TSA-2018. Module 2. Time Series Regression

N. Balakrishna and T. V. Ramanathan

August 2018

Modeling Time Series with Trend

The observed time series may be generated by a regression type model, in which certain functions of time are taken as independent variables. For example, a series with trend is shown below:

$$X_t = m_t + E_t. \tag{1}$$

The function m_t may be linear or non-linear in time t.

For example, m_t may be a polynomial in t of the type:

$$m_t = \beta_0 + \beta_1 t + \beta_2 \frac{t^2}{2!} + \beta_3 \frac{t^3}{3!} + \ldots + \beta_p \frac{t^p}{p!} \tag{2}$$

Method of Least Squares

In this method we fit a parametric family of functions for m_t by minimizing the sum of squares due to error, i.e.,

$$\sum_{t=1}^{n} (X_t - m_t)^2.$$

The order p of the polynomial is a *model selection* problem. That is, we fit polynomials of degrees $1, 2, \ldots$ for m_t and choose the one for which the least squares method provides a close fit for the data.

Let the estimates of the coefficients be denoted by $\hat{\beta}_i, i = 0, 1, 2, \ldots, p$.

The estimate of the trend function is then

$$\hat{m}_t = \beta_0 + \hat{\beta}_1 t + \hat{\beta}_2 \frac{t^2}{2!} + \hat{\beta}_3 \frac{t^3}{3!} + \ldots + \hat{\beta}_p \frac{t^p}{p!}.$$

Now compute the residuals:

$$\hat{E}_t = X_t - \hat{m}_t, t = 1, 2, \ldots, n$$

and test for stationarity of $\{\hat{E}_t\}$.

Global Temperature Deviations

Global mean land-ocean temperature deviations (from 1951-1980 average), measured in degrees centigrade.

```r
library(astsa)
data(globtemp)
#help(globtemp)
globtemp=ts(globtemp) #save as a ts object
length(globtemp)
```

```
## [1] 136
```

Outline of the Approach:

- Select the first *nfit* observations as the Fitting/Training portion.
- Select the next *L=nfore* observations as the Holdout/Test portion.
- Fit MLR models to the Fitting portion of the data.
- Assess the fits. Compute in-sample model selection criteria.
- Use the fitted models to forecast/predict the response time series for the Holdout portion, using coefficients from the fitted models and values of independent variables from the hold-out portion.
- Use forecast evaluation criteria as out-of-sample model selection criteria.
- Use the in-sample and out-of-sample criteria to select the best model(s).
- Use the best model(s) to forecast the future responses.

```r
 # Fitting Portion:globtemp from 1880:1993
xfit=globtemp[11:104]
nfit=length(xfit)   #94

# Holdout Portion: globtemp from 1994:2003
xfore=gtemp[105:114]
nfore=length(xfore) #10
```

```r
# k=1: Linear Trend Model fit to globtemp from 1880:2003, the
# fitting portion of the data.
# Set up a time vector as an independent variable
tfit=time(xfit)   # time is from 1:94

# OLS fit: Linear Trend Model
mlr.lin = lm(xfit~tfit)
summary(mlr.lin)
```

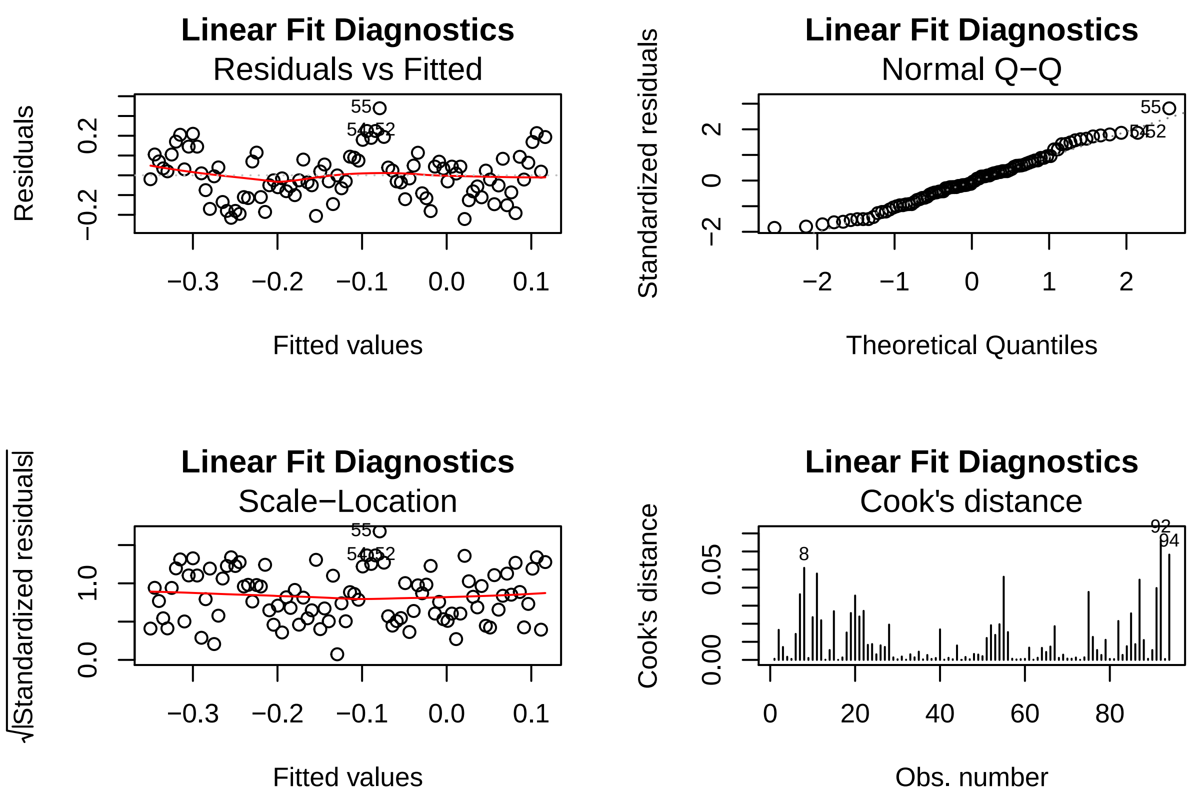
```
##
## Call:
## lm(formula = xfit ~ tfit)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.22120 -0.08511 -0.01026  0.07323  0.33917
##
## Coefficients:
```

```
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.3551819  0.0251696  -14.11   <2e-16 ***
## tfit         0.0050184  0.0004601   10.91   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.121 on 92 degrees of freedom
## Multiple R-squared:  0.5639, Adjusted R-squared:  0.5592
## F-statistic:   119 on 1 and 92 DF,  p-value: < 2.2e-16
```

```
anova(mlr.lin)
```

```
## Analysis of Variance Table
##
## Response: xfit
##           Df Sum Sq Mean Sq F value    Pr(>F)
## tfit       1 1.7429 1.74294  118.96 < 2.2e-16 ***
## Residuals 92 1.3479 0.01465
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Residual Diagnostic Plots for mlr.lin
windows()
par(mfrow=c(2,2)) # plot 4 figures, 2 in each of 2 rows
plot(mlr.lin, main="Linear Fit Diagnostics",which = 1:4)
```



