Economic Time Series Modeling and Forecasting

David Ubilava

September 2021

Contents

Preamble

These notes are prepared to teach an undergraduate-level course on economic forecasting. The content is presented in four parts. The first part, *Preliminaries*, introduces the concept of forecasting in time series context. The second part, *Deterministic Time Series Models*, covers trends and seasonal models. The third part, *Dynamic Time Series Models*, covers linear, multivariate, and nonlinear autoregressive processes. The fourth part, *Forecast Assessment*, goes over forecast evaluation and combination routines, and forecast accuracy tests.

Preliminaries

Chapter 1

Introduction to Forecasting

Economic events tend to co-occur, precede, or succeed one another. Understanding the essence of such relationships – that is, identifying causal mechanisms that facilitate correlation among economic variables – is at the core of econometric analysis. Throughout a relatively brief history of the study of econometrics, numerous methods and techniques have been proposed and developed – all aimed to give an empirical content to economic models. These methods and techniques allow us to test theories, evaluate policy outcomes, etc.

Econometric models rely on correct (and accurate) identification of the causal mechanism in the underlying process. But they are also predictive by nature – they help us make economic forecasts even when the causal mechanism may not be well identified. In other words, while correlation does not necessarily imply causality, if the goal is to make a forecast, a mere correlation might as well suffice.

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 after it is organized and stored, what we end up with is data. A diverse set of forecasting methods typically rely on insights from econometric analysis of time series – chronologically stored data, collected at regular intervals over a period of time. 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. No matter how rich the available data are, or how well the econometric model fits the data, there still is a surprise element concerning the forecast – something that has never happened in past, and is only specific to 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

Stochastic Process and Time Series

Time series are realizations of a chronologically stored sequence of random variables. This sequence of random variables is referred to as the *stochastic process*. Thus, a time series is a realisation of the stochastic process. We index time periods as $1, 2, \ldots, T$, and denote the set of observations as $\{y_1, \ldots, y_T\}$. One can view a time series as a finite sample from an underlying doubly–infinite sequence: $\{\ldots, y_{-1}, y_0, y_1, y_2, \ldots, y_T, y_{T+1}, y_{T+2}, \ldots\}$.

2.1 Stationarity

If all random variables, from where the time series are drawn, have the same distribution, then we refert to such time series as *stationary*. 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 random variable with zero mean and constant variance: $\varepsilon_t \sim iid\left(0, \sigma^2\right)$. The following graph plots this time series against time.

Such time series are referred to as *white noise*. That is, a time series, y, is a white noise process if:

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

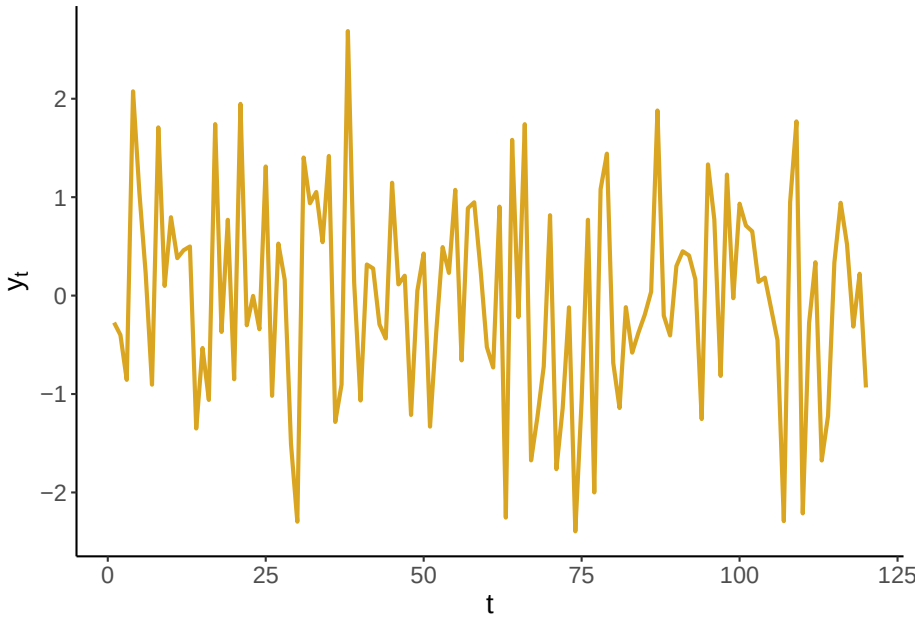


Figure 2.1: White noise: an illustration

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, the mean doesn't need to be zero (as long as it is constant over time), nor covariances need to be equal to zero (as long as they are constant over time, though they may vary with k). 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

In fact, it is more of a norm rather than an 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 correlations are referred to as *autocorrelations*, and are 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:

