Module 2 - Analysis of Trends

MATH1318 Time Series Analysis

Prepared by: Dr. Haydar Demirhan based on the textbook by Cryer and Chan, Time Series Analysis with R, Springer, 2008.

Nomenclature

μ: Mean of series.
μ_t: Mean at time point t.
ρ_k: Autocorrelation of series at lag k.
γ_k: Autocovariance of series at lag k.
$\beta_.$: Regression coefficient.
$\hat{\beta}_.$: Estimate of regression coefficient.
$\bar{\cdot}$: Average of inner expression.
$\hat{\cdot}$: Estimate of inner expression.
f: frequency of a curve.
Φ: phase of the curve.
π: the Pi number.
s: Standard deviation.
R^2: coefficient of determination.
r_k: sample autocorrelation function at lag k.

Introduction

Trend in time series is closely related to the mean function of the series. Changes in mean over time create a trend in the series. In general, the mean function is an arbitrary function of time. We will consider relatively simple functions of time to model the trend in time series.

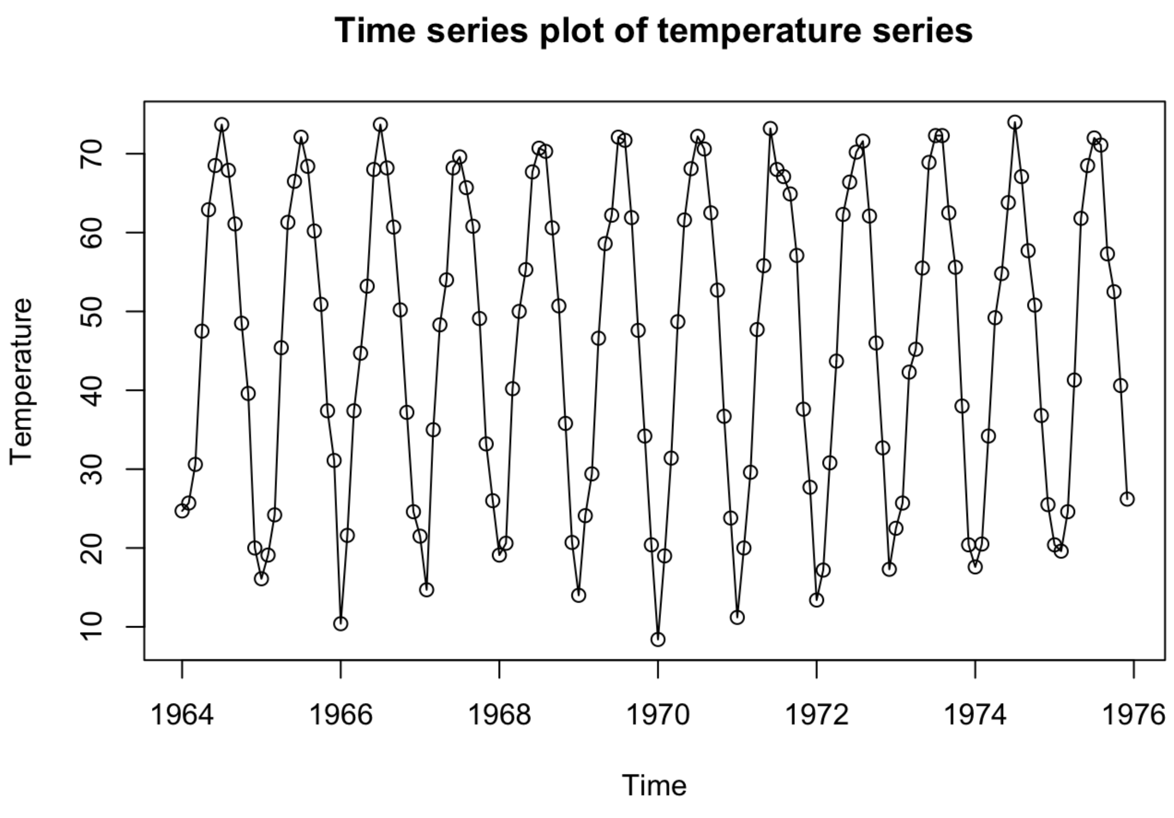
In this module, we will study

- the deterministic and stochastic trend,
- modeling deterministic trends,
- estimation of constant mean,
- regression approach to model the trend,
- analysis of residuals after modelling the trend.

Deterministic Versus Stochastic Trends

One of the challenges in time series analysis is that the same time series may be viewed quite differently by different analysts. For example, one can foresee a trend in a simulated random walk with a constant mean for all time. The perceived trend is a result of the strong positive correlation between the series values at nearby time points and the increasing variance in the process as time goes by. Therefore, one can see different trends in the next simulations. This type of trend is called **stochastic trend**.

In the average monthly temperatures example of the first module, we got the following time series plot:



Here we have a cyclical or seasonal trend, but here the reason for the trend is clear that the Northern Hemisphere's changing inclination toward the sun. We can model this trend by $Y_t = X_t + \mu_t$, where μ_t is a deterministic function that is periodic with period 12 and it should satisfy $\mu_t = \mu_{t-12}$ for all t. We can assume that X_t represent an unobserved variation around μ_t and has zero mean for all t. So, this model assumes that

μ_t is the mean function for the observed series Y_t. Because the mean function is determined beforehand and we can set the functions form of trend, the trend considered here is a **deterministic trend**. It is possible to set a linear mean function such that $\mu_t = \beta_0 + \beta_1 t$ or a quadratic time trend such as $\mu_t = \beta_0 + \beta_1 t + \beta_2 t^2$.

Estimation of a Constant Mean

When we consider a constant mean over time, we set $\mu_t = \mu$, for all t. So, our model is written as

$$Y_t = \mu + X_t$$

Our aim is to estimate the value of μ using the observed series $Y_1, Y_2, \ldots, Y_n$. The straightforward estimate of μ is the sample mean calculated as

$$\bar{Y} = \frac{1}{n}\sum_{t=1}^{n} Y_t$$

Here sample mean is an unbiased estimator of constant mean. To investigate its efficiency, we need to find the variance of the sample mean. Suppose that $\{Y_t\}$ is a stationary time series with autocorrelation function ρ_k. Then, the variance of the sample mean is obtained as

$$Var(\bar{Y}) = \frac{\gamma_0}{n}\left[1 + 2\sum_{k=1}^{n-1}\left(1 - \frac{k}{n}\right)\rho_k\right]$$

Note that if the series $\{Y_t\}$ is just white noise then $\rho_k = 0$ for $k > 0$; and hence, $Var(\bar{Y}$ reduces to simply γ_0/n, which is the population variance divided by the sample size.

Instead of constant mean, we can set a moving average model such that $Y_t = e_t - 1/2e_{t-1}$, which is also stationary. Then, we find that $\rho_1 = -0.4$, which means that we have a negative correlation at lag 1, and $\rho_k = 0$ for $k > 1$. In this case, we have

$$Var(\bar{Y}) = \frac{\gamma_0}{n}\left[1 - 0.8\left(\frac{n-1}{n}\right)\right]$$

For a large n, the correction factor $(n-1)/n$ will approach to 1. Thus, we get

$$Var(\bar{Y}) \approx 0.2\frac{\gamma_0}{n}$$

So, the variance of the estimator of μ for the moving average model is less than that of for the constant mean model: $0.2(\gamma_0/n) < \gamma_0/n$. The reason for getting a more efficient estimator with a moving average model is that in the moving average model, it is possible for the series to oscillate back and forth across the mean. On the other hand, if $\rho_k \geq 0$ for all $k \geq 1$, $Var(\bar{Y})$ will be larger than γ_0/n.

For many stationary processes, the autocorrelation function decays quickly enough with increasing lags. under this assumption and given a large sample, we obtain the following approximation:

$$Var(\bar{Y}) \approx \frac{\gamma_0}{n}\left[\sum_{k=-\infty}^{\infty}\rho_k\right]$$

Here, negative correlations and large sample size both increase the efficiency of the estimator.

We should note that the precision of the sample mean as an estimator of μ can be strikingly different for a nonstationary process with a constant mean. For example, for the random walk process defined in Module 1 (), we find the following:

$$Var(\overline{Y}) = \sigma_e^2(2n+1)\frac{(n+1)}{6n}$$

Notice that in this special case the variance of our estimate of the mean actually increases as the sample size n increases. Because this is unacceptable, we need to consider other estimation techniques for nonstationary series.

Regression Approach

Classical regression analysis can be used to model nonconstant mean trend. We will consider linear, quadratic, seasonal means, and cosine trends.

Linear and Quadratic Trends in Time

The deterministic linear trend model is expressed as follows:

$$\mu_t = \beta_0 + \beta_1 t$$

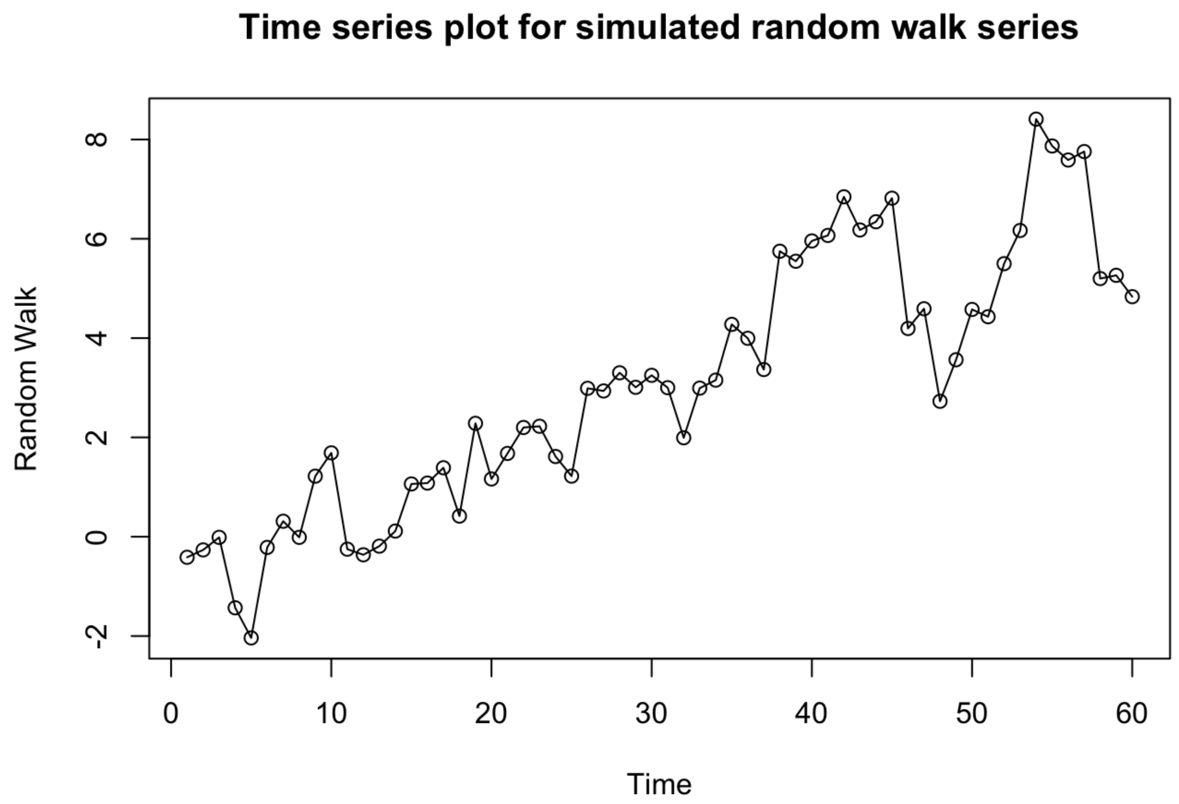
where β_0 represents intercept and β_1 corresponds to the slope of the linear trend. Suppose $\hat{\beta}_0$ and $\hat{\beta}_1$ are the classical least squares estimates of β_0 and β_1, respectively. Then, $\hat{\beta}_0$ and $\hat{\beta}_1$ are obtained as follows:

