Case Study 3: Time Series

Dr. Kempthorne

September 30, 2013

Contents

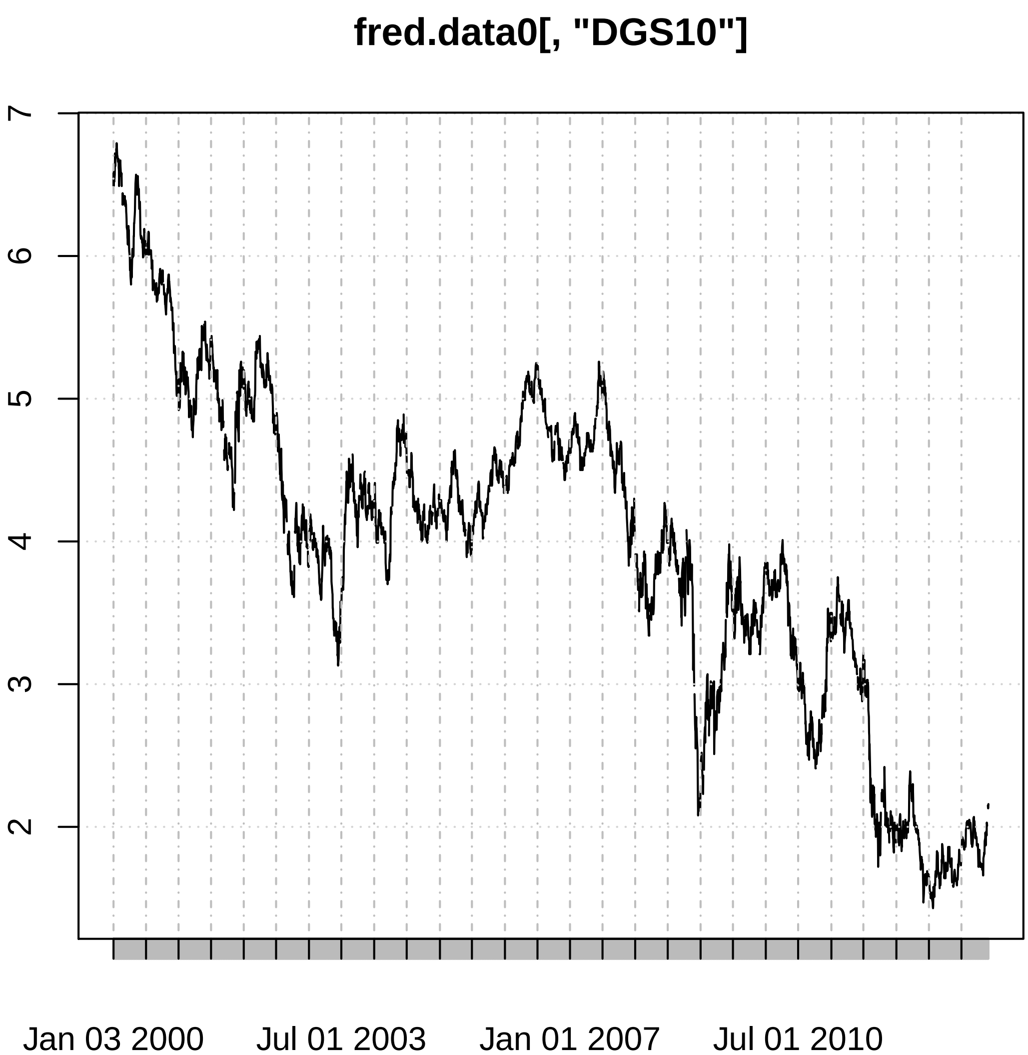
1 Time Series Analysis of the US Treasury 10-Year Yield

1.1 Load R libraries and Federal Reserve data

An R script ("fm_casestudy_1_0.r") collects daily US Treasury yield data from FRED, the Federal Reserve Economic Database, and stores them in the R workspace "casestudy_1.RData".