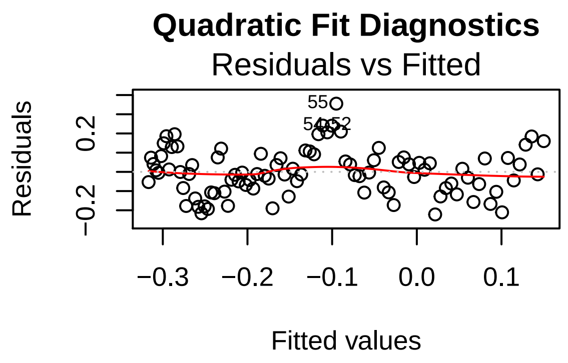
```r
# k=2: Quadratic Trend Model fit
# Set up quadratic term independent variable
tsqfit=tfit^2/factorial(2)

mlr.quad=lm(xfit~tfit+tsqfit)
summary(mlr.quad)
```

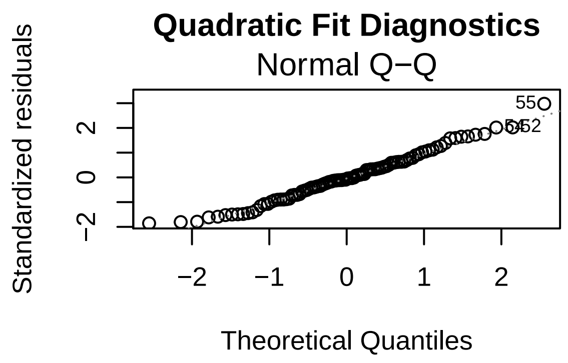
```
##
## Call:
## lm(formula = xfit ~ tfit + tsqfit)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.22167 -0.08431 -0.00803  0.07353  0.35509
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.196e-01  3.815e-02  -8.376 6.33e-13 ***
## tfit         2.794e-03  1.854e-03   1.507    0.135
## tsqfit       4.683e-05  3.781e-05   1.238    0.219
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1207 on 91 degrees of freedom
## Multiple R-squared:  0.5711, Adjusted R-squared:  0.5617
## F-statistic: 60.59 on 2 and 91 DF,  p-value: < 2.2e-16
```

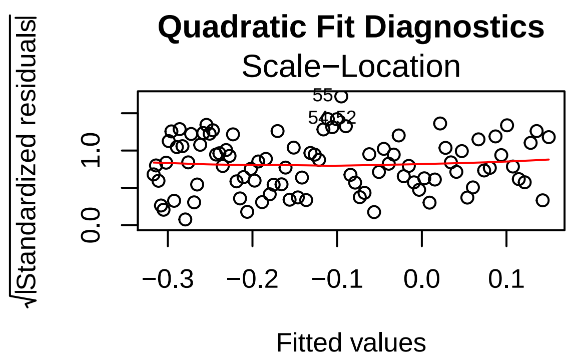
```r
anova(mlr.quad)
```

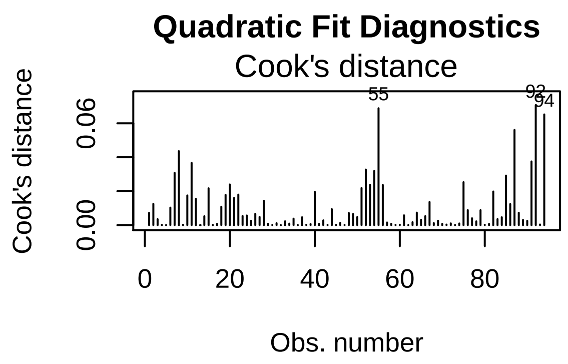
```
## Analysis of Variance Table
##
## Response: xfit
##           Df  Sum Sq Mean Sq  F value Pr(>F)
## tfit       1 1.74294 1.74294 119.6521 <2e-16 ***
## tsqfit     1 0.02234 0.02234   1.5334 0.2188
## Residuals 91 1.32557 0.01457
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Residual Diagnostic Plots for mlr.quad
windows()
par(mfrow=c(2,2))
plot(mlr.quad, main="Quadratic Fit Diagnostics",which = 1:4)
```