$$\hat{\beta}_1 = \frac{\sum_{t=1}^{n}(Y_t - \overline{Y})(t - \overline{t})}{\sum_{t=1}^{n}(t - \overline{t})^2}$$

$$\hat{\beta}_0 = \overline{Y} - \hat{\beta}_1\overline{t}$$

where $t = (n+1)/2$ is the average of integers $1, 2, \ldots, n$.

Consider the following simulated random walk process:



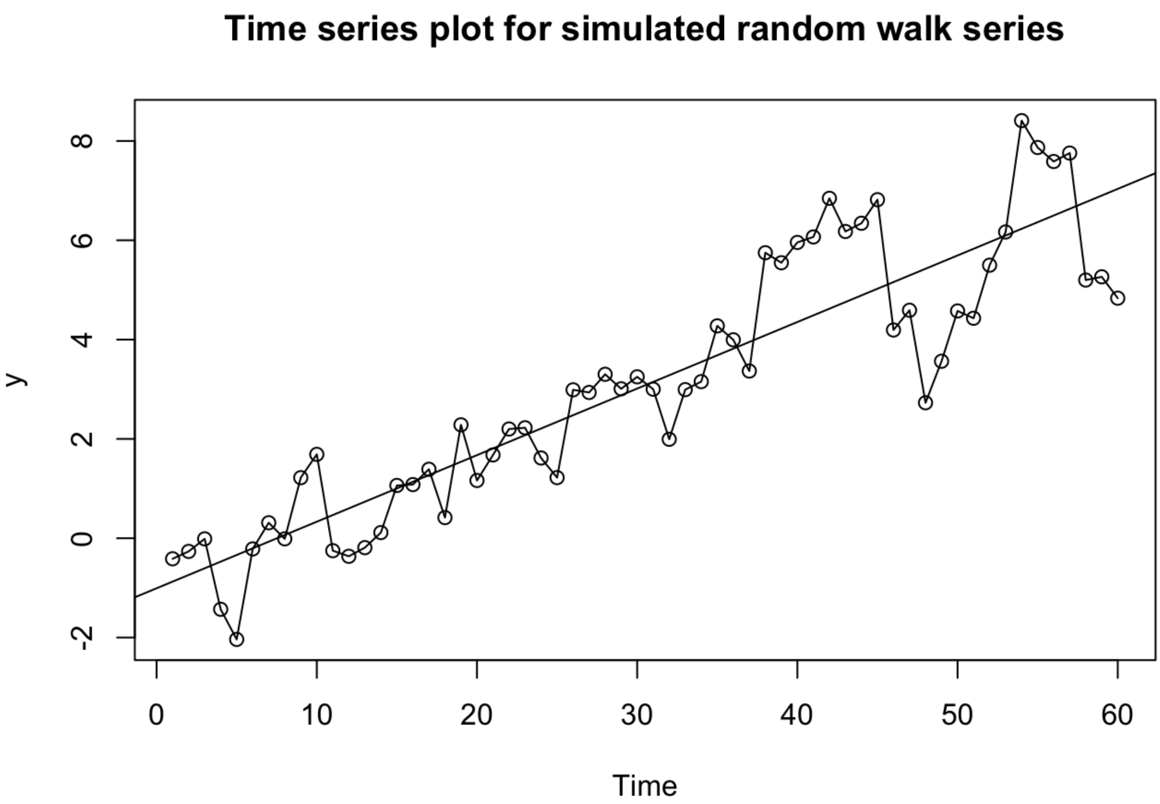
Suppose we (mistakenly) treat this as a linear time trend and estimate the slope and intercept by least-squares regression using the following code chunk:

```
data(rwalk)
model1 = lm(rwalk~time(rwalk)) # label the model as model1
summary(model1)
```

```
##
## Call:
## lm(formula = rwalk ~ time(rwalk))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.70045 -0.79782  0.06391  0.63064  2.22128
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.007888   0.297245  -3.391  0.00126 **
## time(rwalk)  0.134087   0.008475  15.822  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.137 on 58 degrees of freedom
## Multiple R-squared:  0.8119, Adjusted R-squared:  0.8086
## F-statistic: 250.3 on 1 and 58 DF,  p-value: < 2.2e-16
```

Estimates of slope and intercept are $\hat{\beta}_1 = 0.1341$ and $\hat{\beta}_0 = -1.008$, respectively. Here slope is statistically significant at 5% significance level. The trend line is plotted over the time series in the following plot:

```
plot(rwalk,type='o',ylab='y',, main = "Time series plot for simulated r
  andom walk series")
abline(model1) # add the fitted least squares line from model1
```



Appropriateness of this linear trend model will be considered later.

The deterministic quadratic trend model is expressed as follows

$$\mu_t = \beta_0 + \beta_1 t + \beta_2 t^2$$

where β_0 represents intercept, β_1 corresponds to the linear trend, and β_2 corresponds to quadratic trend in time.

The following code chunk fits a quadratic trend model to the random walk data:

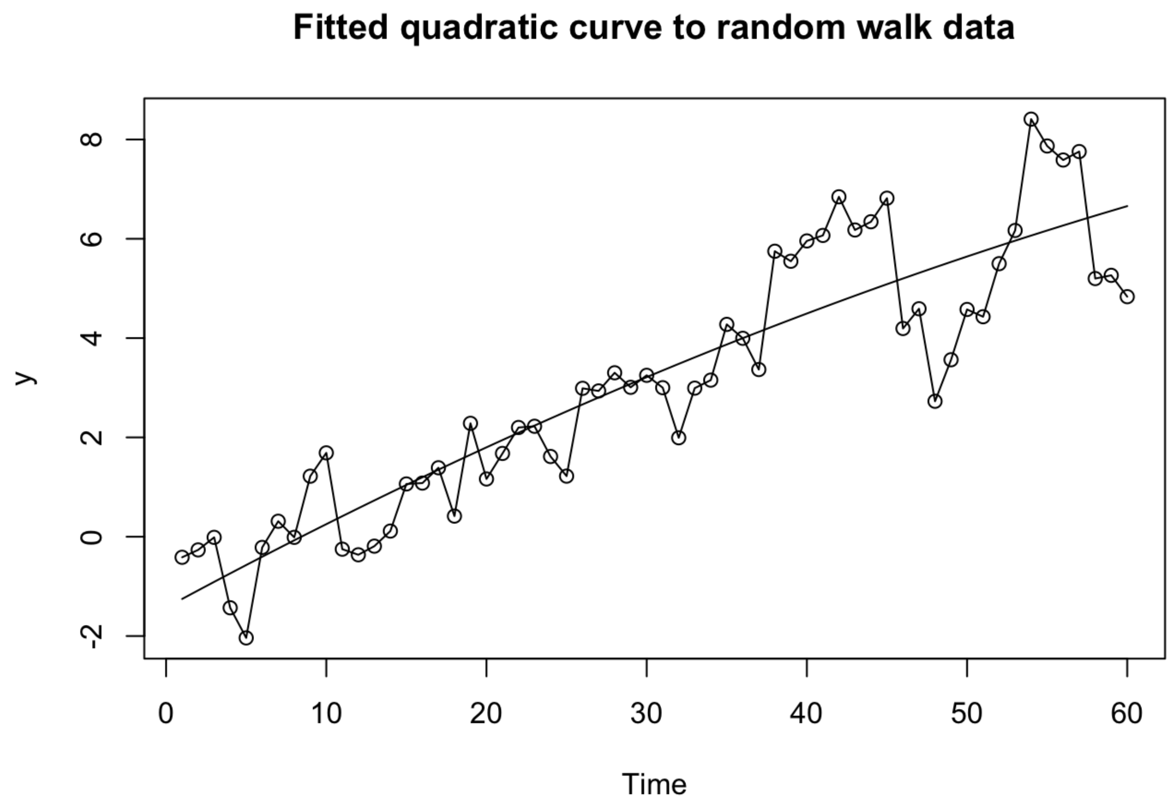
```
t = time(rwalk)
t2 = t^2
model1.1 = lm(rwalk~t+t2) # label the model as model1
summary(model1.1)
```

```
##
## Call:
## lm(formula = rwalk ~ t + t2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.69623 -0.76802  0.00826  0.85337  2.34468
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.4272911  0.4534893  -3.147  0.00262 **
## t            0.1746746  0.0343028   5.092 4.16e-06 ***
## t2          -0.0006654  0.0005451  -1.221  0.22721
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.132 on 57 degrees of freedom
## Multiple R-squared:  0.8167, Adjusted R-squared:  0.8102
## F-statistic:   127 on 2 and 57 DF,  p-value: < 2.2e-16
```

According to the p-values, quadratic trend term is found insignificant and the value of multiple R-squared is nearly the same as the linear trend model.