The following commands re-load the data and evaluates the presence and nature of missing values.

```
> source("fm_casestudy_0_InstallOrLoadLibraries.r")
> # load the R workspace created by the script file
> #   fm_casestudy_1_0.r
>
> dbnames0<-load(file="casestudy_1_0.RData")
> # print(dbnames0)
>
>
> # 2. Extract DGS10 time series from the time series object fred.data0
> #   (an object of class xts and zoo, used in the R Packages of the same names)
> head(fred.data0)

           DGS3MO DGS1 DGS5 DGS10 DAAA DBAA DCOILWTICO
2000-01-03   5.48 6.09 6.50  6.58 7.75 8.27         NA
2000-01-04   5.43 6.00 6.40  6.49 7.69 8.21      25.56
2000-01-05   5.44 6.05 6.51  6.62 7.78 8.29      24.65
2000-01-06   5.41 6.03 6.46  6.57 7.72 8.24      24.79
2000-01-07   5.38 6.00 6.42  6.52 7.69 8.22      24.79
2000-01-10   5.42 6.07 6.49  6.57 7.72 8.27      24.71

> tail(fred.data0)

           DGS3MO DGS1 DGS5 DGS10 DAAA DBAA DCOILWTICO
2013-05-24   0.04 0.12 0.90  2.01 3.94 4.76      93.84
2013-05-27     NA   NA   NA    NA   NA   NA         NA
2013-05-28   0.05 0.13 1.02  2.15 4.06 4.88      94.65
2013-05-29   0.05 0.14 1.02  2.13 4.04 4.88      93.13
2013-05-30   0.04 0.13 1.01  2.13 4.06 4.90      93.57
2013-05-31   0.04 0.14 1.05  2.16 4.09 4.95      91.93

> #   DGS10 is the 4th column of the matrix object fred.data0

> library ("graphics")
> library("quantmod")
> plot(fred.data0[,"DGS10"])
```




```
> # There are dates (rows of fred.data0)  with missing values (NAs)
> # Print out the counts of missing values
> #     using the function apply to count the TRUE values in each colum of the
> #     logical matrix is.na(fred.data0), which replaces the matrix fred.data0 with
> #     the element-wise evaluation of the function is.na() which is TRUE if
> #     the argumnet is missing (i.e., NA)
> print( apply(is.na(fred.data0),2,sum))
```

