i:(R+i-1)])

  dt$a1[R+i] <- a1$coefficients%*%as.numeric(c(1,dt[R+i,c("y1")]))
  dt$a2[R+i] <- a2$coefficients%*%as.numeric(c(1,dt[R+i,c("y1","y2")]))
  dt$tr[R+i] <- tr$coefficients%*%as.numeric(c(1,dt[R+i,c("y1","y2","trend")]))

}
```

Does either of the considered models 'sttaistically significantly' outperform the random walk?

```
dt$rw_e <- dt$y-dt$rw
dt$a1_e <- dt$y-dt$a1
dt$a2_e <- dt$y-dt$a2
dt$tr_e <- dt$y-dt$tr

dt$ld1 <- dt$rw_e^2-dt$a1_e^2
dt$ld2 <- dt$rw_e^2-dt$a2_e^2
dt$ldt <- dt$rw_e^2-dt$tr_e^2

reg.ld1 <- lm(ld1~1,data=dt)
reg.ld2 <- lm(ld2~1,data=dt)
reg.ldt <- lm(ldt~1,data=dt)

coeftest(reg.ld1,vcov.=vcovHAC(reg.ld1))
```

```
##
## t test of coefficients:
##
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.081204   0.048040  1.6903  0.09624 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
coeftest(reg.ld2,vcov.=vcovHAC(reg.ld2))
```

```
##
## t test of coefficients:
##
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.32634    0.14912  2.1883  0.03262 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
coeftest(reg.ldt,vcov.=vcovHAC(reg.ldt))
```

```
##
## t test of coefficients:
##
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.24461    0.14943  1.6369    0.107
```

All of the models do, on average, generate more accurate forecasts than random walk. The AR(2) generates statistically significantly more accurate forecasts, based on Diebold-Mariano test applied on quadratic loss function.

Might each model contain some useful information for improving forecast accuracy? Let's combine the forecasts from the AR(1) and the linear trend model using equal weights scheme and assess the combined forecast.

```
dt$t1 <- dt$a1*.5+dt$tr*.5
dt$t1_e <- dt$y-dt$t1

dt$ldt1 <- dt$rw_e^2-dt$t1_e^2

reg.ldt1 <- lm(ldt1~1,data=dt)
```

```
coeftest(reg.ldt1,vcov.=vcovHAC(reg.ldt1))
```

```
##
## t test of coefficients:
##
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.232926   0.098544  2.3637  0.02141 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Tutorial 10: Forecast Combination

(this is a direct/iterated stuff, will need to move forward)

In this tutorial, we will generate a time series, we will obtain multi-step-ahead forecasts using direct and iterated methods from autoregressive models using a rolling window procedure, and we will assess the accuracy of the forecast. We will then turn to interval forecasts, and we will assess their coverage accuracy. To run the code, the `data.table`, `ggplot2`, `lmtest`, and `sandwich` packages need to be installed and loaded.

Let's generate a time series that follow an AR(2) process as follows:

$$y_t = 0.01t + 0.6y_{t-1} + 0.2y_{t-2} + \varepsilon_t$$

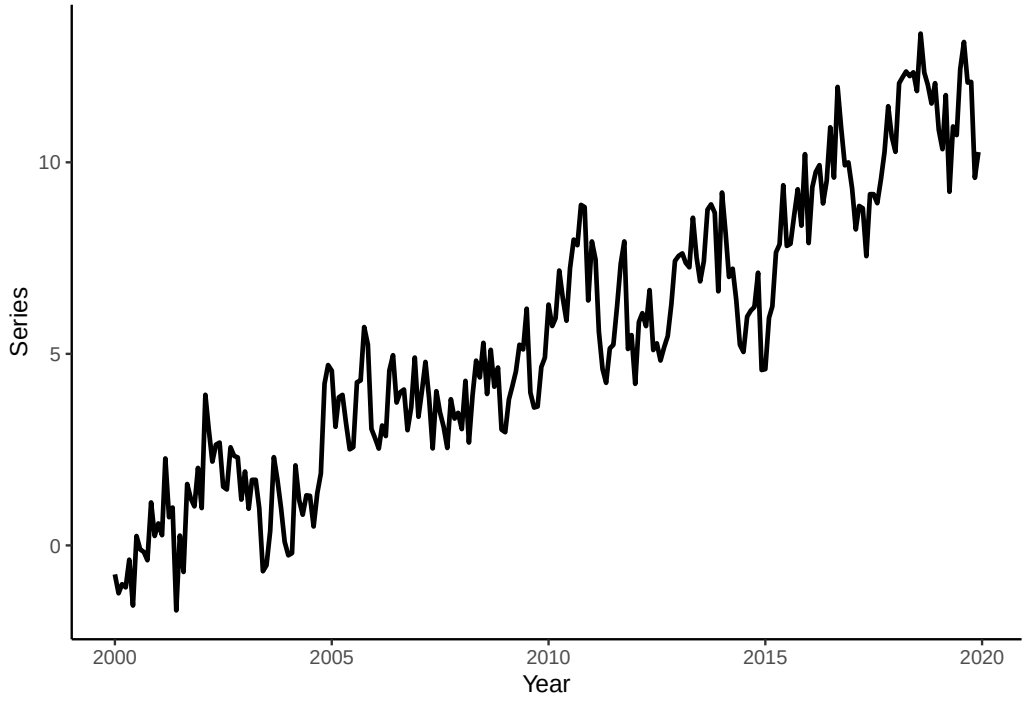
where $e_t \sim N(0, 1)$.

```
n <- 240

set.seed(9)
e <- rnorm(n,0,1)

y <- rep(NA,n)
y[1] <- 0.01+e[1]
y[2] <- 0.02+0.6*y[1]+e[2]
for(i in 3:n){
  y[i] <- 0.01*i+0.6*y[i-1]+0.2*y[i-2]+e[i]
}
```