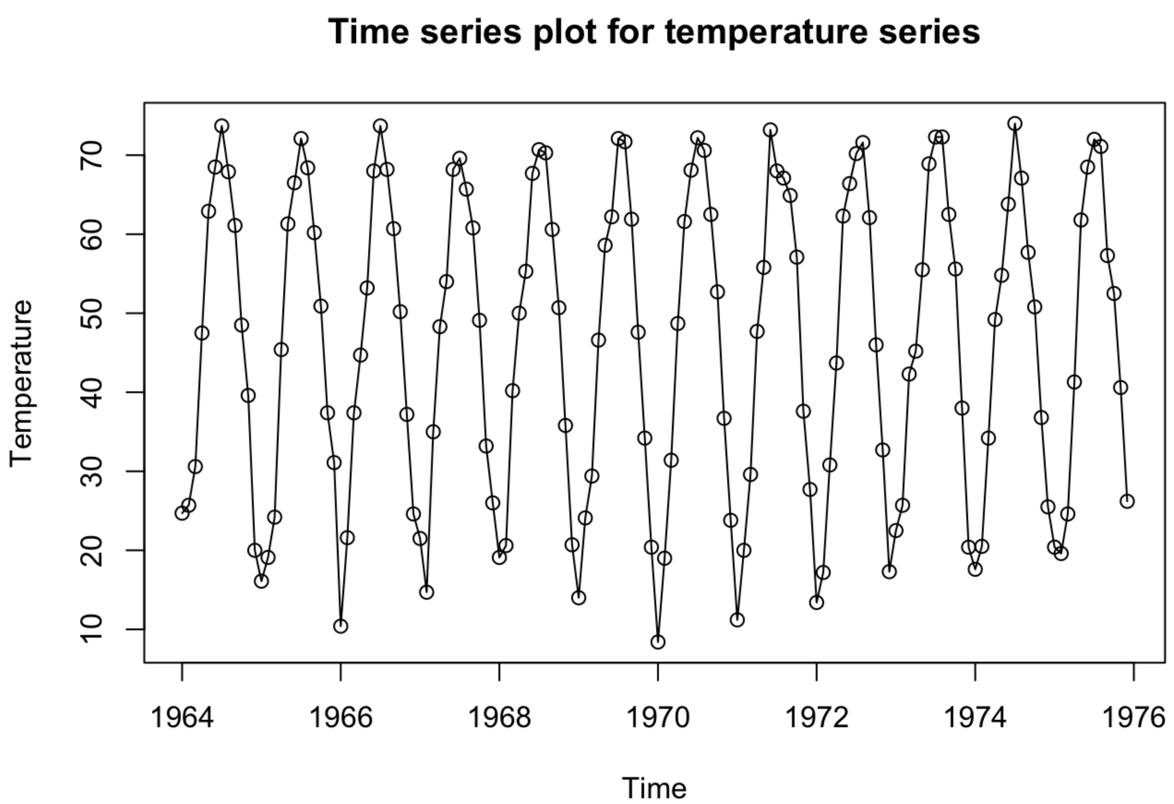
Fitted quadratic trend is shown below:

```
plot(ts(fitted(model1.1)), ylim = c(min(c(fitted(model1.1),
     as.vector(rwalk))), max(c(fitted(model1.1),as.vector(rwalk)))),yla
  b='y' ,
     main = "Fitted quadratic curve to random walk data")
lines(as.vector(rwalk),type="o")
```



Cyclical or Seasonal Trends

Consider now modeling and estimating seasonal trends, such as for the average monthly temperature data.



Here we assume that the observed series can be represented as

$$Y_t = \mu_t + X_t$$

where $E(X_t) = 0$ for all t. The most general assumption for μ_t with monthly seasonal data is that there are 12 parameters, $\beta_1, \beta_2, \ldots, \beta_{12}$, giving the expected average temperature for each of the 12 months. To represent seasonality, we may write a **seasonal model** such that

$$\mu_t = \begin{cases} \beta_1 & \text{for } t = 1, 13, 25, \ldots \\ \beta_2 & \text{for } t = 2, 14, 26, \ldots \\ \vdots & \\ \beta_{12} & \text{for } t = 12, 24, 36, \ldots \end{cases}$$

We need to set up indicator variables (sometimes called dummy variables) that indicate the month to which each of the data points pertains before going on with estimation of parameters. We can also include an intercept term β_0 in the model.

```
data(tempdub)
month.=season(tempdub) # period added to improve table display and this
   line sets up indicators
model2=lm(tempdub~month.-1) # -1 removes the intercept term
summary(model2)
```

```
##
## Call:
## lm(formula = tempdub ~ month. - 1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.2750 -2.2479  0.1125  1.8896  9.8250
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## month.January      16.608      0.987   16.83   <2e-16 ***
## month.February     20.650      0.987   20.92   <2e-16 ***
## month.March        32.475      0.987   32.90   <2e-16 ***
## month.April        46.525      0.987   47.14   <2e-16 ***
## month.May          58.092      0.987   58.86   <2e-16 ***
## month.June         67.500      0.987   68.39   <2e-16 ***
## month.July         71.717      0.987   72.66   <2e-16 ***
## month.August       69.333      0.987   70.25   <2e-16 ***
## month.September    61.025      0.987   61.83   <2e-16 ***
## month.October      50.975      0.987   51.65   <2e-16 ***
## month.November     36.650      0.987   37.13   <2e-16 ***
## month.December     23.642      0.987   23.95   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.419 on 132 degrees of freedom
## Multiple R-squared:  0.9957, Adjusted R-squared:  0.9953
## F-statistic:  2569 on 12 and 132 DF,  p-value: < 2.2e-16
```

All of the parameters corresponding to months are statistically significant at 5% level. We can include the intercept parameter as follows:

```
model3=lm(tempdub~month.) # remove -1 to include the intercept term in
   the model
summary(model3)
```

```
##
## Call:
## lm(formula = tempdub ~ month.)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.2750 -2.2479  0.1125  1.8896  9.8250
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)       16.608      0.987  16.828  < 2e-16 ***
## month.February     4.042      1.396   2.896  0.00443 **
## month.March       15.867      1.396  11.368  < 2e-16 ***
## month.April       29.917      1.396  21.434  < 2e-16 ***
## month.May         41.483      1.396  29.721  < 2e-16 ***
## month.June        50.892      1.396  36.461  < 2e-16 ***
## month.July        55.108      1.396  39.482  < 2e-16 ***
## month.August      52.725      1.396  37.775  < 2e-16 ***
## month.September   44.417      1.396  31.822  < 2e-16 ***
## month.October     34.367      1.396  24.622  < 2e-16 ***
## month.November    20.042      1.396  14.359  < 2e-16 ***
## month.December     7.033      1.396   5.039 1.51e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.419 on 132 degrees of freedom
## Multiple R-squared:  0.9712, Adjusted R-squared:  0.9688
## F-statistic: 405.1 on 11 and 132 DF,  p-value: < 2.2e-16
```

`R` omits the January coefficient in this case. Notice that when we remove intercept, we interpret resulting parameters as the difference between the first month and the related one. Now the February coefficient is interpreted as the difference between February and January average temperatures, the March coefficient is the difference between March and January average temperatures, and so forth. In this model, all of the differences between January and the other months are statistically significant at 5% level in both models. Notice that the Intercept coefficient plus the February coefficient here equals the February coefficient the model with no intercept parameter.

Cosine Trends

In the seasonal means model, we separate the effect of each month. However, there is nothing about the shape of the seasonal trend in the seasonal means model. We can include the information on the shape of the seasonal trend in the model by assigning a cosine curve as the mean function μ_t:

$$\mu_t = \beta\cos(2\pi f t + \Phi)$$

Here, $\beta(> 0), f$, and Φ are called the amplitude, frequency, and phase of the curve. As t varies, the curve oscillates within $[-\beta, \beta]$ interval. Since the curve repeats itself exactly every $1/f$ time units, $1/f$ is called the period of the cosine wave. When we set $f = 1/12$, a cosine wave will repeat itself every 12 months. So we say that the period is 12.

For the estimation purposes, we need to make the above cosine trend model linear in terms of its parameters. With the following misinterpretation, we get

$$\beta\cos(2\pi f t + \Phi) = \beta_1 \cos(2\pi f t) + \beta_2 \sin(2\pi f t)$$

where

$$\beta = \sqrt{\beta_1^2 + \beta_2^2} \text{ and } \Phi = \text{atan}(-\beta_2/\beta_1)$$

and, conversely,

$$\beta_1 = \beta\cos(\Phi) \text{ and } \beta_2 = \beta\sin(\Phi).$$

Consequently, we will use $cos(2\pi f t)$ and $sin(2\pi f t)$ to estimate β_1 and β_2, respectively. The simplest such model for the trend would be expressed as

$$\mu_t = \beta_0 + \beta_1 \cos(2\pi f t) + \beta_2 \sin(2\pi f t)$$

Here the constant term β_0 represents a cosine with frequency zero.

In any practical example, we must be careful how we measure time, as our choice of time measurement will affect the values of the frequencies of interest. For example, if we have monthly data but use $1, 2, 3, \ldots$ as our time scale, then 1/12 would be the most interesting frequency, with a corresponding period of 12 months. However, if we measure time by year and fractional year, say 1980 for January, 1980.08333 for February of 1980, and so forth, then a frequency of 1 corresponds to an annual or 12-month periodicity.