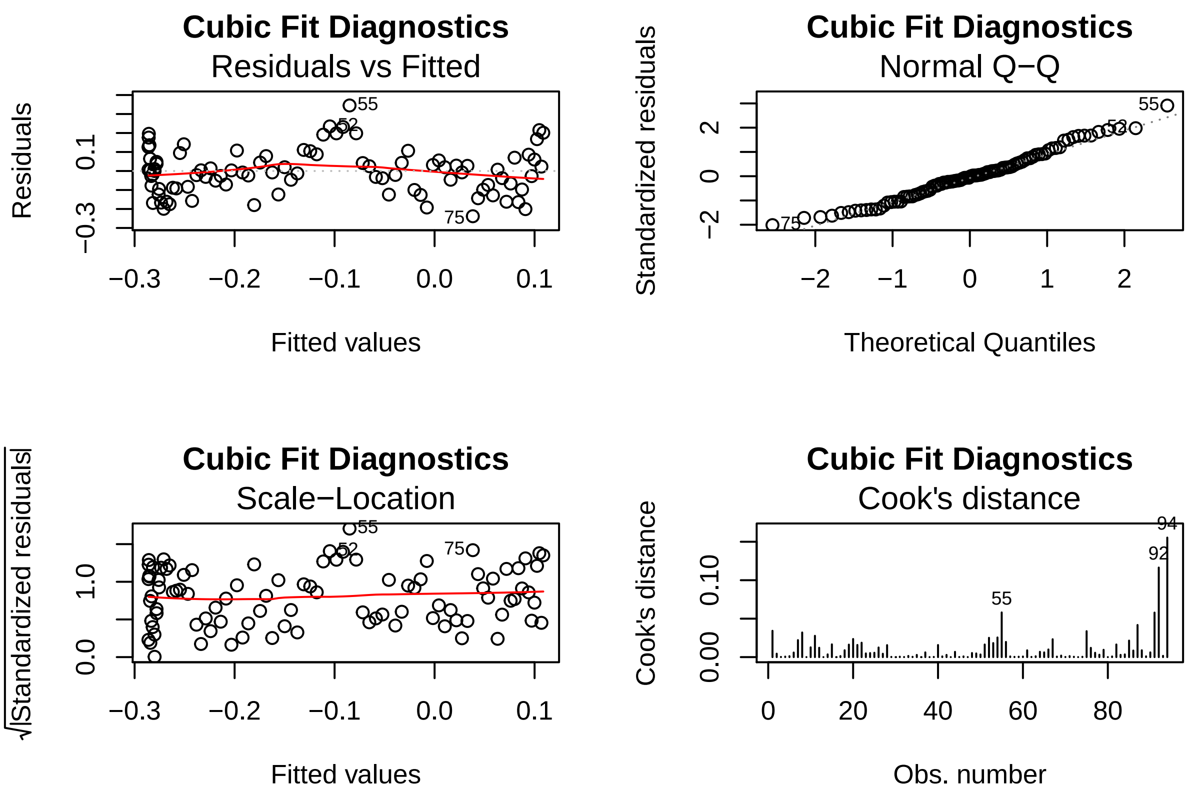
```r
# k=3: Cubic Trend Model fit
# Set up cubic term independent variable
tcubfit=tfit^3/factorial(3)

mlr.cub=lm(xfit~tfit+tsqfit+tcubfit)
summary(mlr.cub)
```

```
##
## Call:
## lm(formula = xfit ~ tfit + tsqfit + tcubfit)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.23823 -0.08700 -0.00369  0.06860  0.34501
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.727e-01  5.162e-02  -5.283 8.77e-07 ***
## tfit        -2.972e-03  4.681e-03  -0.635    0.527
## tsqfit       3.487e-04  2.283e-04   1.527    0.130
## tcubfit     -6.355e-06  4.741e-06  -1.340    0.183
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1202 on 90 degrees of freedom
## Multiple R-squared:  0.5795, Adjusted R-squared:  0.5655
## F-statistic: 41.35 on 3 and 90 DF,  p-value: < 2.2e-16
```

```r
anova(mlr.cub)
```

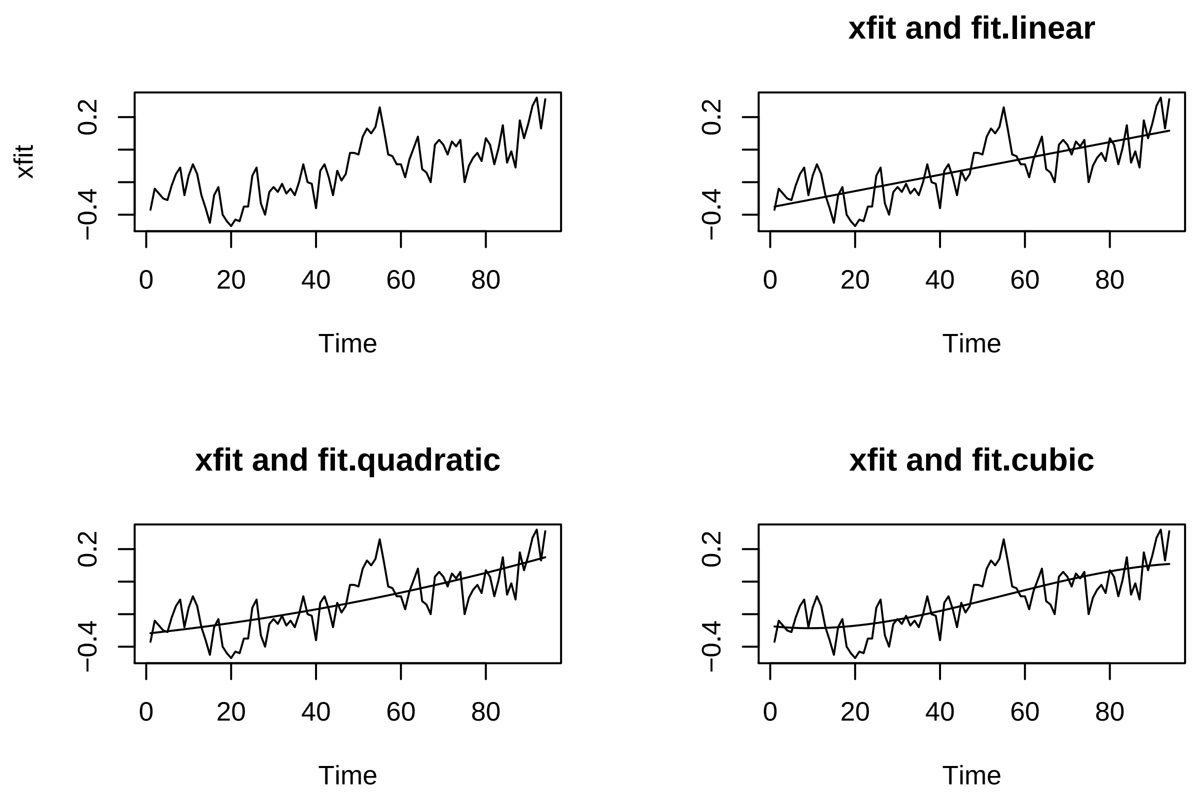
```
## Analysis of Variance Table
##
## Response: xfit
##           Df  Sum Sq Mean Sq  F value Pr(>F)
## tfit       1 1.74294 1.74294 120.6995 <2e-16 ***
## tsqfit     1 0.02234 0.02234   1.5469 0.2168
## tcubfit    1 0.02594 0.02594   1.7966 0.1835
## Residuals 90 1.29963 0.01444
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Residual Diagnostic Plots for mlr.cub
windows()
par(mfrow=c(2,2))
plot(mlr.cub, main="Cubic Fit Diagnostics",which = 1:4)
```



```r
# Plot time series with Fits from k=1,2,3 Models
par(mfrow=c(2,2))
ts.plot(xfit)  # Time Series Plot
# Plot of xfit vs mlr.lin$fitted
plin=cbind(xfit,mlr.lin$fitted)
ts.plot(plin,main="xfit and fit.linear")
pquad=cbind(xfit,mlr.quad$fitted)
```

```
ts.plot(pquad,main="xfit and fit.quadratic")
pcub=cbind(xfit,mlr.cub$fitted)
ts.plot(pcub,main="xfit and fit.cubic")
```