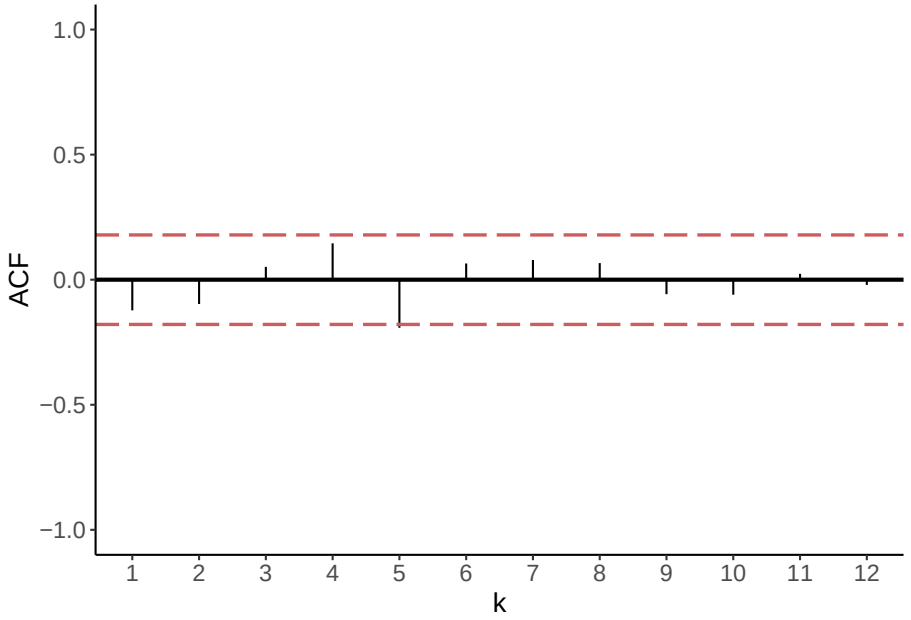


Figure 2.2: Autocorrelation

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 for time series dependence is partial autocorrelation, which is correlation between y_t and y_{t-k} net of any correlations between 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:

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

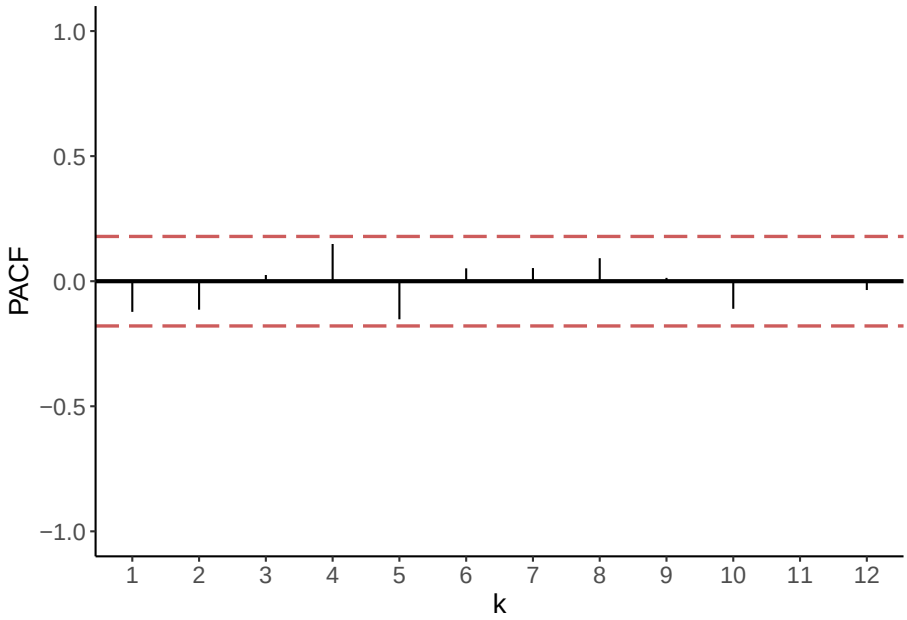


Figure 2.3: Partial Autocorrelation

time series, and (iii) their differences (i.e., the log-differences of the original series).

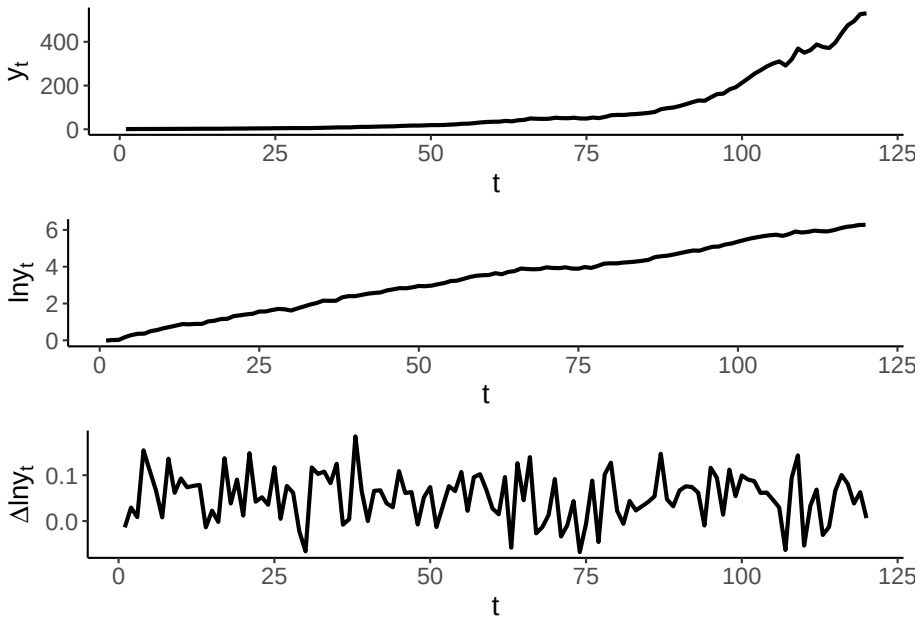


Figure 2.4: A time series and its transformations

Chapter 3

Basics of Forecasting

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:

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

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 used to fit 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.