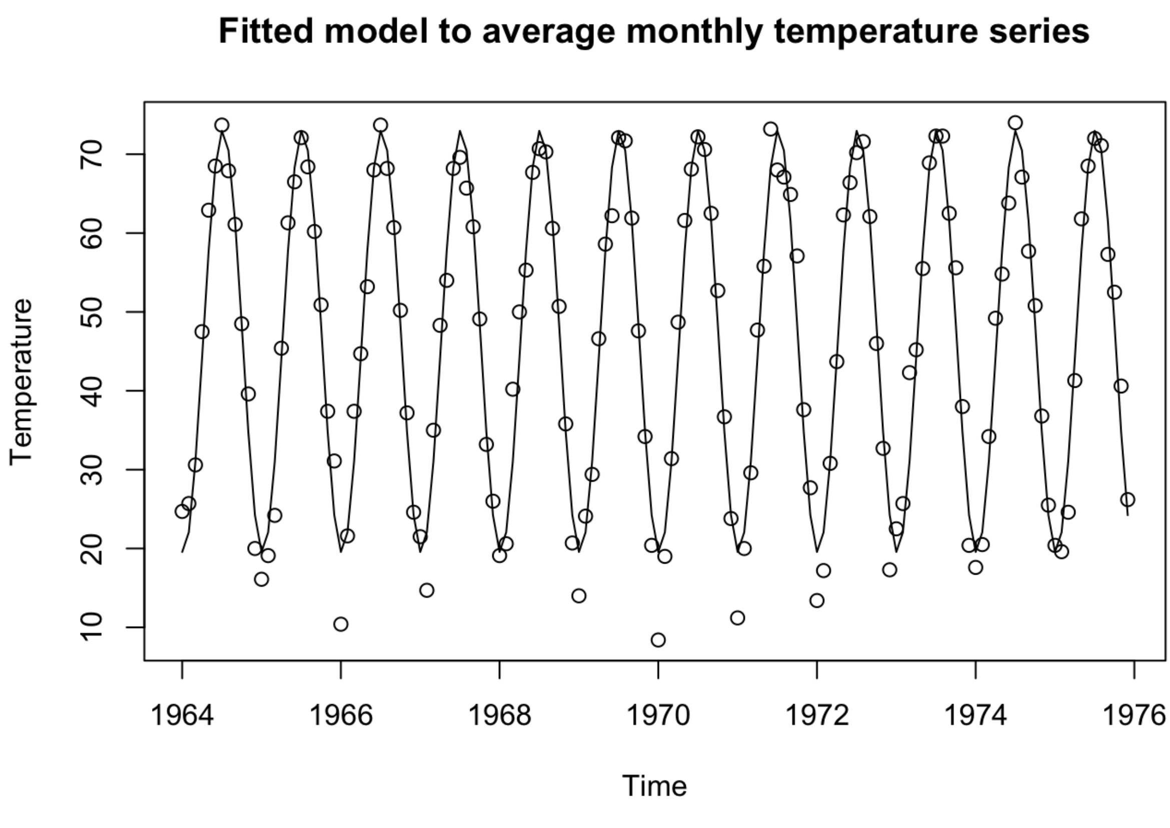
The following code chunk fits a cosine curve at the fundamental frequency to the average monthly temperature series.

```
har.=harmonic(tempdub,1) # calculate cos(2*pi*t) and sin(2*pi*t)
model4=lm(tempdub~har.)
summary(model4)
```

```
##
## Call:
## lm(formula = tempdub ~ har.)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.1580  -2.2756  -0.1457   2.3754  11.2671
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      46.2660     0.3088 149.816  < 2e-16 ***
## har.cos(2*pi*t) -26.7079     0.4367 -61.154  < 2e-16 ***
## har.sin(2*pi*t)  -2.1697     0.4367  -4.968 1.93e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.706 on 141 degrees of freedom
## Multiple R-squared:  0.9639, Adjusted R-squared:  0.9634
## F-statistic:  1882 on 2 and 141 DF,  p-value: < 2.2e-16
```

The following code chunk plots the fitted curve along with the observed average monthly temperature series.

```
plot(ts(fitted(model4),freq=12,start=c(1964,1)),ylab='Temperature',type
  ='l',
ylim=range(c(fitted(model4),tempdub)),main="Fitted model to average mon
  thly temperature series") # ylim ensures that the y axis range fits t
  he raw data and the fitted values
points(tempdub)
```



The cosine trend model fits the data quite well with the exception of most of the January values, where the observations are lower than the model would predict.

Interpreting Regression Output

Estimates of regression parameters are obtained under some assumptions on the stochastic component $\{X_t\}$ of linear trend model. So, some properties of regression output heavily depend on the assumption that X_t is white noise and some other parts depend on approximate normality of X_t.

When we have $\mu_t = \beta_0 + \beta_1 t$ as the mean function, the unobserved stochastic component X_t can be estimated (predicted) by $Y_t - \hat{\mu}_t$. If X_t has a constant variance, we estimate the standard deviation of X_t, namely $\sqrt{\gamma_0}$, by the **residual standard deviation**

$$s = \sqrt{\frac{1}{n-p} \sum_{t=1}^{n} (Y_t - \hat{\mu}_t)^2}$$

where p is the number of parameters estimated in μ_t and $n - p$ is the so-called *degrees of freedom* for s.

> The smaller the value of s, the better the fit.

Another measure of goodness of fit of the trend is the *coefficient of determination*, namely R^2. One interpretation of R^2 is that it is the square of the sample correlation coefficient between the observed series and the estimated trend. It is also the fraction of the variation in the series that is explained by the estimated trend.

> High but not close to 1 values of R^2 implies a satisfactory fit.

When we fit the straight line to the random walk data, we get the following output:

```
model1=lm(rwalk~time(rwalk))
summary(model1)
```

```
##
## Call:
## lm(formula = rwalk ~ time(rwalk))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.70045 -0.79782  0.06391  0.63064  2.22128
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.007888   0.297245  -3.391  0.00126 **
## time(rwalk)  0.134087   0.008475  15.822  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.137 on 58 degrees of freedom
## Multiple R-squared:  0.8119, Adjusted R-squared:  0.8086
## F-statistic: 250.3 on 1 and 58 DF,  p-value: < 2.2e-16
```

According to multiple R^2, about 81% of the variation in the random walk series is explained by the linear time trend. The adjusted version of multiple R^2 provides an approximately unbiased estimate of true R^2.

The standard deviations of the coefficients labeled Std. Error on the output need to be interpreted carefully. They are appropriate only when the usual regression assumption that the stochastic component is white noise. This assumption rarely true for time series data!

If the stochastic component is normally distributed white noise, then the p-values are given under "Pr(>|t|)" can be used to test the null hypothesis that the corresponding unknown regression coefficient is zero.

Residual Analysis

The *estimator* or *predictor* of unobserved stochastic component $\{X_t\}$,

$$\hat{X}_t = Y_t - \hat{\mu}_t$$

is called **residual** corresponding to the tth observation.

> An estimate is the guess of an unknown parameter and a prediction is an estimate of an unobserved random variable.

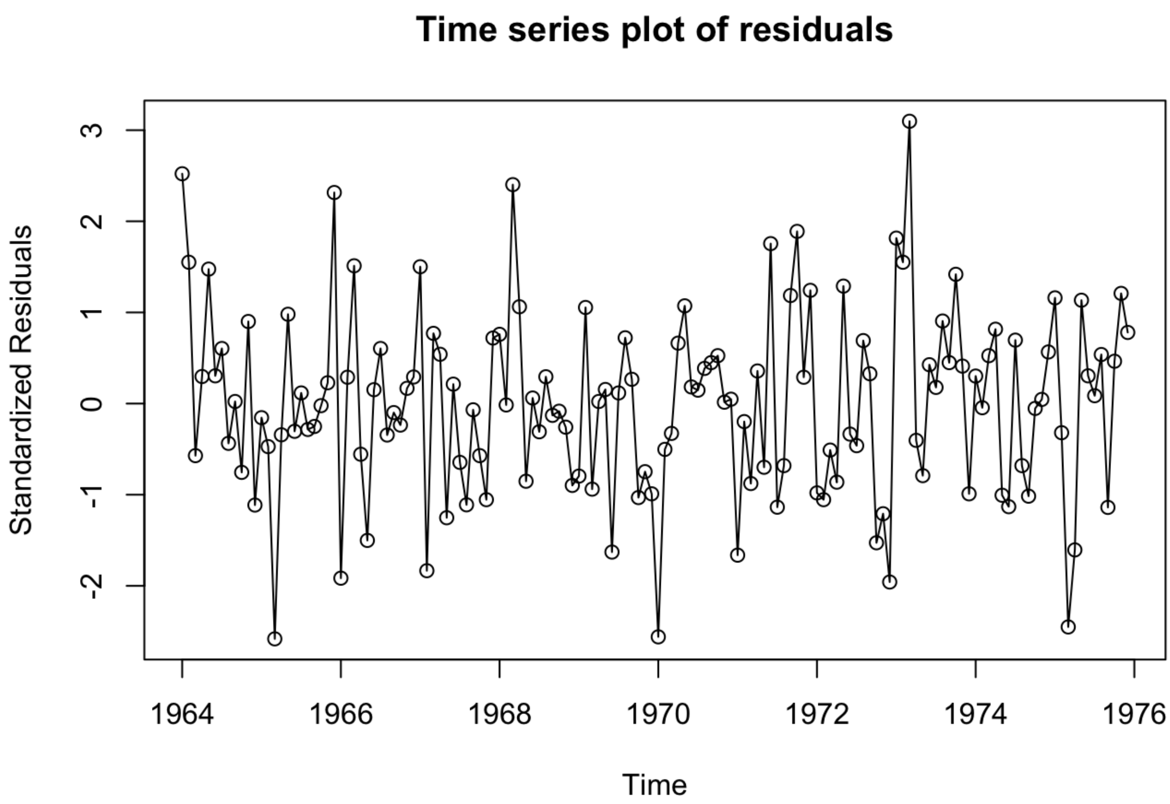
If the trend model is reasonably correct, then the residuals should behave roughly like the true stochastic component, and various assumptions about the stochastic component can be assessed by looking at the residuals.

If the stochastic component is white noise, then the residuals should behave roughly like independent (normal) random variables with zero mean and standard deviation of s. We can standardise residuals to make their mean zero.

After computation of residuals or standardised residual, we examine various residual plots. The first plot to examine is the plot of the residuals over time. If the series is seasonal, we can use labels while plotting to identify the seasonality better.