In-sample Model selection criteria:

Which model is better? Compare residual mean squares in ANOVA tables, AIC, BIC. The smaller, the better.

```
#Residual Mean Squares
sigsq.lin=anova(mlr.lin)[["Mean Sq"]][2]
sigsq.quad=anova(mlr.quad)[["Mean Sq"]][3]
sigsq.cub=anova(mlr.cub)[["Mean Sq"]][4]
```

```
### Linear Trend Model, k=1
k=1
# Akaike Information Criterion,AIC
AIC.lin = AIC(mlr.lin)/nfit
cat("AIC for k=1 is",AIC.lin, "\n")
```

```
## AIC for k=1 is -1.343035
```

```
# Or RHS is AIC(mlr.lin)/length(xfit)
# Or RHS is AIC(logLik(mlr.lin))/nfit
# Or RHS is log(sigsq.lin)+(nfit+2*k)/nfit
```

```r
# Bayesian Information Criterion,BIC
BIC.lin = BIC(mlr.lin)/nfit
cat("BIC for k=1 is",BIC.lin, "\n")
```

```
## BIC for k=1 is -1.261866
```

```r
# Or RHS is BIC(mlr.lin)/length(xfit)
# Or RHS is BIC(logLik(mlr.lin))/nfit
# Or RHS is log(sigsq.lin)+(k*log(nfit))/nfit

# Corrected AIC, i.e., AICc using formula
AICc.lin=log(sigsq.lin)+(nfit+k)/(nfit-k-2)
cat("AICc for k=1 is",AICc.lin, "\n")
```

```
## AICc for k=1 is -3.17928
```

```r
### Quadratic Trend Model, k=2
k=2
# AIC: Try the other ways
AIC.quad = AIC(mlr.quad)/nfit
cat("AIC for k=2 is",AIC.quad, "\n")
```

```
## AIC for k=2 is -1.338469
```

```r
#BIC: Try the other ways
BIC.quad = BIC(mlr.quad)/nfit
cat("BIC for k=2 is",BIC.quad, "\n")
```

```
## BIC for k=2 is -1.230244
```

```r
# AICc using formula
AICc.quad=log(sigsq.quad)+(nfit+k)/(nfit-k-2)
cat("AICc for k=2 is",AICc.quad, "\n")
```

```
## AICc for k=2 is -3.162351
```

```r
### Cubic Trend Model, k=3
k=3
# AIC:Try the other ways
AIC.cub = AIC(mlr.cub)/nfit
cat("AIC for k=3 is",AIC.cub, "\n")
```

```
## AIC for k=3 is -1.336958
```

```r
# BIC: Try the other ways
BIC.cub = BIC(mlr.cub)/nfit
cat("BIC for k=3 is",BIC.cub, "\n")
```

```
## BIC for k=3 is -1.201677
```

```r
# AICc using formula
AICc.cub=log(sigsq.cub)+(nfit+k)/(nfit-k-2)
cat("AICc for k=3 is",AICc.cub, "\n")
```

```
## AICc for k=3 is -3.147845
```

Forecasting Holdout/Test Data

```
# (iv) Forecast Evaluation for Different Models
# Set up and predict for next L=nfore=10 times in the future.
# That is, predict the Test/Forecast Validation portion
# of data from 1994 to 2003

# Set up for SLR (k=1) model
new <- data.frame(tfit=c(95:104))
# use predict function
pfore.lin=predict(mlr.lin,new,se.fit = TRUE)
pfore.lin$fit   #point predictions
```

```
##         1         2         3         4         5         6         7
## 0.1215649 0.1265832 0.1316016 0.1366200 0.1416384 0.1466568 0.1516752
##         8         9        10
## 0.1566936 0.1617120 0.1667303
```

```
#Note: you can also get all this info, if you like
pfore.lin$se.fit
```

```
##          1          2          3          4          5          6
## 0.02516961 0.02557015 0.02597266 0.02637705 0.02678324 0.02719115
##          7          8          9         10
## 0.02760070 0.02801183 0.02842445 0.02883851
```

```
pfore.lin$df
```

```
## [1] 92
```

```
pfore.lin$residual.scale
```

```
## [1] 0.121042
```

```
# Observed data for 1994-2003 is in xfore
# Forecast/Prediction errors: Observed - Predicted
efore.lin=xfore-pfore.lin$fit     #length is nfore=length(xfore)
```

Forecast Evaluation Criteria:

Suppose we denote the forecast errors by $e_n(\ell)$ where n is the forecast origin and ℓ is the forecast lead.

We compute and discuss Mean error, Mean Percent Error, Mean Squared Error, Mean Absolute Error, Mean Absolute Percent Error.

$$\begin{aligned} ME &= \bar{e} = \frac{1}{L}\sum_{\ell=1}^{L} e_n(\ell); & (3)\\ MPE &= \frac{100}{L}\sum_{\ell=1}^{L}\{e_n(\ell)/X_{n+\ell}\}; & (4)\\ MSE &= \frac{1}{L}\sum_{\ell=1}^{L} e_n^2(\ell); & (5)\\ MAE &= \frac{1}{L}\sum_{\ell=1}^{L} |e_n(\ell)|; & (6)\\ MAPE &= \frac{100}{L}\sum_{\ell=1}^{L} |e_n(\ell)/X_{n+\ell}|; & (7)\end{aligned}$$

The first two criteria are Mean Error (ME) and Mean Percent Error (MPE) and indicate bias in forecasts.

The last three are Mean Squared Error (MSE), Mean Absolute Error (MAE) and Mean Absolute Percent Error (MAPE) and assess forecast accuracy.

In many cases, the MAPE is used for model selection, because it is a percentage which accounts for the scale of the data and can be useful for broad comparison across models/data. However, when data exhibits periodic behavior or some of the data values are very close to zero, the MAPE can be misleading. It is a good idea to explore the forecasts using plots and perhaps use other forecast accuracy criteria such as MSE or MAE.