Generate a vector of arbitrary dates (e.g., the monthly series beginning from

January 2000), store these along with y in a **data.table**, call it 'dt,' and plot the realized time series using the **ggplot** function.

```
date <- seq(as.Date("2000-01-01"),by="month",along.with=y)

dt <- data.table(date,y)

ggplot(dt,aes(x=date,y=y))+
  geom_line(size=1)+
  labs(x="Year",y="Series")+
  theme_classic()
```



Suppose we believe the series follow either the AR(1) or the AR(2) process, and that we want to compare iterated multi-step forecasts obtained from these models to direct multi-step forecasts. In what follows, we will generate twelve-step-ahead point forecasts from these two methods for the considered two models using the rolling window scheme, where the first rolling window ranges from period 1 to period 180. In obtaining the iterated multi-step forecasts, for each rolling window, we will generate a sequence of one-to-twelve-step-ahead

forecasts, but only retain the twelve-step-ahead forecasts.

```
h <- 12

dt <- dt[,.(date,y,y1=shift(y,1),y2=shift(y,2),y12=shift(y,h),y13=shift(y,h+1))]
dt <- dt[complete.cases(dt)]

dt[,`:=`(a1i=as.numeric(NA),a1d=as.numeric(NA),a2i=as.numeric(NA),a2d=as.numeric(N

R <- 180
P <- nrow(dt)-R-h+1

for(i in 1:P){

  ### iterated multi-step method
  a1i <- lm(y~y1,data=dt[i:(R+i-1)])
  a2i <- lm(y~y1+y2,data=dt[i:(R+i-1)])

  iter_dt <- data.table(hor=1:h,a1=as.numeric(NA),a2=as.numeric(NA))

  ## AR(1)
  iter_dt$a1[1] <- a1i$coefficients[1]+a1i$coefficients[2]*dt[R+i-1]$y
  for(j in 2:h){
    iter_dt$a1[j] <- a1i$coefficients[1]+a1i$coefficients[2]*iter_dt$a1[j-1]
  }

  ## AR(2)
  iter_dt$a2[1] <- a2i$coefficients[1]+a2i$coefficients[2]*dt[R+i-1]$y+a2i$coeffic
  iter_dt$a2[2] <- a2i$coefficients[1]+a2i$coefficients[2]*iter_dt$a2[1]+a2i$coeff
  for(j in 3:h){
    iter_dt$a2[j] <- a2i$coefficients[1]+a2i$coefficients[2]*iter_dt$a2[j-1]+a2i$c
  }

  dt$a1i[R+i+h-1] <- iter_dt$a1[h]
  dt$a2i[R+i+h-1] <- iter_dt$a2[h]

  ### direct multi-step method
  a1d <- lm(y~y12,data=dt[i:(R+i-1)])
```

```
  a2d <- lm(y~y12+y13,data=dt[i:(R+i-1)])

  dt$a1d[R+i+h-1] <- a1d$coefficients[1]+a1d$coefficients[2]*dt[R+i-1]$y
  dt$a2d[R+i+h-1] <- a2d$coefficients[1]+a2d$coefficients[2]*dt[R+i-1]$y+a2d$coeff

}
```