In the first example, We will use the monthly average temperature series which we fitted with seasonal means as our first example to illustrate some of the ideas of residual analysis. The following chunk generates a time series plot for the *standardized residuals* of the monthly temperature data fitted by seasonal means:

```
plot(y=rstudent(model3),x=as.vector(time(tempdub)), xlab='Time',ylab='S
  tandardized Residuals',type='o', main = "Time series plot of residual
  s")
```

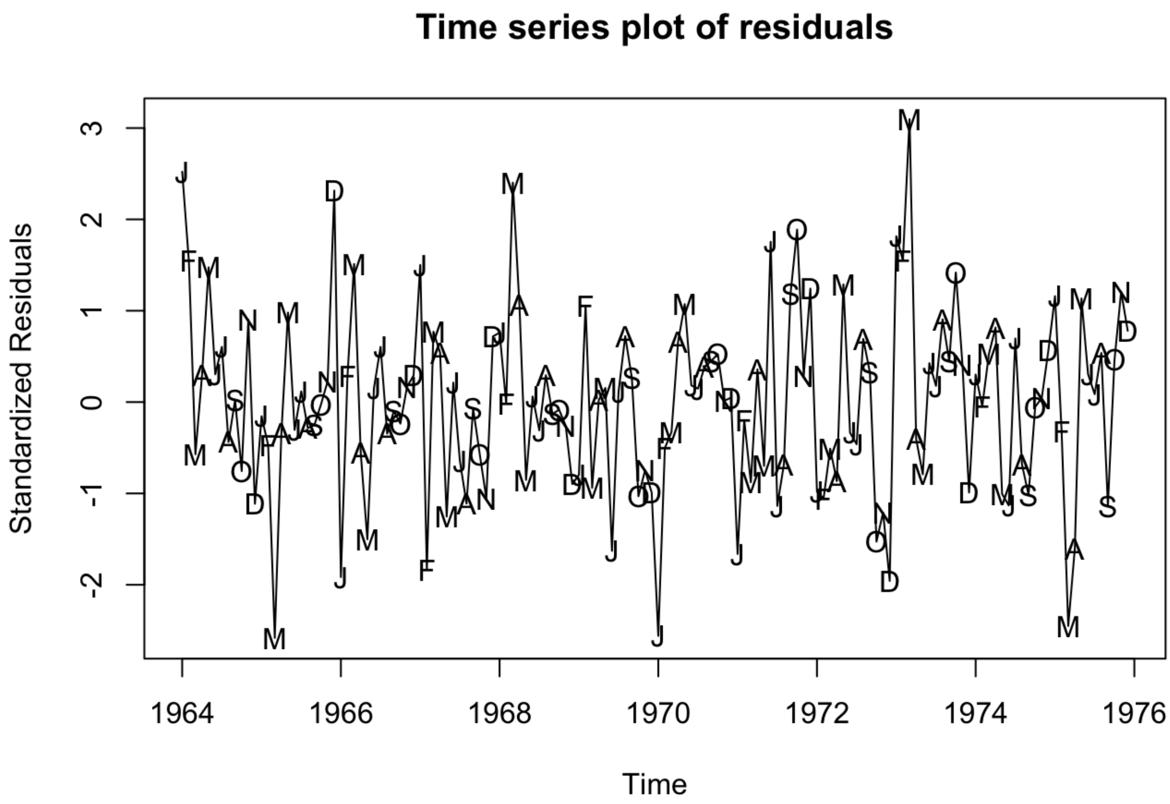


> If the stochastic component is white noise and the trend is adequately modeled, we would expect such a plot to suggest a rectangular scatter with no discernible trends whatsoever.

There are striking departures from randomness seen in the plot.

The labels of months are added in the next plot.

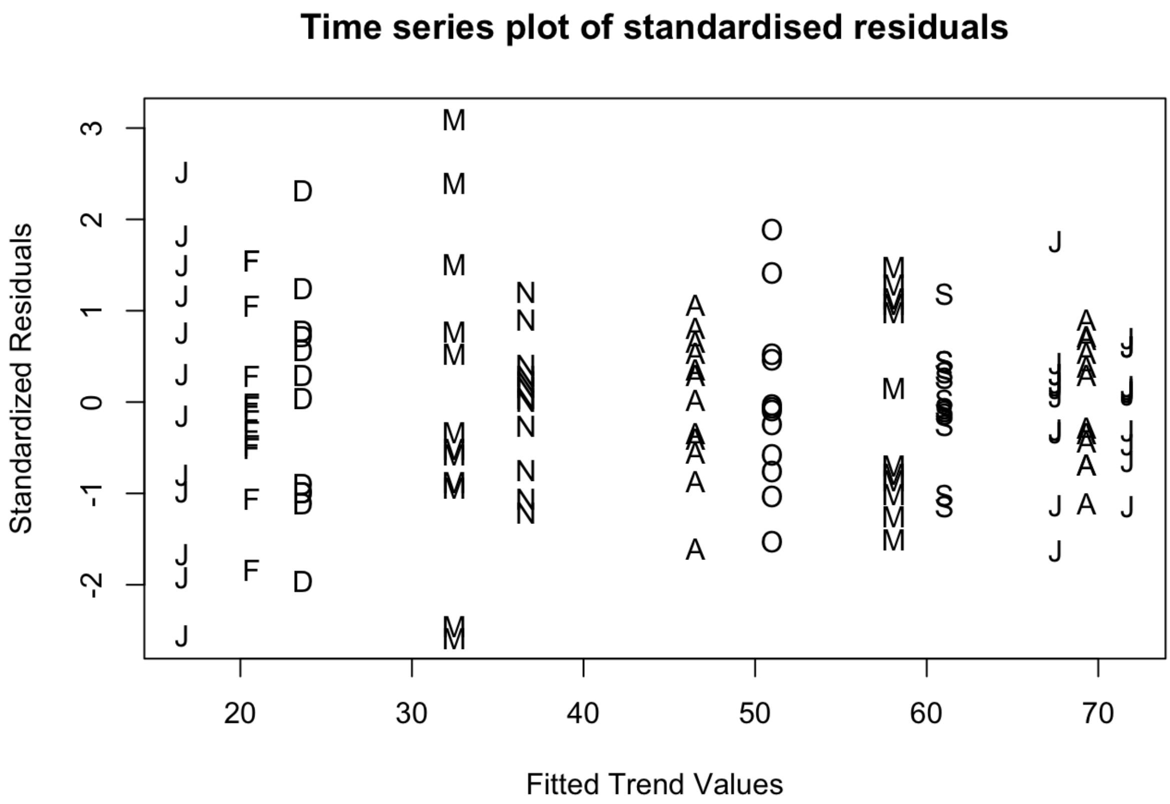
```
plot(y=rstudent(model3),x=as.vector(time(tempdub)),xlab='Time', ylab='S
  tandardized Residuals',type='l', main = "Time series plot of residual
  s")
points(y=rstudent(model3),x=as.vector(time(tempdub)), pch=as.vector(sea
  son(tempdub)))
```



There is no apparent pattern relating to different months of the year.

Next, we look at the standardized residuals versus the corresponding trend estimate, or fitted value running the following code chunk. the function `rstudent()` computes standardised residuals.

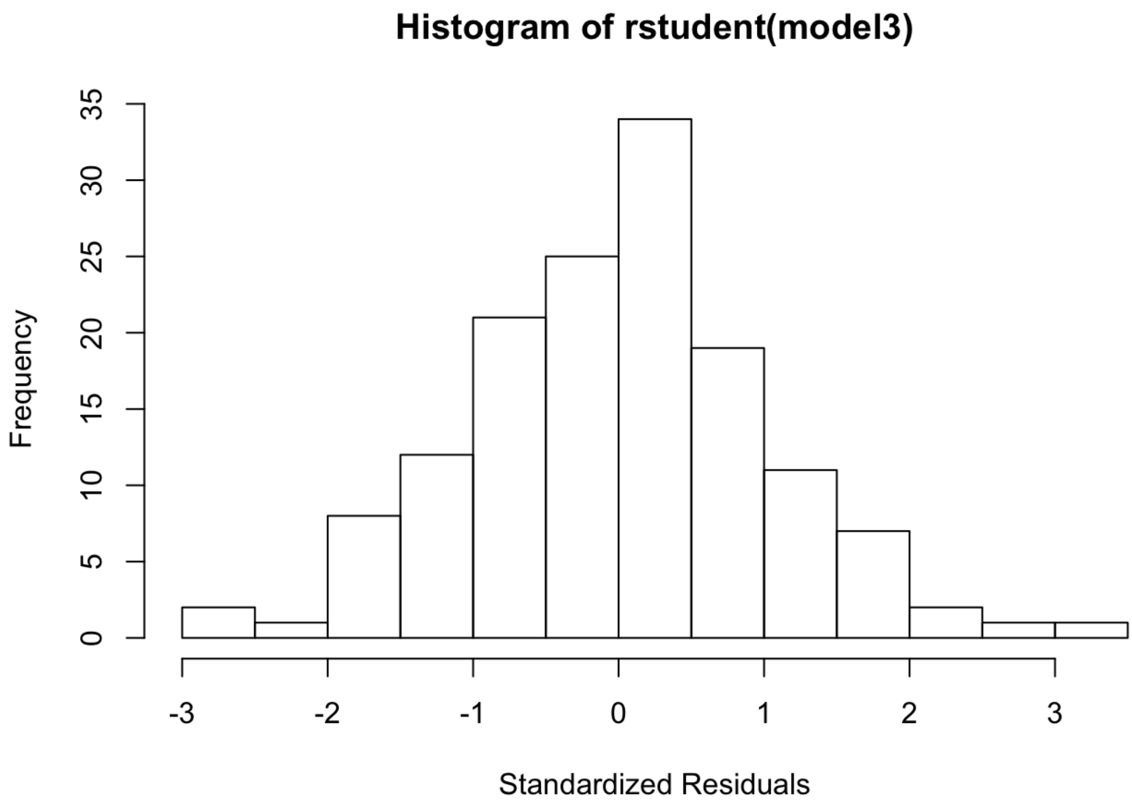
```
plot(y=rstudent(model3),x=as.vector(fitted(model3)), xlab='Fitted Trend
   Values', ylab='Standardized Residuals',type='n', main = "Time series
   plot of standardised residuals")
points(y=rstudent(model3),x=as.vector(fitted(model3)),pch=as.vector(sea
  son(tempdub)))
```



As anomaly with this plot small residuals would be associated with small fitted trend values and large residuals with large fitted trend values, or there would be less variation for residuals associated with certain sized fitted trend values or more variation with other fitted trend values. Although there is somewhat more variation for the March residuals and less for November, the plot does not indicate any dramatic patterns that would cause us to doubt the seasonal means model.