In some situations, the information criteria and forecast accuracy criteria may select different models! The user relies on whichever is more important, in-sample accuracy or out-of-sample forecast accuracy.

```
# Forecast evaluation criteria for k=1 (linear)
me.lin=mean(efore.lin)                          # Mean Error
mpe.lin=100*(mean(efore.lin/xfore))             # Mean Percent Error
mse.lin=sum(efore.lin**2)/nfore                 # Mean Squared Error
mae.lin=mean(abs(efore.lin))                    # Mean Absolute Error
mape.lin=100*mean(abs(efore.lin/xfore))         # Mean Absolute Percent Error
cat("Mean Error (ME) for k=1 is",me.lin, "\n")
```

```
## Mean Error (ME) for k=1 is 0.0548524
```

```
cat("Mean Percent Error (MPE) for k=1  is",mpe.lin, "\n")
```

```
## Mean Percent Error (MPE) for k=1  is -2.210729
```

```
cat("Mean Squared Error (MSE) for k=1  is",mse.lin, "\n")
```

```
## Mean Squared Error (MSE) for k=1  is 0.01405649
```

```
cat("Mean Absolute Error (MAE) for k=1  is",mae.lin, "\n")
```

```
## Mean Absolute Error (MAE) for k=1  is 0.09449081
```

```
cat("Mean Absolute Percent Error (MAPE) for k=1  is",mape.lin, "\n")
```

```
## Mean Absolute Percent Error (MAPE) for k=1  is 49.9734
```

```
# Set up for prediction with k=2 model
tfit=c(95:104)
tsqfit=tfit^2/factorial(2)
matq=matrix(c(tfit,tsqfit),nrow=10,ncol=2,dimnames=list(c(),c("tfit","tsqfit")))
matq
```

```
##       tfit tsqfit
##  [1,]   95 4512.5
##  [2,]   96 4608.0
##  [3,]   97 4704.5
##  [4,]   98 4802.0
##  [5,]   99 4900.5
##  [6,]  100 5000.0
##  [7,]  101 5100.5
##  [8,]  102 5202.0
##  [9,]  103 5304.5
## [10,]  104 5408.0
```

```
newnq <- data.frame(matq)

pfore.quad=predict(mlr.quad,newnq,se.fit=TRUE)
pfore.quad$fit  # point predictions
```

```
##         1         2         3         4         5         6         7
## 0.1571524 0.1644185 0.1717313 0.1790910 0.1864975 0.1939508 0.2014509
##         8         9        10
## 0.2089979 0.2165917 0.2242324
```

```
efore.quad=xfore-pfore.quad$fit  # Forecast errors
me.quad=mean(efore.quad)
mpe.quad=100*(mean(efore.quad/xfore))
mse.quad=sum(efore.quad**2)/nfore
mae.quad=mean(abs(efore.quad))
mape.quad=100*mean(abs(efore.quad/xfore))

cat("Mean Error (ME) for k=2 is",me.quad, "\n")
```

```
## Mean Error (ME) for k=2 is 0.008588562
```

```
cat("Mean Percent Error (MPE) for k=2  is",mpe.quad, "\n")
```

```
## Mean Percent Error (MPE) for k=2  is -34.48205
```

```
cat("Mean Squared Error (MSE) for k=2  is",mse.quad, "\n")
```

```
## Mean Squared Error (MSE) for k=2  is 0.0108592
```

```
cat("Mean Absolute Error (MAE) for k=2  is",mae.quad, "\n")
```

```
## Mean Absolute Error (MAE) for k=2  is 0.09420397
```

```
cat("Mean Absolute Percent Error (MAPE) for k=2  is",mape.quad, "\n")
```

```
## Mean Absolute Percent Error (MAPE) for k=2  is 65.84175
```

```
# Set up for prediction with k=3 model
tfit=c(95:104)
tsqfit=tfit^2/factorial(2)
```

```
tcubfit=tfit^3/factorial(3)
matc=matrix(c(tfit,tsqfit,tcubfit),nrow=10,ncol=3,dimnames = list(c(),c("tfit","tsqfit","tcubfit")))
matc
```

```
##       tfit tsqfit  tcubfit
##  [1,]   95 4512.5 142895.8
##  [2,]   96 4608.0 147456.0
##  [3,]   97 4704.5 152112.2
##  [4,]   98 4802.0 156865.3
##  [5,]   99 4900.5 161716.5
##  [6,]  100 5000.0 166666.7
##  [7,]  101 5100.5 171716.8
##  [8,]  102 5202.0 176868.0
##  [9,]  103 5304.5 182121.2
## [10,]  104 5408.0 187477.3
```

```
newnc=data.frame(matc)

pfore.cub=predict(mlr.cub,newnc,se.fit = TRUE)
pfore.cub$fit  # point predictions
```

```
##         1         2         3         4         5         6         7
## 0.1103007 0.1116486 0.1127351 0.1135539 0.1140985 0.1143626 0.1143400
##         8         9        10
## 0.1140241 0.1134087 0.1124874
```

```
efore.cub=xfore-pfore.cub$fit  # Forecast errors
me.cub=mean(efore.cub)
mpe.cub=100*(mean(efore.cub/xfore))
mse.cub=sum(efore.cub**2)/nfore
mae.cub=mean(abs(efore.cub))
mape.cub=100*mean(abs(efore.cub/xfore))

cat("Mean Error (ME) for k=3 is",me.cub, "\n")
```

```
## Mean Error (ME) for k=3 is 0.08590403
```

```
cat("Mean Percent Error (MPE) for k=3  is",mpe.cub, "\n")
```

```
## Mean Percent Error (MPE) for k=3  is 17.5012
```

```
cat("Mean Squared Error (MSE) for k=3  is",mse.cub, "\n")
```

```
## Mean Squared Error (MSE) for k=3  is 0.01912071
```

```
cat("Mean Absolute Error (MAE) for k=3  is",mae.cub, "\n")
```

```
## Mean Absolute Error (MAE) for k=3  is 0.1022939
```