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*.

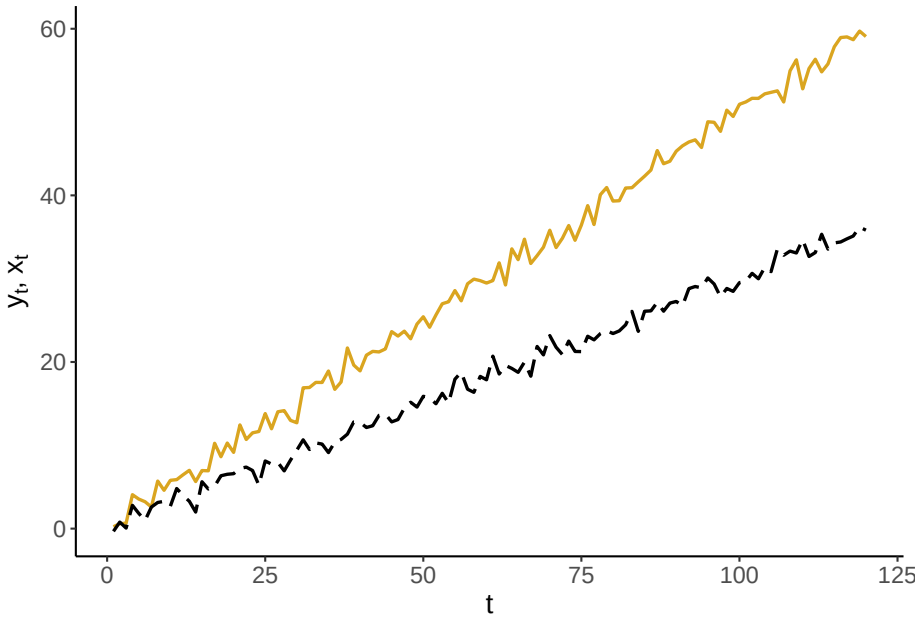
To illustrate, consider two trending variables:

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

and

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

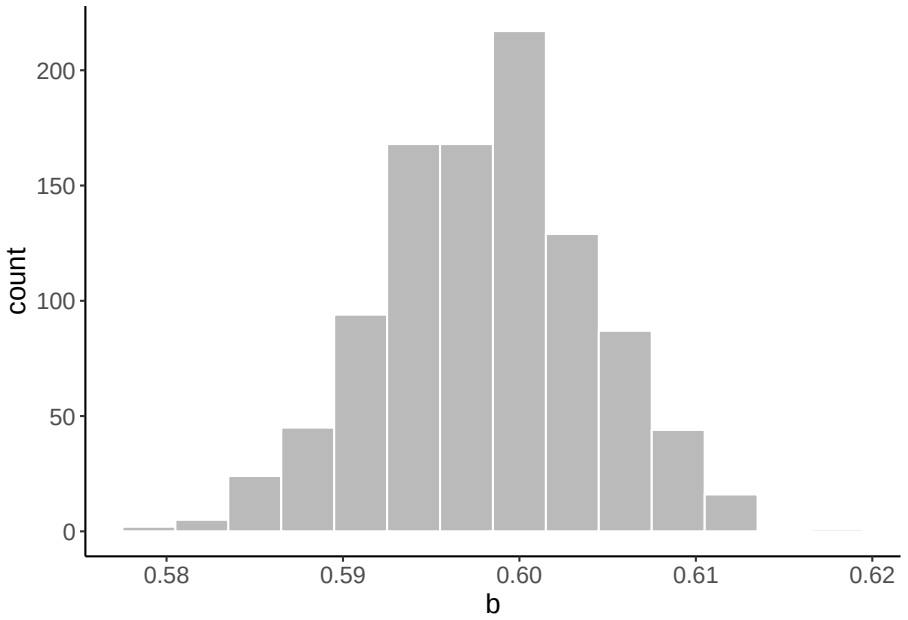
where $Cov(\nu_t, \upsilon_t) = 0$. For simplicity, we can assume $\sigma_\nu^2 = \sigma_\upsilon^2 = 1$. Suppose, γ and δ are some positive scalars, say, 0.3 and 0.5, respectively. That is, y and x are trending in the same direction. Below is an example of such time series:



If we were to estimate

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

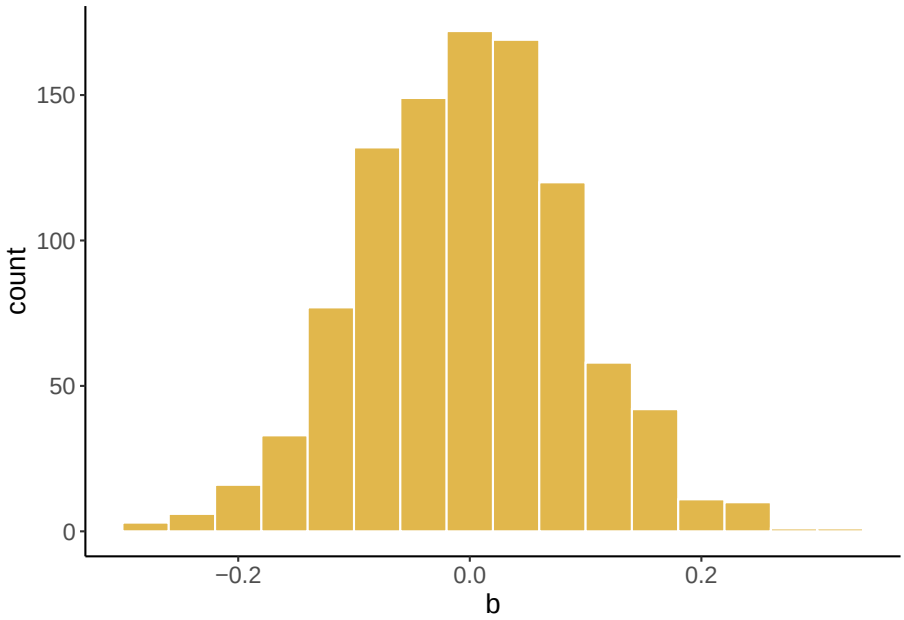
we are likely to find the relationship between the two – in this case $\beta > 0$ – even though, we know, the two are not related. To illustrate this, we will generate 1000 samples of size 120 for y and x, and in each case we will estimate the parameter β. The following graph illustrates the empirical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

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

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.

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.

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

5.2 Forecasting

Any future realization of a random variable that i assumed to follow a seasonal model 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 given by:

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

The interval forecast at any horizon 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:

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

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:

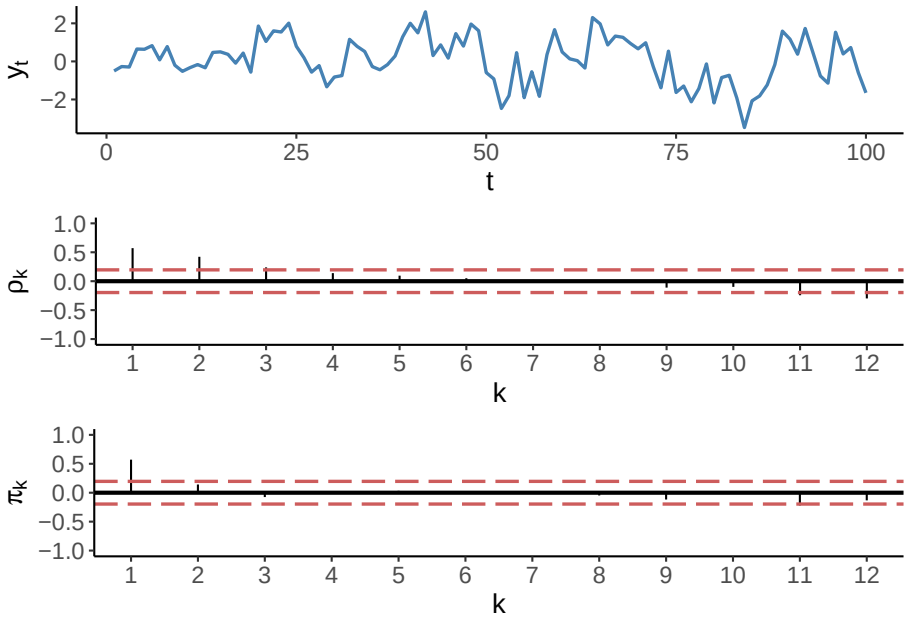
- If $\beta_1 = 0$, y_t is equivalent to a white noise.
- If $\beta_1 = 1$ and $\alpha = 0$, y_t is a random walk.
- If $\beta_1 = 1$ and $\alpha \neq 0$, y_t is a random walk with drift.

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.

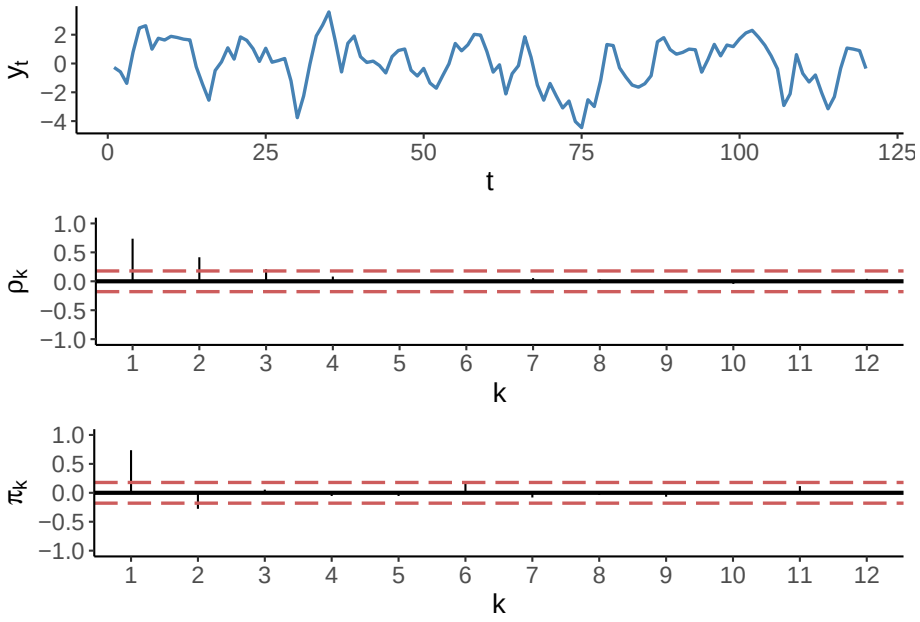
In what follows, the necessary (1 and 2) and sufficient (3 and 4) conditions for an $AR(2)$ process to be covariance stationary are:

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

To generate 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

Chapter 8

Threshold Autoregression

Forecast Assessment

Chapter 9

Forecast Evaluation

Chapter 10

Forecast Combination

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:

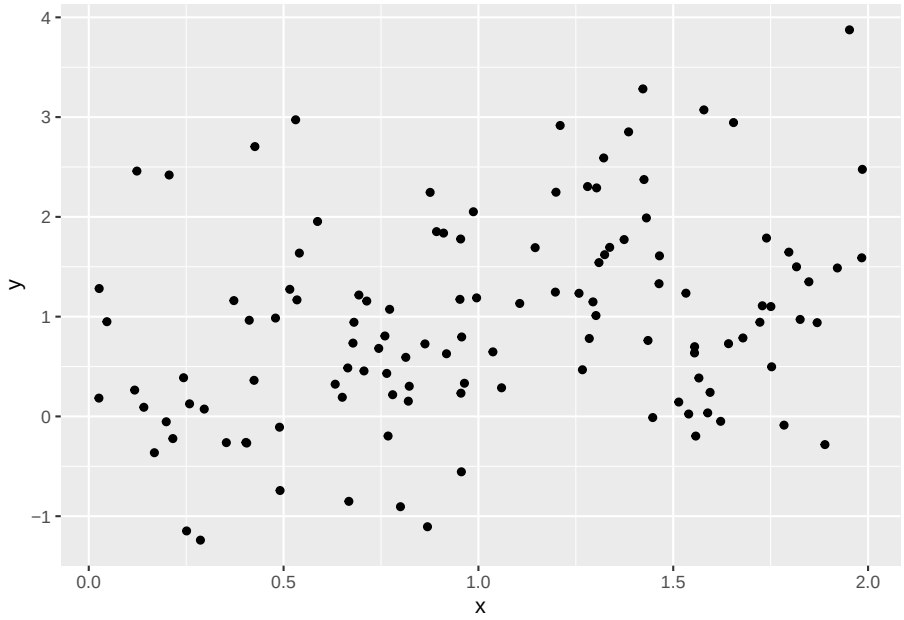
```
set.seed(1)
x <- runif(120,0,2)
y <- 0.2+0.7*x+rnorm(120)

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
```

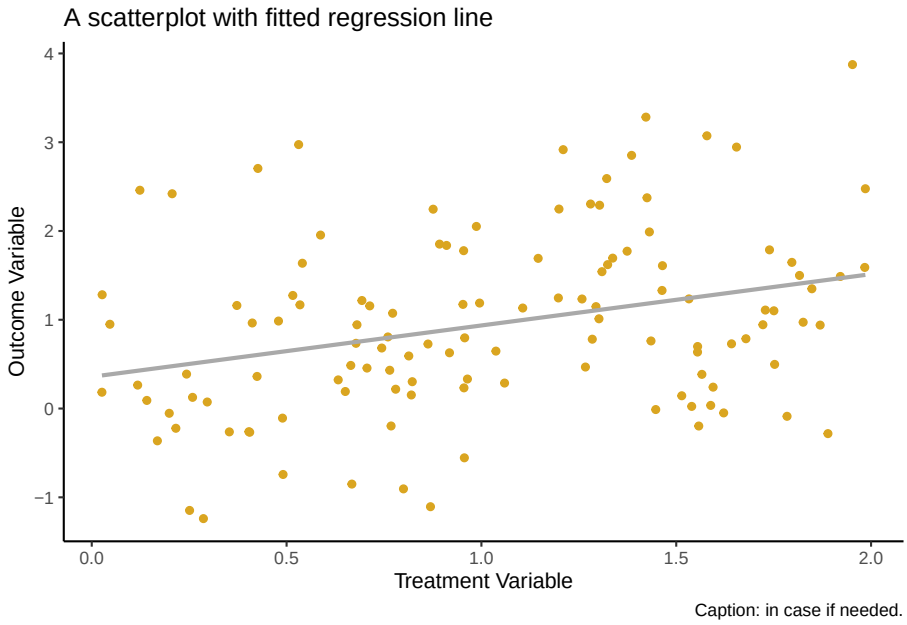
Now, let's load **ggplot2** and generate a simple scatter plot:

```
library(ggplot2)

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



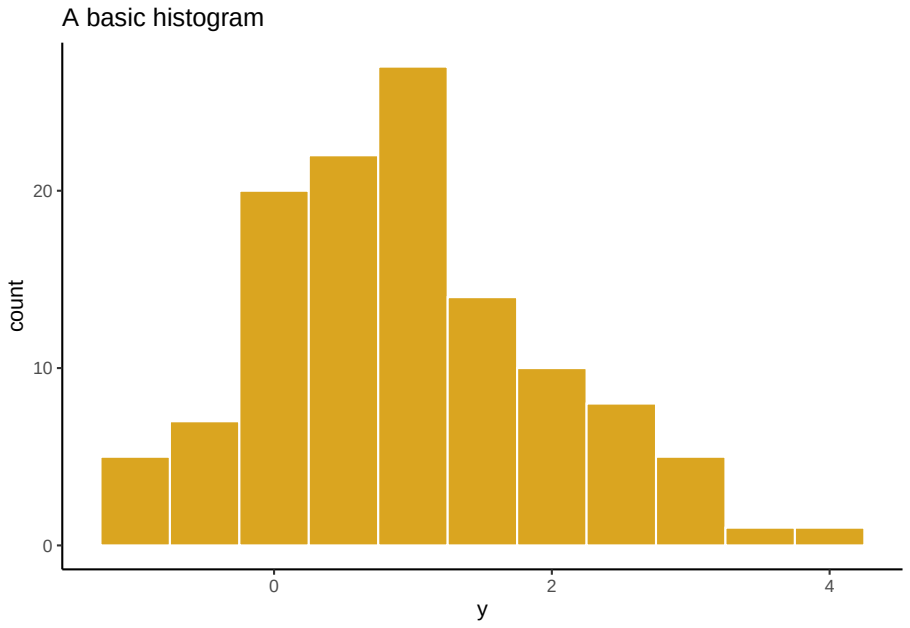
We can augment this plot in a number of different ways. Here we change the point color to red, add the fitted regression line to the plot, add labels to the figure, and apply a 'classic' background theme:

```
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()
```



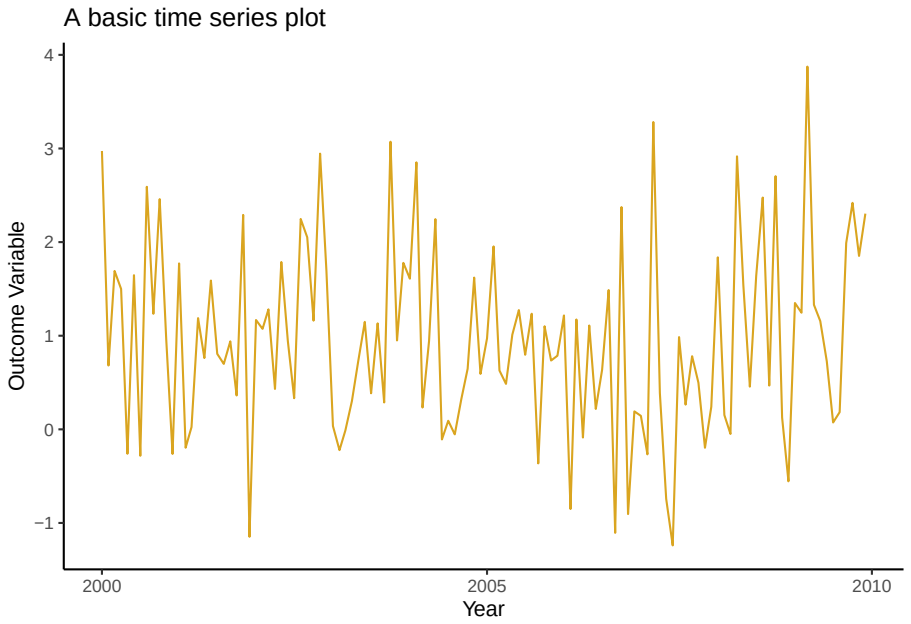
As another example, let's generate a histogram (of the dependent variable):

```
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 1

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 2

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 $t = 1, \ldots, 120$.