Normality of residuals can be checked with a histogram. The following displays a frequency histogram of the standardized residuals from the seasonal means model for the temperature series.

```
hist(rstudent(model3),xlab='Standardized Residuals')
```



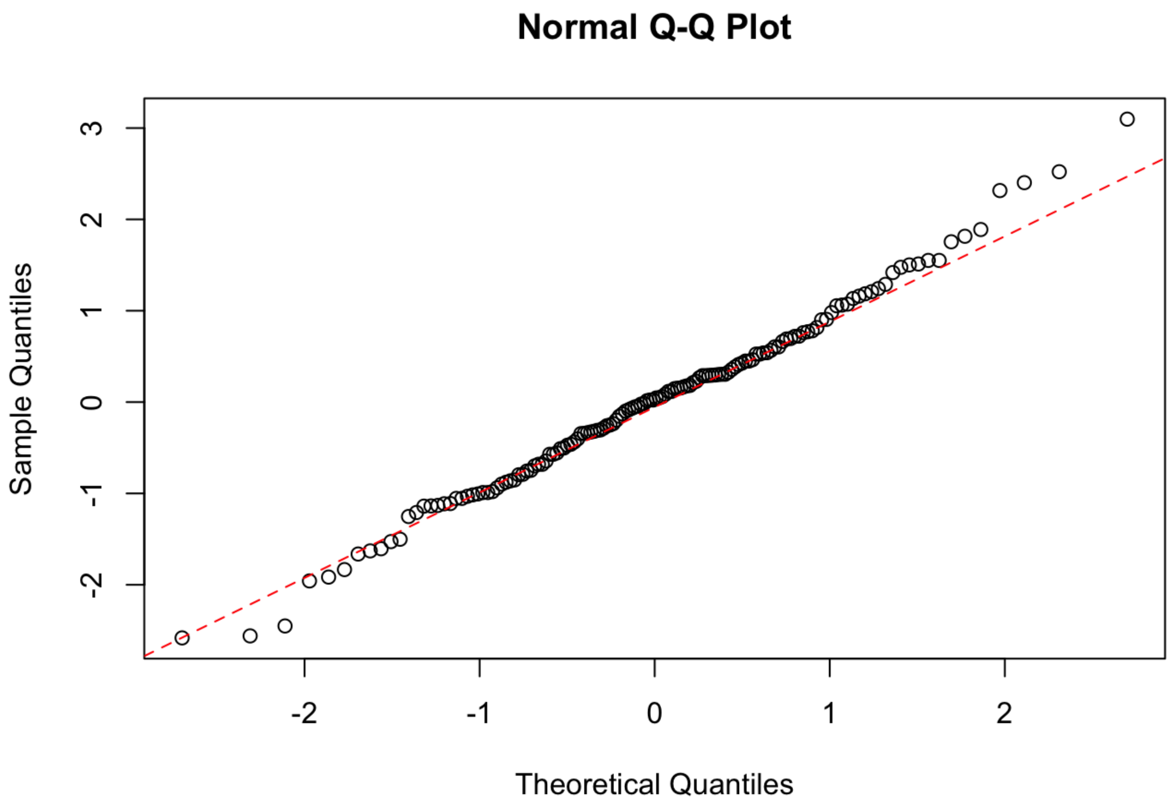
The plot is somewhat symmetric and tails off at both the high and low ends as a normal distribution does.

Another plot to check normality is the quantile-quantile (QQ) plot. Such a plot displays the quantiles of the data versus the theoretical quantiles of a normal distribution.

> With normally distributed data, the QQ plot looks approximately like a straight line.

The following plot shows the QQ normal scores plot for the standardized residuals from the seasonal means model for the temperature series.

```
y = rstudent(model3)
qqnorm(y)
qqline(y, col = 2, lwd = 1, lty = 2)
```



The straight-line pattern here supports the assumption of a normally distributed stochastic component in this model.

In addition to visualisations, there are various hypothesis tests that can be used to check the normality assumption of the stochastic component. One of these tests is the Shapiro-Wilk test that calculates the correlation between the residuals and the corresponding normal quantiles. We apply the Shapiro-Wilk test to the residuals of temperature series using the following code chunk

```
y = rstudent(model3)
shapiro.test(y)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  y
## W = 0.9929, p-value = 0.6954
```

We get the p-value of 0.6954. So we conclude not to reject the null hypothesis that the stochastic component of this model is normally distributed.

Independence in the stochastic component is another assumption to check. The runs test can be applied over the residuals. The runs test applied over the residuals of temperature series leads to a p-value of 0.216. Thus, we conclude not to reject the null hypothesis stating the independence of the stochastic component in this seasonal means model.

Sample Autocorrelation Function

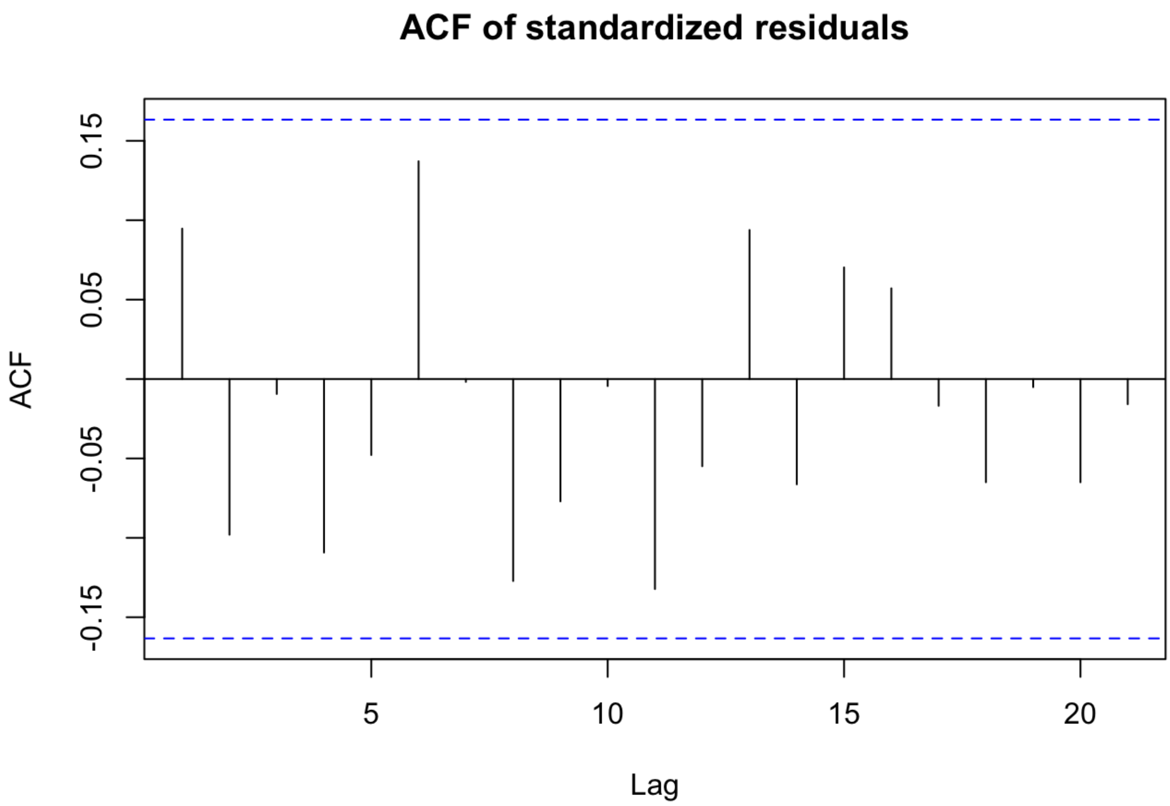
Sample autocorrelation function (ACF) is a very useful and important tool in the analysis of time series data. We compute the sample correlation between the pairs k units apart in time. However, we modify this slightly, taking into account that we are assuming stationarity, which implies a common mean and variance for the series. With this in mind, we define the sample autocorrelation function, r_k, at lag k as

$$r_k = \frac{\sum_{t=k+1}^{n} (Y_t - \bar{Y})(Y_{t-k} - \bar{Y})}{\sum_{t=1}^{n} (Y_t - \bar{Y})^2}$$

for $k = 1, 2, \ldots$. A plot of r_k versus lag k is often called a **correlogram**.

Because we are interested in discovering possible dependence in the stochastic component, the sample autocorrelation function for the standardized residuals is of interest. The following displays the sample autocorrelation for the standardized residuals from the seasonal means model of the temperature series.

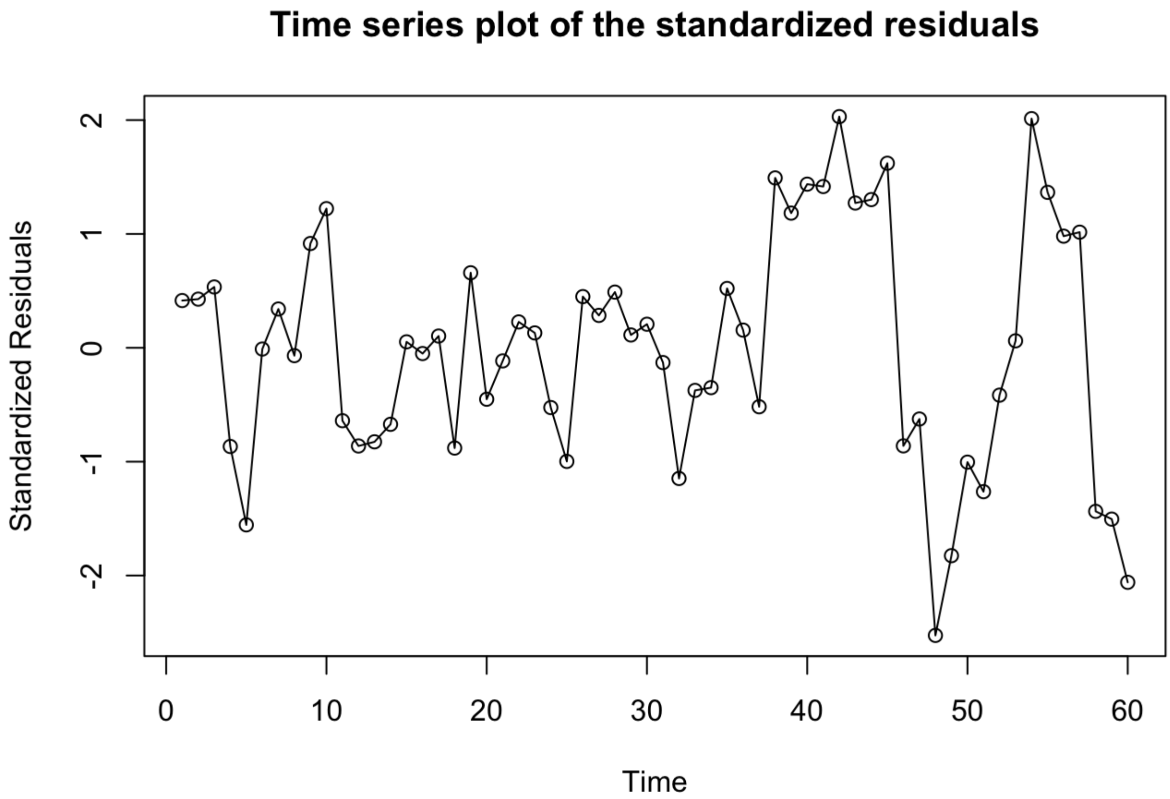
```
acf(rstudent(model3), main = "ACF of standardized residuals")
```



All values are within the horizontal dashed lines, which are placed at $\pm 2/\sqrt{n}$. According to the ACF plot none of the hypotheses $\rho_k = 0$ can be rejected at the usual significance levels for $k = 1, 2, \ldots, 21$. Thus, we infer that the stochastic component of the series is white noise.