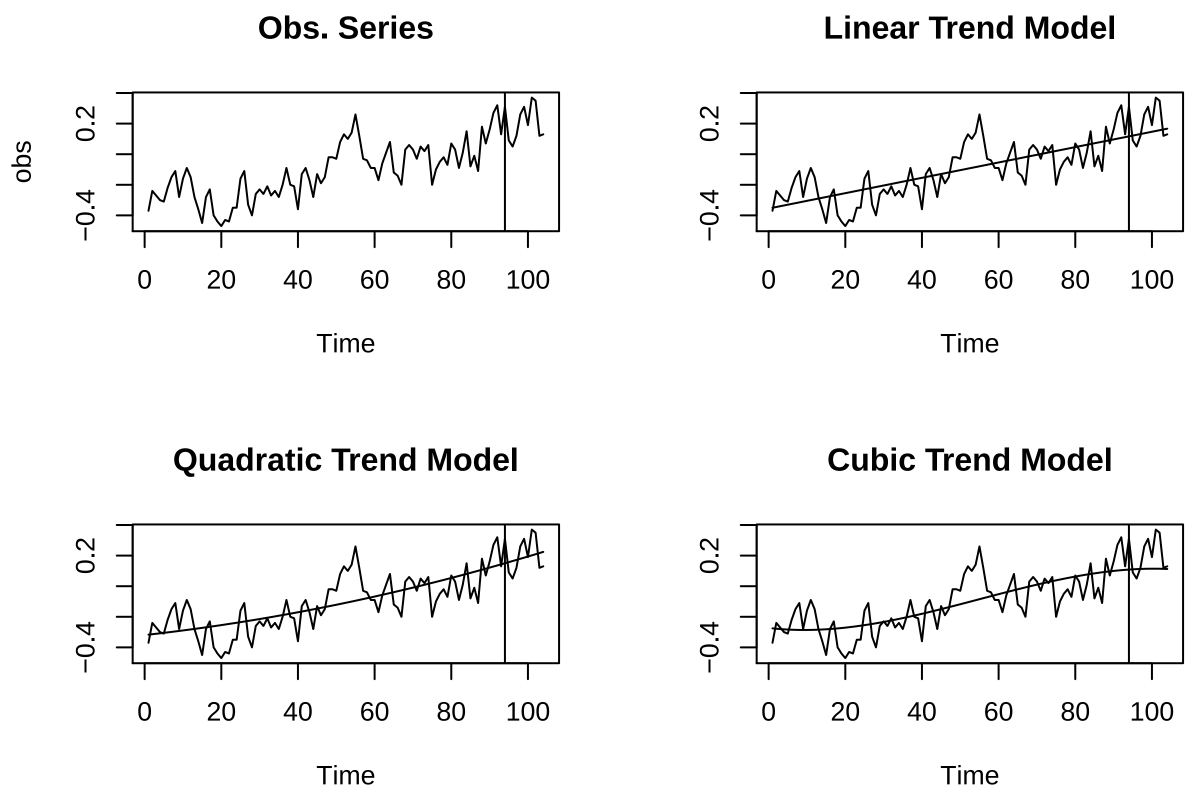
```
cat("Mean Absolute Percent Error (MAPE) for k=3  is",mape.cub, "\n")
```

```
## Mean Absolute Percent Error (MAPE) for k=3  is 46.67193
```

Comparison of forecasted and actual values on Holdout Data

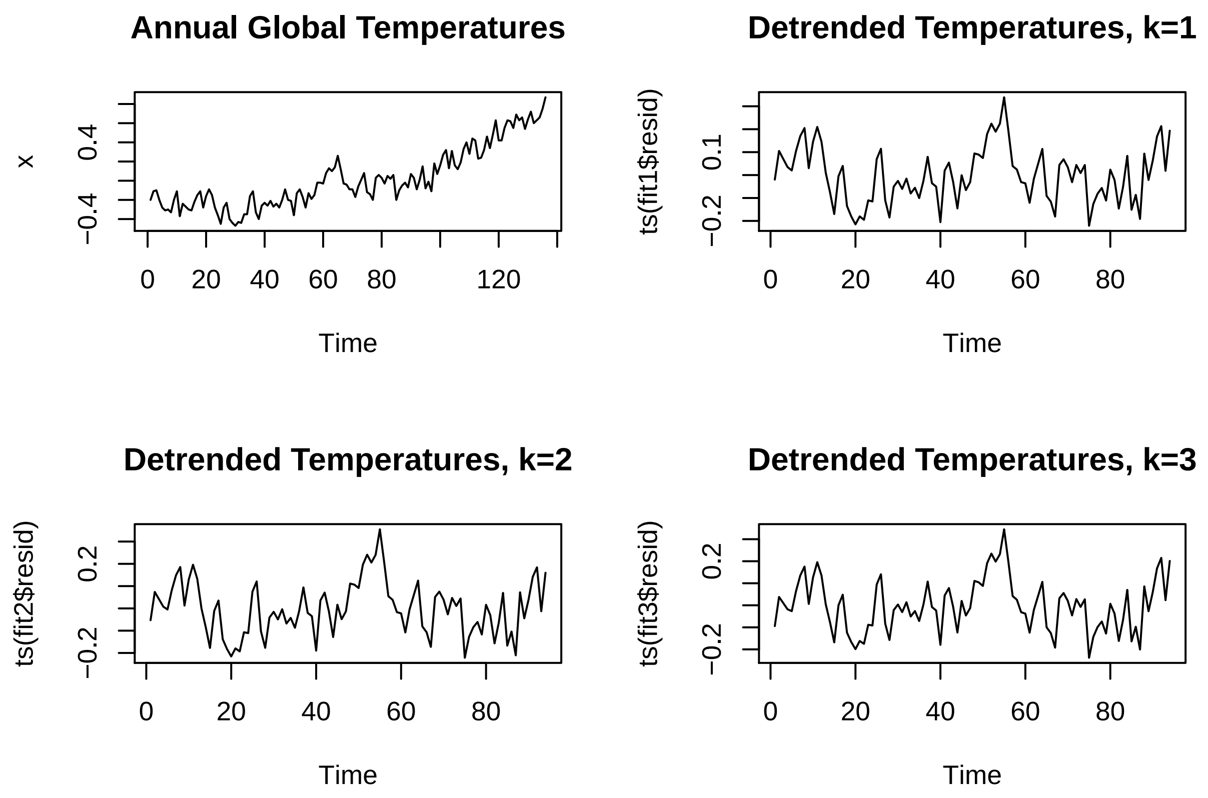
```
### Plot of time series with fits and out of sample forecasts
### Linear Trend Model fits and forecasts
linff=c(mlr.lin$fitted,pfore.lin$fit)
```

```r
### Quadratic Trend Model fits and forecasts
quadff=c(mlr.quad$fitted,pfore.quad$fit)
### Cubic Trend Model fits and forecasts
cubff=c(mlr.cub$fitted,pfore.cub$fit)

# observed data (n=98)
obs=c(xfit,xfore)
# Bind observed data and fits+forecasts
obslin=cbind(obs,linff)
obsquad=cbind(obs,quadff)
obscub=cbind(obs,cubff)
time=c(1:length(obs))

par(mfrow=c(2,2))
ts.plot(obs,main="Obs. Series")
abline(v=94)
ts.plot(obslin,main="Linear Trend Model")
abline(v=94)
ts.plot(obsquad,main="Quadratic Trend Model")
abline(v=94)
ts.plot(obscub,main="Cubic Trend Model")
abline(v=94)
```