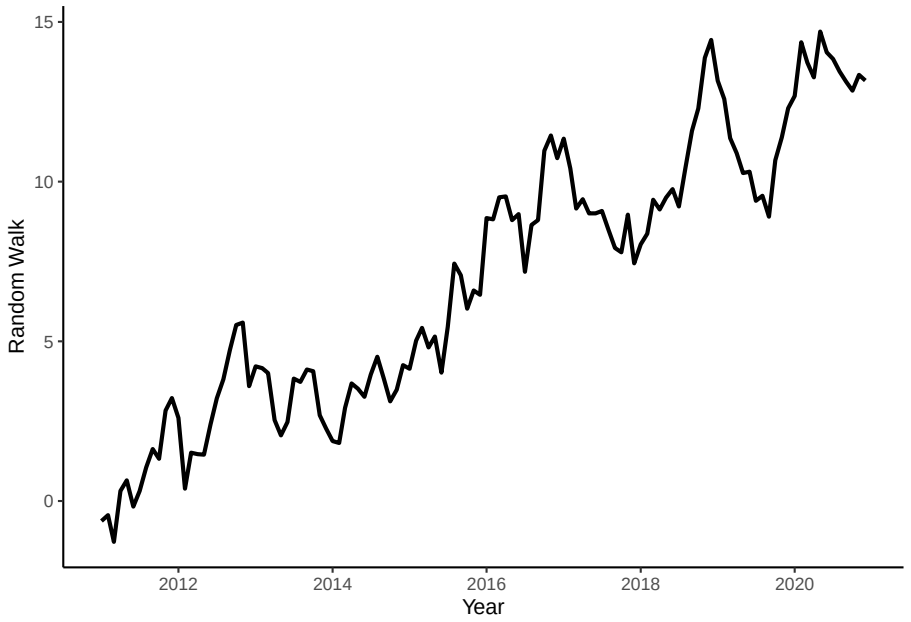
```
n <- 120

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 3

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 $t = 1, \ldots, 180$.

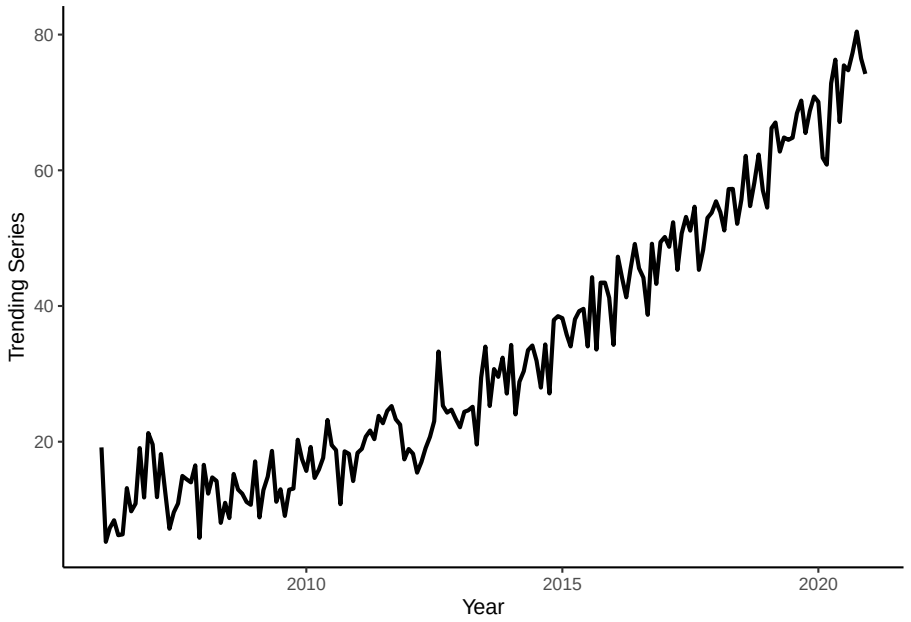
```
n <- 180

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
```

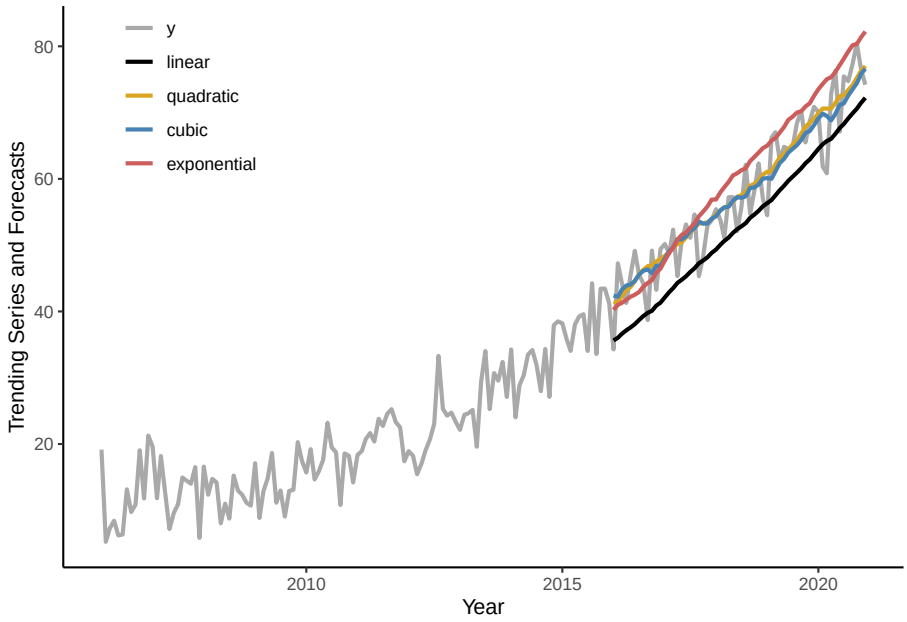
```
##          [,1]     [,2]     [,3]     [,4]
## [1,] 8.879646 7.841645 7.849521 7.975999
```

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]*((R+i)^3

  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.vars="date'

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","indianred"))+
  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 4

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 $t = 1, \ldots, 180$.

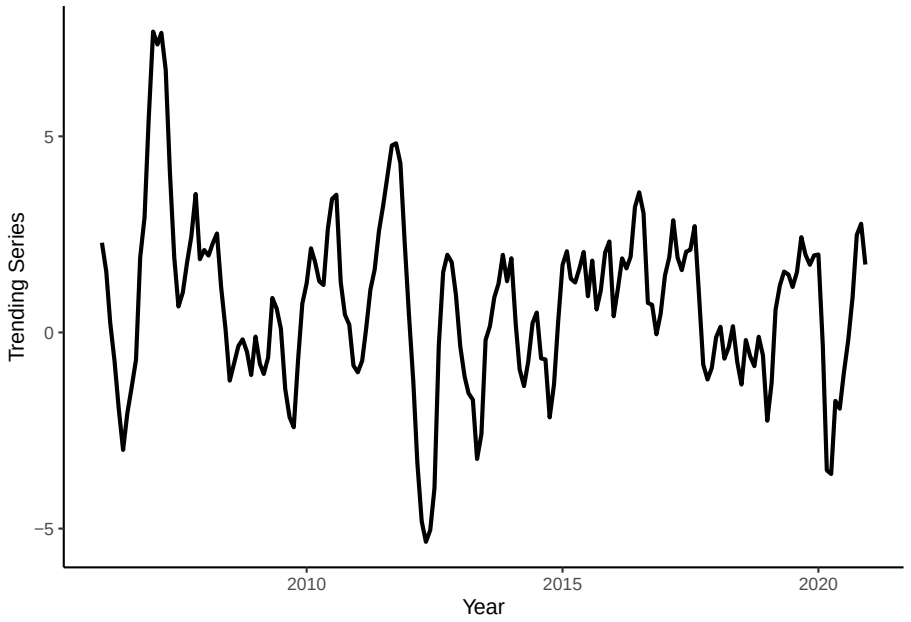
```
n <- 180