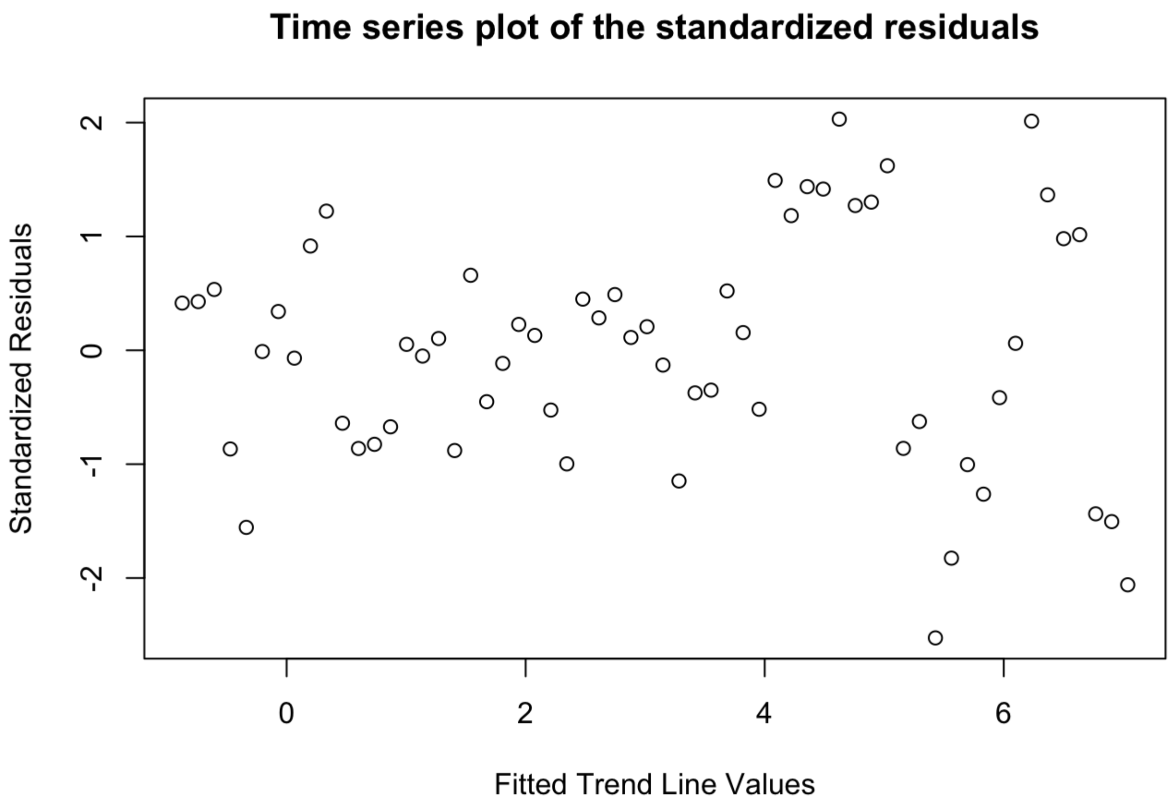
As a second example, a time series plot of the standardized residuals arising from fitting a straight line to the random walk time series is shown below:

```
plot(y=rstudent(model1),x=as.vector(time(rwalk)), ylab='Standardized Re
  siduals',xlab='Time',type='o', main = "Time series plot of the standa
  rdized residuals")
```



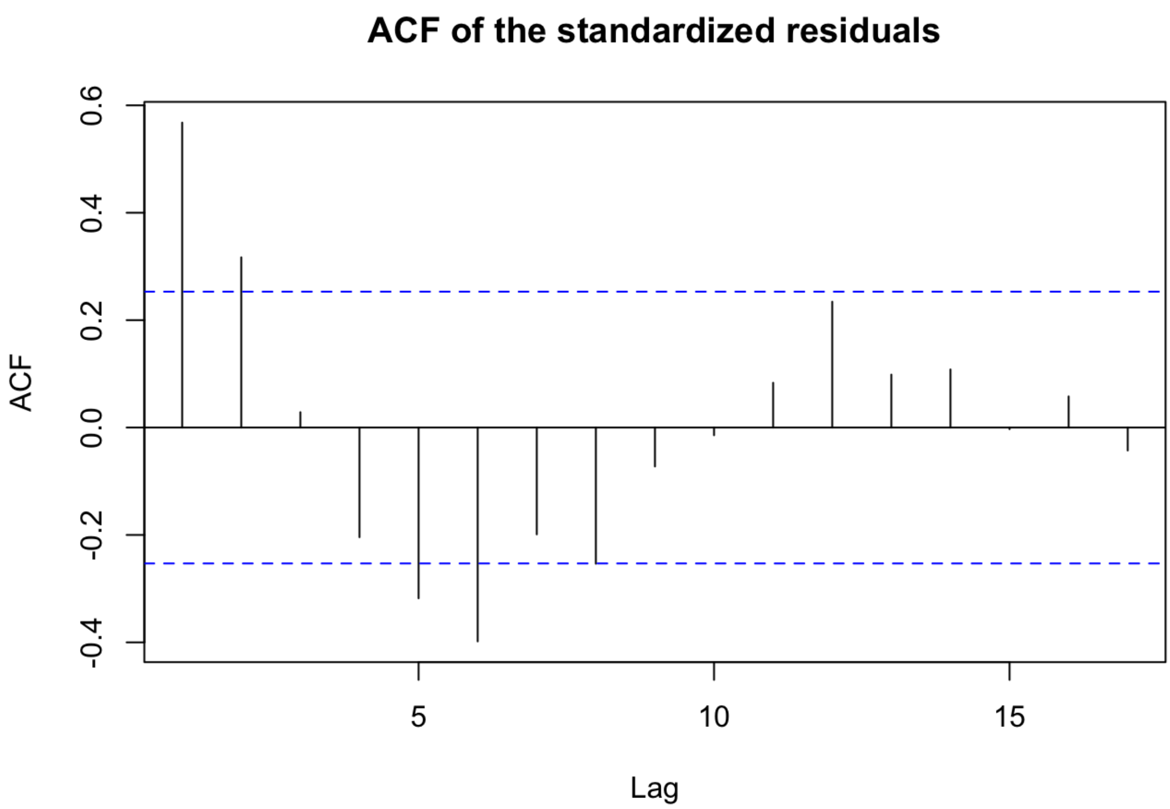
In this plot, the residuals “hang together” too much for the white noise-the plot is too smooth. Furthermore, there seems to be more variation in the last third of the series than in the first two-thirds. When we plot standardised residuals versus fitted trend line values, we observe a similar effect with larger residuals associated with larger fitted values.

```
plot(y=rstudent(model1),x=fitted(model1), ylab='Standardized Residuals'
  ,xlab='Fitted Trend Line Values', type='p', main = "Time series plot
   of the standardized residuals")
```



The sample ACF of the standardized residuals is given below:

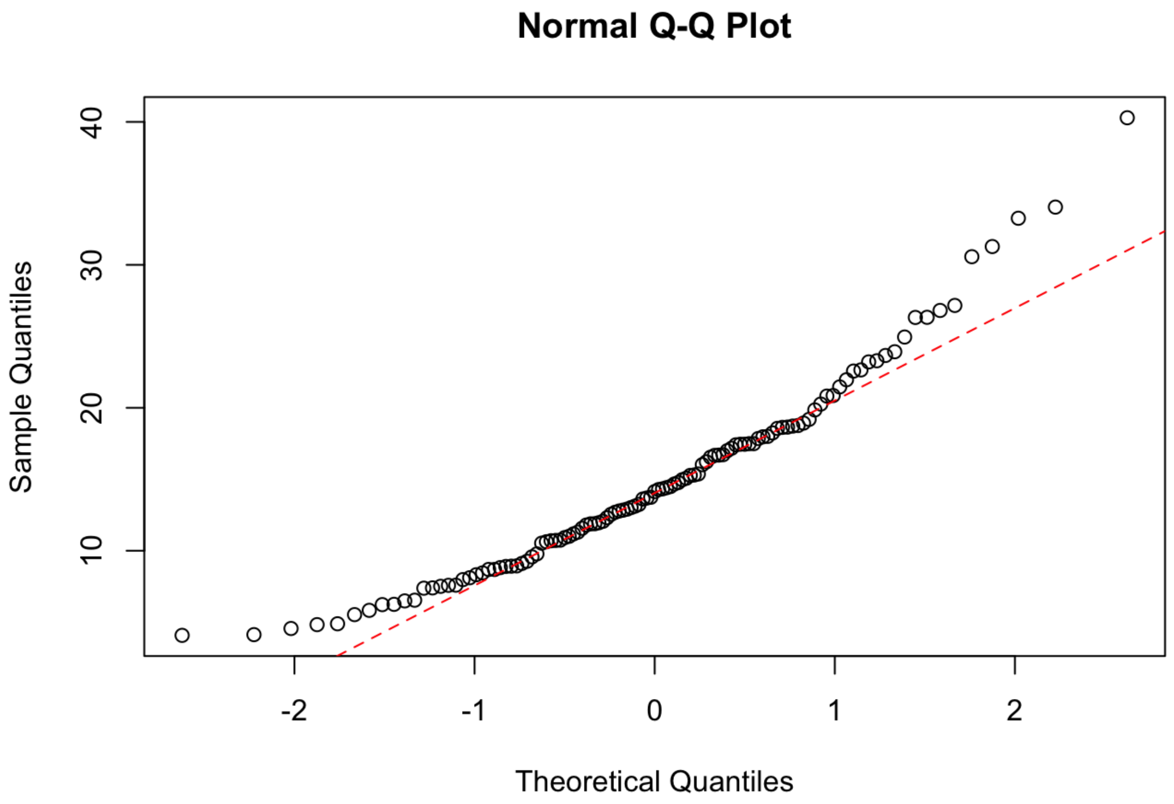
```
acf(rstudent(model1), main = "ACF of the standardized residuals")
```



This ACF plot confirms the smoothness of the time series plot as we have correlation values higher than the confidence bound at several lags. This is not what we expect from a white noise process.