Detrending the Time Series

```r
### Detrending a Time Series after a Polynomial Model Fit
x=globtemp
fit1=mlr.lin
fit2=mlr.quad
fit3=mlr.cub
par(mfrow=c(2,2))
### Original series as x
ts.plot(x,main="Annual Global Temperatures")
### Detrended series, k=1
ts.plot(ts(fit1$resid),main="Detrended Temperatures, k=1")
### Detrended series, k=2
ts.plot(ts(fit2$resid),main="Detrended Temperatures, k=2")
### Detrended series, k=3
ts.plot(ts(fit3$resid),main="Detrended Temperatures, k=3")
```



Modeling Time Series with Trend and Seasonality

A time series X_t may have an additive decomposition:

$$X_t = m_t + S_t + E_t, \tag{8}$$

where the trend component could be a polynomial of degree p (as before) and S_t may be a periodic function with period s.

S_t may be modeled using **indicator functions of time**:

$$S_t = \sum_{i=1}^{s} \gamma_i IND_{i,t}$$

where $\text{IND}_{i,t}$ are indicator functions of time t, defined by

$$\text{IND}_{i,t} = \begin{cases} 1 & \text{if } t \text{ corresponds to seasonal period } i \\ & 0 \text{ otherwise} \end{cases}$$

When $s = 12$, we set up indicator variables, one for each month. That is, for $i = 1, \ldots, s$, define

For any Jan. we have $\text{IND}_{1,t} = 1$, while $\text{IND}_{i,t} = 0$ for $i \neq 1$

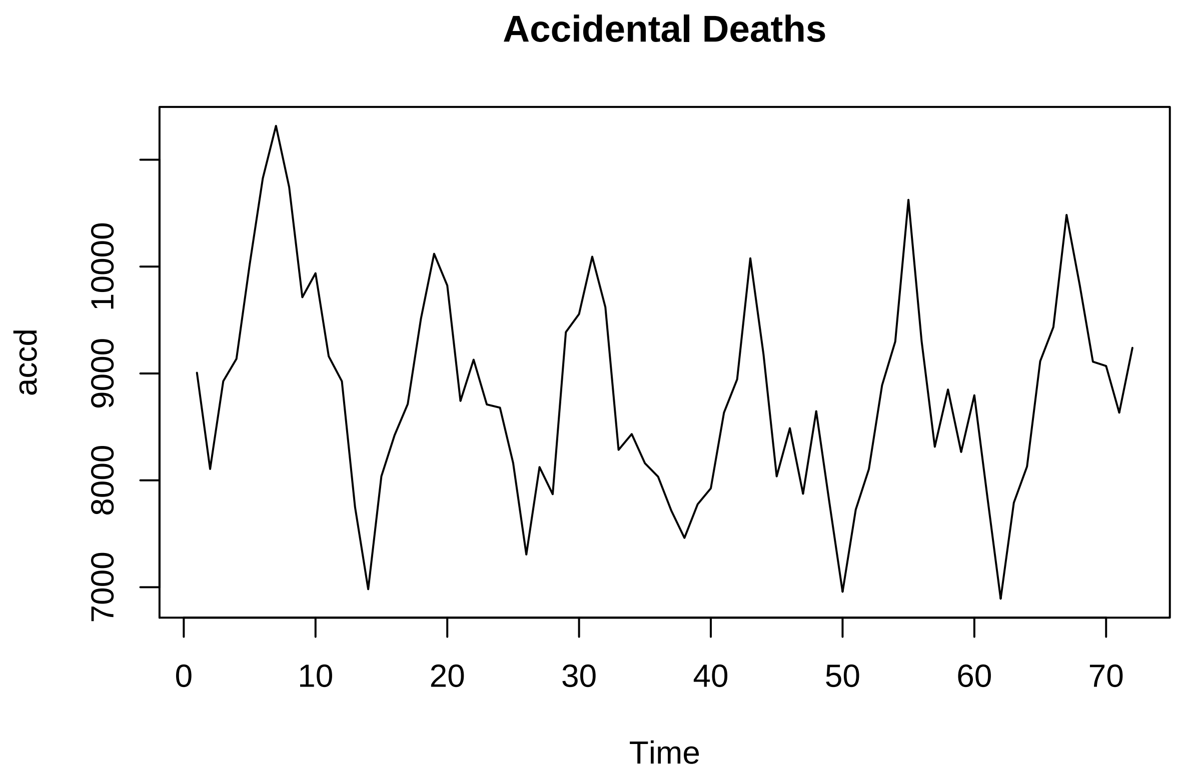
For any Feb. we have $\text{IND}_{2,t} = 1$, while $\text{IND}_{i,t} = 0$ for $i \neq 2$

etc.

For any Dec. we have $\text{IND}_{12,t} = 1$, while $\text{IND}_{i,t} = 0$ for $i \neq 12$

Example: Accidental Deaths Data

```r
accd <- read.csv("C:/Users/STATISTICS/Desktop/Manipal/Data/accdeaths.csv",header=F)

accd=ts(accd)    #length =72
ts.plot(accd,main="Accidental Deaths")
```