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)
```

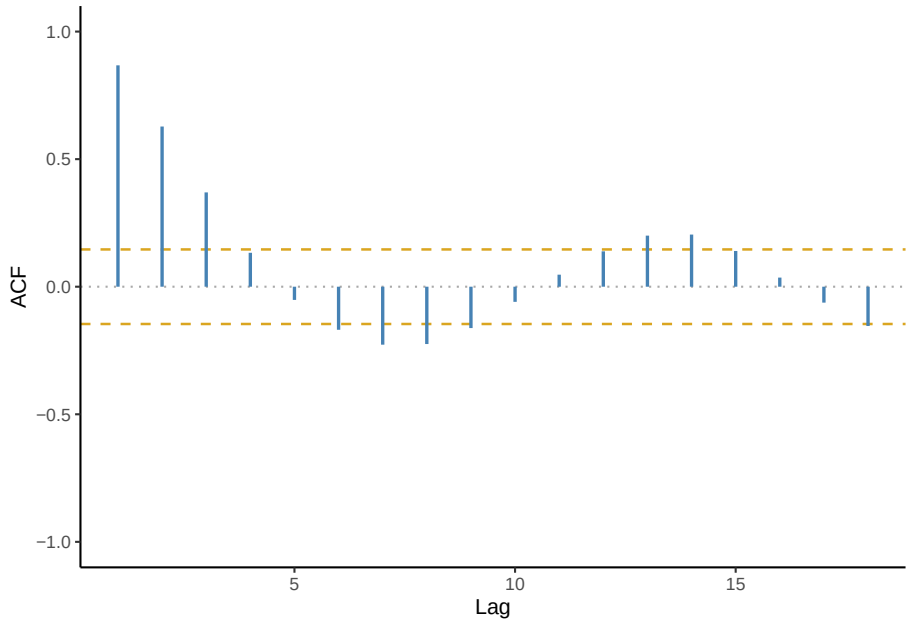
Plot the realized time series using **ggplot** function.

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

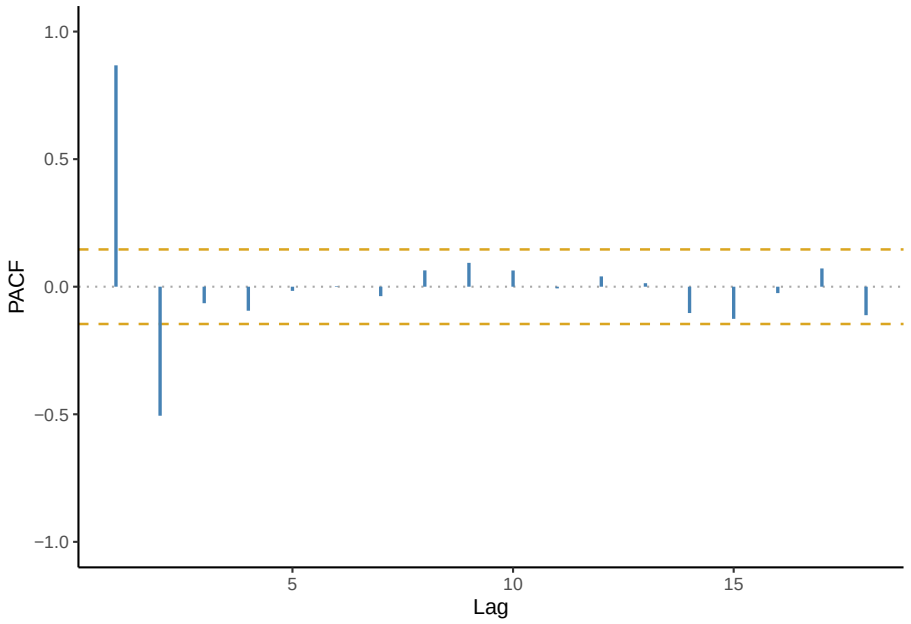


Generate and plot the autocorrelation function and the partial autocorrelation function for lags 1 through 18.

```
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,size=.6) +
  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,size=.6) +
  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 5.298452 5.334480
## 2:   2 5.010470 5.064513
```

```
## 3:   3 5.016720 5.088776
## 4:   4 5.018299 5.108369
```

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$coefficients[3]*dt$
}
```

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] 1.017931
```

```
rmsfe_ar1
```

```
## [1] 0.9782388
```

```
rmsfe_ar2
```

```
## [1] 0.8970604
```

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]+ar2$coefficients[3]*dt$ar2_multi[R-2+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()
```