As another example, we return to the annual rainfall in Los Angeles for which we found no evidence of dependence in that series and check the normality assumption using the following QQ plot.

```
data(larain)
y = larain
qqnorm(y)
qqline(y, col = 2, lwd = 1, lty = 2)
```



Because we see a considerable amount of departure from the reference line, we conclude that the normality assumption does not hold for the annual rainfall series in Los Angeles. The Shapiro-Wilk test also confirms this inference with a p-value less than 0.05.

```
y = larain
shapiro.test(y)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  y
## W = 0.94617, p-value = 0.0001614
```

Forecasting with regression models

After ensuring that the fitted model is suitable for prediction purposes, we use the model to find forecasts. For time series regression models, this task is simply based on the straightforward use of the fitted regression model. We apply the following steps to find h steps ahead forecasts:

1. Generate a sequence of time points of lengths h starting from the last observation point. For example, suppose we have a time series of length 10 and $h = 4$. Then the new sequence becomes $t = 11, 12, 13, 14$.
2. Write each value of the new sequence generated in the previous step in place in the fitted model and calculate forecasts.

We can implement these steps using the `predict()` function with fitted model object and the sequence created at step 1 as inputs.

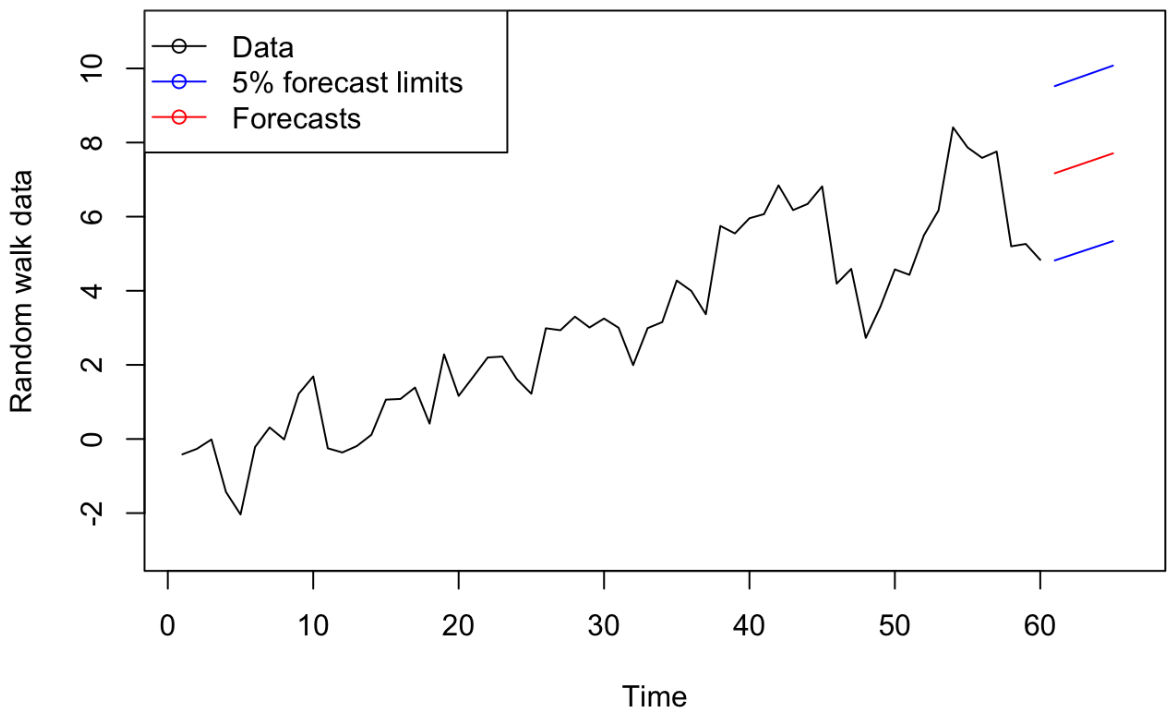
To illustrate, let's use the fitted linear model for the random walk data to find 5 steps ahead forecasts. The following code chunk does this task:

```
data(rwalk) # Read the data
t = time(rwalk) # Create time points for model fitting
model1 = lm(rwalk~t) # label the model as model1
h = 5 # 5 steps ahed forecasts
# Now we will implement the two-step algorithm
new = data.frame(t = seq((length(t)+1), (length(t)+h), 1)) # Step 1
# Notice here that I'm using the same variable name "t" as in the
# fitted model above, where the name of the variable showing time
# is also "t". To run the predict() function properly,
# the names of variables in fitted model and "new" data frame
# must be the same!!!
forecasts = predict(model1, new, interval = "prediction")
# Here interval argument shows the prediction interval
print(forecasts)
```

```
##        fit      lwr       upr
## 1 7.171430 4.819249  9.523611
## 2 7.305517 4.949546  9.661487
## 3 7.439604 5.079727  9.799480
## 4 7.573691 5.209794  9.937588
## 5 7.707778 5.339745 10.075811
```

We can plot these forecasts next to the time series of interest by the following code chunk:

```
plot(rwalk, xlim = c(1,66), ylim = c(-3, 11), ylab = "Random walk data"
  )
# We need to convert forecasts to time series object starting from the
   first
# time steps-ahead to be able to use plot function.
# We do this for all columns of forecasts
lines(ts(as.vector(forecasts[,1]), start = 61), col="red", type="l")
lines(ts(as.vector(forecasts[,2]), start = 61), col="blue", type="l")
lines(ts(as.vector(forecasts[,3]), start = 61), col="blue", type="l")
legend("topleft", lty=1, pch=1, col=c("black","blue","red"), text.width
   = 18,
       c("Data","5% forecast limits", "Forecasts"))
```



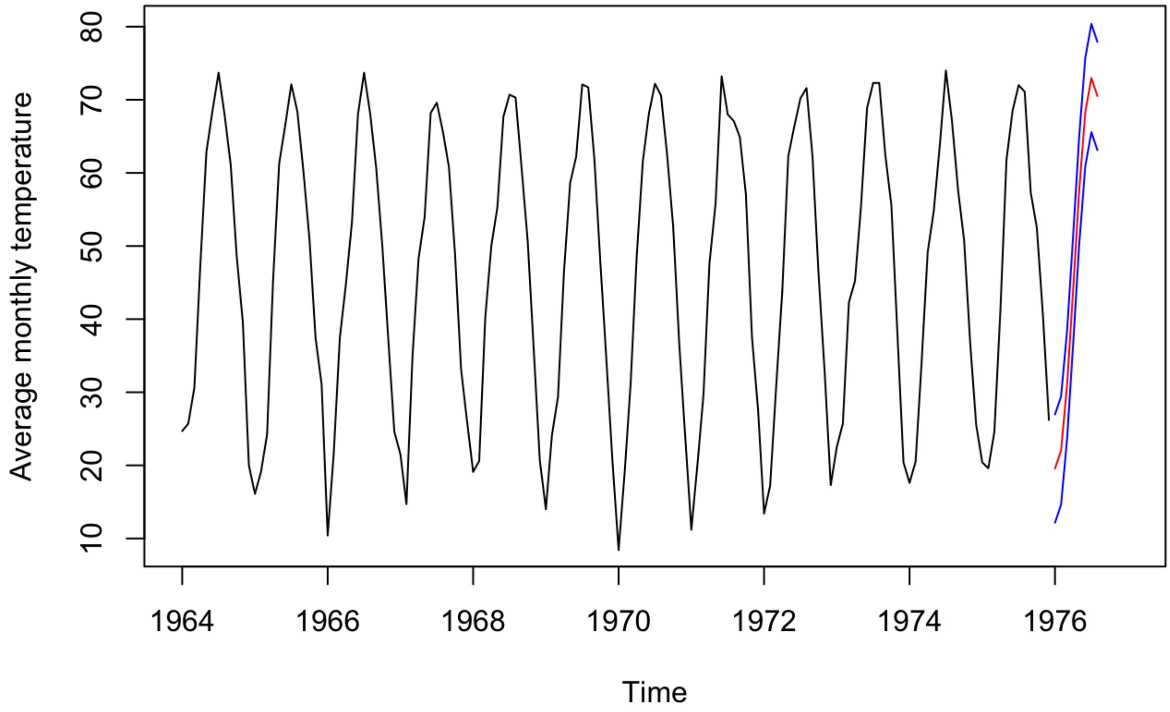
As another example, the harmonic model fitted to the average monthly temperature series and find forecasts for 7 months ahead.

```
har.=harmonic(tempdub,1) # calculate cos(2*pi*t) and sin(2*pi*t)
t1 = har.[,1] # To make it easier assign harmonic variables to separate
   variables
t2 = har.[,2]
model4=lm(tempdub~t1+t2) # Fit the model with separate variables
# We need to create continuous time for 7 months starting from the firs
  t month of 1976
t = c(1976.000, 1976.083, 1976.167 ,1976.250, 1976.333, 1976.417 ,1976.
  500, 1976.583 )
t1 = cos(2*pi*t)
t2 = sin(2*pi*t)
new = data.frame(t1 , t2) # Step 1
# Notice here that I'm using the same variable names "t1" and "t2" as i
  n the
# fitted model above, where the name of the variables showing sine and
   cosine
# components are also "t1" and "t2". To run the predict() function prop
  erly,
# the names of variables in fitted model and "new" data frame
# must be the same!!!
forecasts = predict(model4, new, interval = "prediction")
print(forecasts)
```

```
##        fit      lwr      upr
## 1 19.55804 12.15595 26.96012
## 2 22.02737 14.62528 29.42945
## 3 31.07915 23.67707 38.48124
## 4 44.09622 36.69414 51.49831
## 5 57.69014 50.28806 65.09223
## 6 68.34270 60.94062 75.74479
## 7 72.97391 65.57182 80.37599
## 8 70.50458 63.10249 77.90666
```

We plot the forecasts along with the original series with the following code chunk. The meaning of the colors is the same as the previous plot.

```
plot(tempdub, xlim = c(1964,1977), ylim = c(9, 80), ylab = "Average mon
  thly temperature")
# Here we convert the forecasts and prediction limits to monthly time s
  eries!
lines(ts(as.vector(forecasts[,1]), start = c(1976,1), frequency = 12),
  col="red", type="l")
lines(ts(as.vector(forecasts[,2]), start = c(1976,1), frequency = 12),
  col="blue", type="l")
lines(ts(as.vector(forecasts[,3]), start = c(1976,1), frequency = 12),
  col="blue", type="l")
```



Forecasts from the harmonic model successfully follow the repeating pattern in the original series.

Summary

In this module, we focused on describing, modeling, and estimating deterministic trends in time series. The simplest deterministic “trend” is a constant-mean function. Regression methods were then pursued to estimate trends that are linear or quadratic in time. Methods for modeling cyclical or seasonal trends came next, and the reliability and efficiency of all of these regression methods were investigated. Finally, we studied residual analysis to investigate the quality of the fitted model. We also introduced the important sample autocorrelation function, which is a very useful and important tool in the analysis of time series.