We can now test whether the forecasts of the iterated method 'statistically significantly' outperform those of the direct method?

```
dt$a1i_e <- dt$y-dt$a1i
dt$a2i_e <- dt$y-dt$a2i
dt$a1d_e <- dt$y-dt$a1d
dt$a2d_e <- dt$y-dt$a2d

dt$ld1 <- dt$a1d_e^2-dt$a1i_e^2
dt$ld2 <- dt$a2d_e^2-dt$a2i_e^2

reg_ld1 <- lm(ld1~1,data=dt)
reg_ld2 <- lm(ld2~1,data=dt)

coeftest(reg_ld1,vcov.=vcovHAC(reg_ld1))
```

```
##
## t test of coefficients:
##
##             Estimate Std. Error t value  Pr(>|t|)
## (Intercept)  -6.9266     1.7170 -4.0342 0.0002828 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

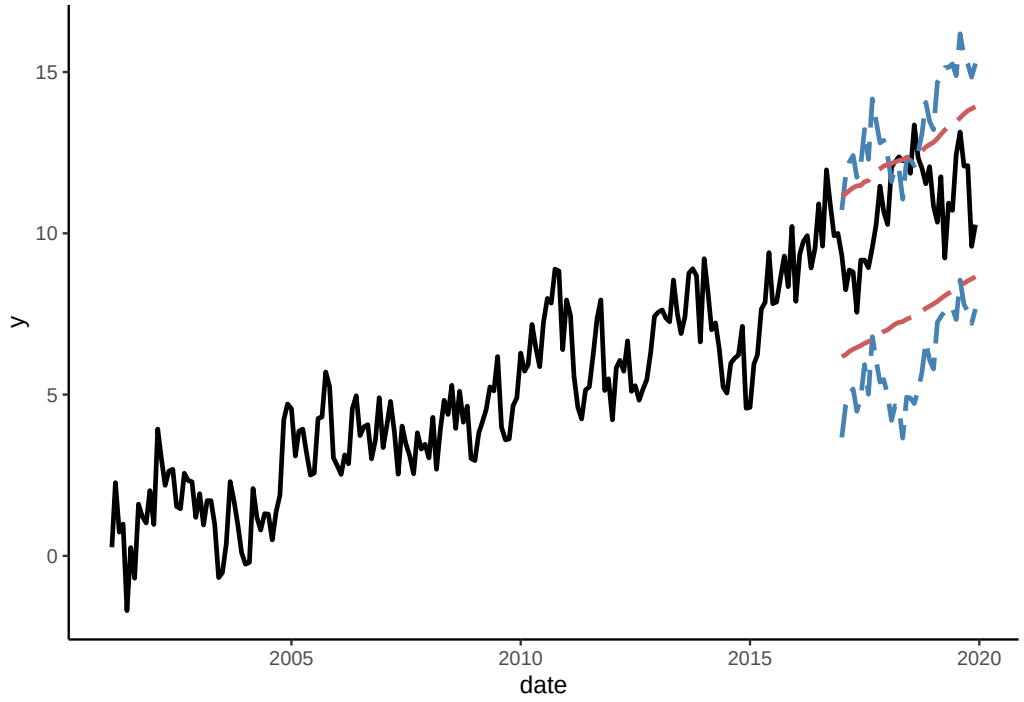
```
coeftest(reg_ld2,vcov.=vcovHAC(reg_ld2))
```

```
##
## t test of coefficients:
##
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -3.1094     1.3271  -2.343  0.02494 *
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Obtain interval forecasts for horizon 12 using the direct method for an AR(1) model and a linear trend model; plot these interval forecasts along with the observed series.

```
dt$trend <- c(1:nrow(dt))
dt[,`:=`(a1dl=as.numeric(NA),a1du=as.numeric(NA),trdl=as.numeric(NA),trdu=as.numer

for(i in 1:P){

  a1d <- lm(y~y12,data=dt[i:(R+i-1)])
  trd <- lm(y~trend,data=dt[i:(R+i-1)])

  a1d_pf <- a1d$coefficients[1]+a1d$coefficients[2]*dt[R+i-1]$y
  trd_pf <- trd$coefficients[1]+trd$coefficients[2]*dt[R+i-1+h]$trend

  a1d_sd <- summary(a1d)$sigma
  trd_sd <- summary(trd)$sigma

  dt$a1dl[R+i+h-1] <- a1d_pf-1.96*a1d_sd
  dt$a1du[R+i+h-1] <- a1d_pf+1.96*a1d_sd

  dt$trdl[R+i+h-1] <- trd_pf-1.96*trd_sd
  dt$trdu[R+i+h-1] <- trd_pf+1.96*trd_sd
}

ggplot(dt,aes(x=date,y=y))+
  geom_line(size=1,color="black")+
  geom_line(aes(y=a1dl),size=1,color="steelblue",linetype=5,na.rm=T)+
  geom_line(aes(y=a1du),size=1,color="steelblue",linetype=5,na.rm=T)+
  geom_line(aes(y=trdl),size=1,color="indianred",linetype=5,na.rm=T)+
  geom_line(aes(y=trdu),size=1,color="indianred",linetype=5,na.rm=T)+
  theme_classic()
```



Test the unconditional coverage of the interval forecasts from the considered two models.

```
dt[,`:=`(i1=ifelse(y>=a1dl & y<=a1du,1,0),it=ifelse(y>=trdl & y<=trdu,1,0))]


# AR(1)
p1 <- mean(dt$i1,na.rm=T)

L10 <- (1-.95)^(P*(1-p1))*(.95)^(P*p1)
L11 <- (1-p1)^(P*(1-p1))*(p1)^(P*p1)

LR1 <- -2*log(L10/L11)

pchisq(LR1,df=1,lower.tail=F)
```

```
## [1] 0.01027851
```

```
# AR(2)
p2 <- mean(dt$it,na.rm=T)

L20 <- (1-.95)^(P*(1-p2))*(.95)^(P*p2)
L21 <- (1-p2)^(P*(1-p2))*(p2)^(P*p2)

LR2 <- -2*log(L20/L21)

pchisq(LR2,df=1,lower.tail=F)
```

```
## [1] 0.1441849
```