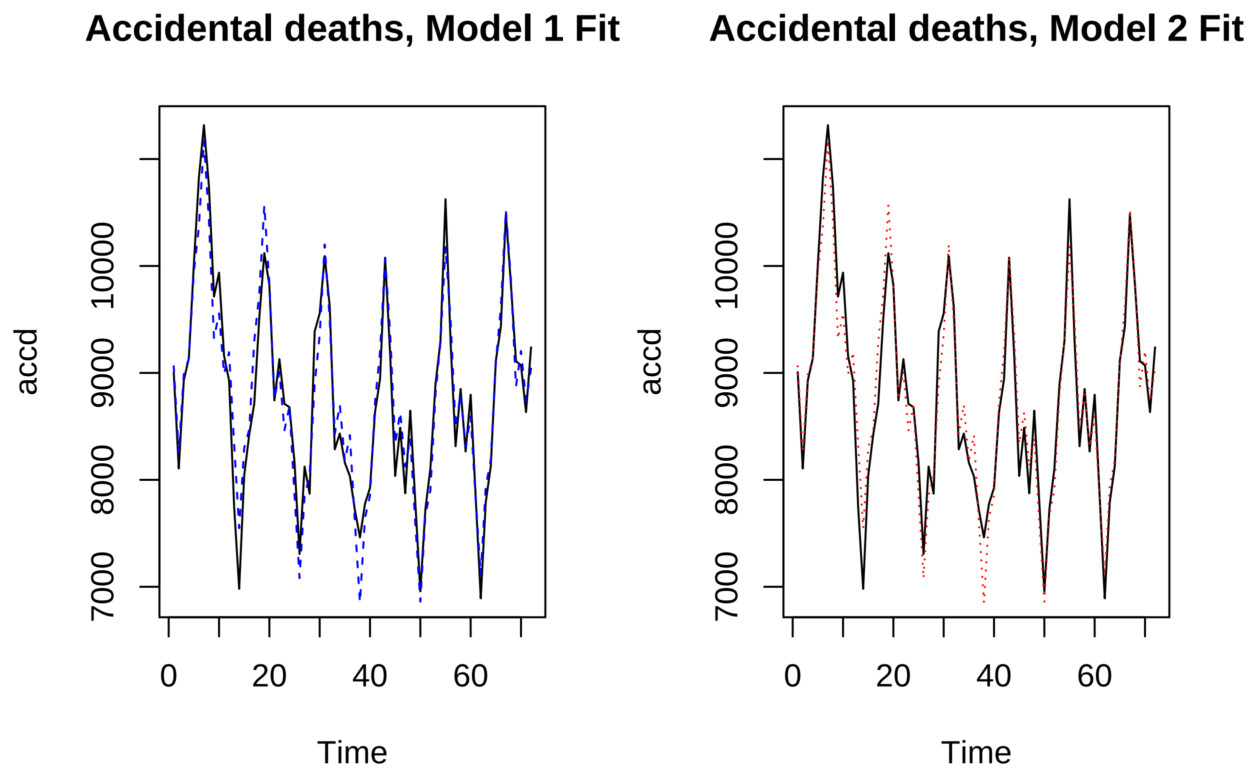
```r
# Regression with Polynomial terms for m_t and
# Monthly indicators for S_t
# Polynomial terms: t, t^2/2!
t=1:length(accd)
tsq=t^2/factorial(2)
# Seasonal terms: 12 monthly indicators
per=12
sets=length(accd)/per
month=factor(rep(1:per,sets))

###  Model 1. t,tsq,12 seasonal indicators,no intercept
accd.model1 = lm(accd~t+tsq+month -1)
summary(accd.model1)
```

```
##
## Call:
## lm(formula = accd ~ t + tsq + month - 1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -587.84 -153.18   -9.46  178.04  609.04
##
## Coefficients:
##           Estimate Std. Error t value Pr(>|t|)
## t         -71.9782     6.3391  -11.36 2.27e-16 ***
## tsq         1.6531     0.1682    9.83 5.83e-14 ***
## month1   9133.8707   143.3724   63.71  < 2e-16 ***
## month2   8393.6101   144.3576   58.15  < 2e-16 ***
## month3   9191.8631   145.2706   63.27  < 2e-16 ***
## month4   9409.4631   146.1104   64.40  < 2e-16 ***
## month5  10285.7432   146.8766   70.03  < 2e-16 ***
## month6  10768.2037   147.5690   72.97  < 2e-16 ***
## month7  11637.3443   148.1883   78.53  < 2e-16 ***
## month8  10943.6653   148.7352   73.58  < 2e-16 ***
## month9   9903.1664   149.2112   66.37  < 2e-16 ***
## month10 10194.1812   149.6181   68.14  < 2e-16 ***
## month11  9681.7095   149.9582   64.56  < 2e-16 ***
## month12  9938.5848   150.2340   66.15  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 275.4 on 58 degrees of freedom
## Multiple R-squared:  0.9992, Adjusted R-squared:  0.999
## F-statistic:  5292 on 14 and 58 DF,  p-value: < 2.2e-16
```

```
###  Model 2. t,tsq,11 seasonal indicators and intercept
mon=factor(rep(c(1:(per-1),0),sets))
accd.model2=lm(accd~t+tsq+mon)
summary(accd.model2)
```

```
##
## Call:
## lm(formula = accd ~ t + tsq + mon)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -587.84 -153.18   -9.46  178.04  609.04
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  9938.5848   150.2340  66.154  < 2e-16 ***
## t             -71.9782     6.3391 -11.355 2.27e-16 ***
## tsq             1.6531     0.1682   9.830 5.83e-14 ***
## mon1         -804.7141   159.9783  -5.030 5.04e-06 ***
## mon2        -1544.9747   159.8158  -9.667 1.07e-13 ***
## mon3         -746.7217   159.6715  -4.677 1.79e-05 ***
## mon4         -529.1217   159.5435  -3.316  0.00158 **
## mon5          347.1584   159.4301   2.177  0.03352 *
## mon6          829.6189   159.3304   5.207 2.64e-06 ***
## mon7         1698.7595   159.2438  10.668 2.68e-15 ***
## mon8         1005.0805   159.1703   6.314 4.11e-08 ***
## mon9          -35.4184   159.1105  -0.223  0.82463
## mon10         255.5964   159.0655   1.607  0.11352
## mon11        -256.8753   159.0368  -1.615  0.11170
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 275.4 on 58 degrees of freedom
## Multiple R-squared:  0.9325, Adjusted R-squared:  0.9174
## F-statistic: 61.65 on 13 and 58 DF,  p-value: < 2.2e-16
```

```
#tq=t^3/factorial(3)
#accd.model3=lm(accd~t+tsq+tq+mon)
#AIC increased

### Plot data and fits from Model 1 and Model 2
### Verify Model 1 and Model 2 are equivalent!
windows()
par(mfrow=c(1,2))
ts.plot(accd,main="Accidental deaths, Model 1 Fit")
lines(accd.model1$fit,col="blue", lty="dashed")
ts.plot(accd,main="Accidental deaths, Model 2 Fit")
lines(accd.model2$fit,col="red", lty="dotted")
```



Exercise. Fit a suitable time series regression model to Air Passengers data and use this to forecast the next year.