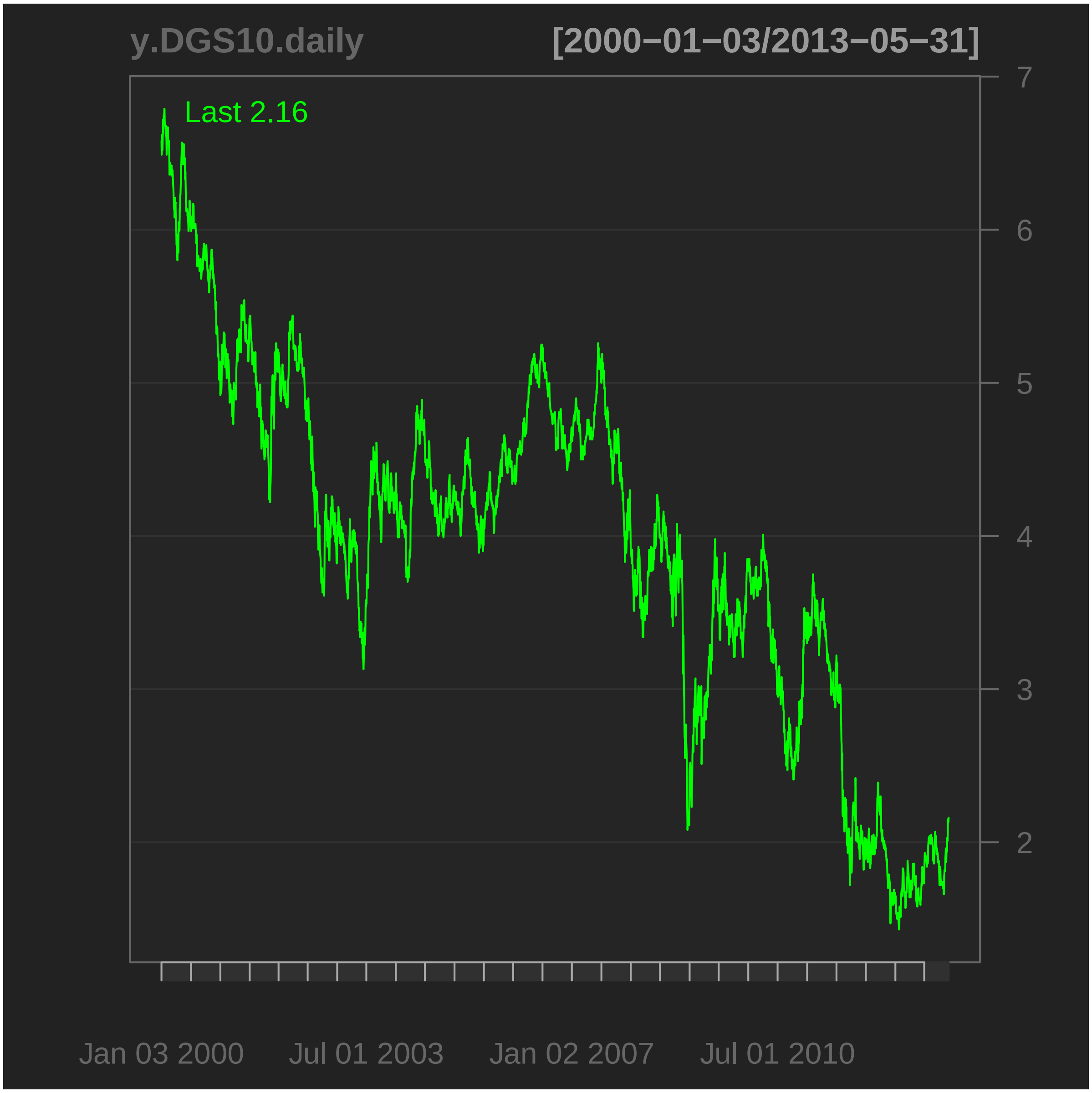
```
   DGS3MO       DGS1       DGS5      DGS10       DAAA       DBAA DCOILWTICO
      144        144        144        144        144        144        134

> # Identify rows for which DGS10 data is missing
> index.fred.data0.notavail<-which(is.na(fred.data0[,"DGS10"])==TRUE)
> print( time(fred.data0)[index.fred.data0.notavail])

  [1] "2000-01-17" "2000-02-21" "2000-04-21" "2000-05-29" "2000-07-04"
  [6] "2000-09-04" "2000-10-09" "2000-11-23" "2000-12-25" "2001-01-01"
 [11] "2001-01-15" "2001-02-19" "2001-04-13" "2001-05-28" "2001-07-04"
 [16] "2001-09-03" "2001-09-11" "2001-09-12" "2001-10-08" "2001-11-12"
 [21] "2001-11-22" "2001-12-25" "2002-01-01" "2002-01-21" "2002-02-18"
 [26] "2002-03-29" "2002-05-27" "2002-07-04" "2002-09-02" "2002-10-14"
 [31] "2002-11-11" "2002-11-28" "2002-12-25" "2003-01-01" "2003-01-20"
 [36] "2003-02-17" "2003-04-18" "2003-05-26" "2003-07-04" "2003-09-01"
 [41] "2003-10-13" "2003-11-11" "2003-11-27" "2003-12-25" "2004-01-01"
 [46] "2004-01-19" "2004-02-16" "2004-04-09" "2004-05-31" "2004-06-11"
 [51] "2004-07-05" "2004-09-06" "2004-10-11" "2004-11-11" "2004-11-25"
 [56] "2004-12-24" "2005-01-17" "2005-02-21" "2005-03-25" "2005-05-30"
 [61] "2005-07-04" "2005-09-05" "2005-10-10" "2005-11-11" "2005-11-24"
 [66] "2005-12-26" "2006-01-02" "2006-01-16" "2006-02-20" "2006-04-14"
 [71] "2006-05-29" "2006-07-04" "2006-09-04" "2006-10-09" "2006-11-23"
 [76] "2006-12-25" "2007-01-01" "2007-01-15" "2007-02-19" "2007-05-28"
 [81] "2007-07-04" "2007-09-03" "2007-10-08" "2007-11-12" "2007-11-22"
 [86] "2007-12-25" "2008-01-01" "2008-01-21" "2008-02-18" "2008-03-21"
 [91] "2008-05-26" "2008-07-04" "2008-09-01" "2008-10-13" "2008-11-11"
 [96] "2008-11-27" "2008-12-25" "2009-01-01" "2009-01-19" "2009-02-16"
[101] "2009-04-10" "2009-05-25" "2009-07-03" "2009-09-07" "2009-10-12"
[106] "2009-11-11" "2009-11-26" "2009-12-25" "2010-01-01" "2010-01-18"
[111] "2010-02-15" "2010-05-31" "2010-07-05" "2010-09-06" "2010-10-11"
[116] "2010-11-11" "2010-11-25" "2010-12-24" "2011-01-17" "2011-02-21"
[121] "2011-04-22" "2011-05-30" "2011-07-04" "2011-09-05" "2011-10-10"
[126] "2011-11-11" "2011-11-24" "2011-12-26" "2012-01-02" "2012-01-16"
[131] "2012-02-20" "2012-05-28" "2012-07-04" "2012-09-03" "2012-10-08"
[136] "2012-10-30" "2012-11-12" "2012-11-22" "2012-12-25" "2013-01-01"
[141] "2013-01-21" "2013-02-18" "2013-03-29" "2013-05-27"

> #Note that the FRED data is missing when there are holidays or market-closes
> #in the bond market of the US.
>
> # Define fred.data0.0 as sub matrix with nonmissing data for DGS10
> fred.data0.0<-fred.data0[which(is.na(fred.data0[,"DGS10"])==FALSE),]
> print(apply(is.na(fred.data0.0),2,sum))

   DGS3MO       DGS1       DGS5      DGS10       DAAA       DBAA DCOILWTICO
        0          0          0          0          1          1         17
```

```
> # Some column variables of fred.data0.0 have missing values
> #   (i.e., DAAA, DBBB, DCOILWTICO).
> #
> # Our focus is on DGS10, the yield of constant-maturity 10 Year US bond.
>
> y.DGS10.daily<-na.omit(fred.data0.0[,"DGS10"])
> dim(y.DGS10.daily)

[1] 3356    1

> dimnames(y.DGS10.daily)[[2]]

[1] "DGS10"

> head(y.DGS10.daily)

           DGS10
2000-01-03  6.58
2000-01-04  6.49
2000-01-05  6.62
2000-01-06  6.57
2000-01-07  6.52
2000-01-10  6.57
```

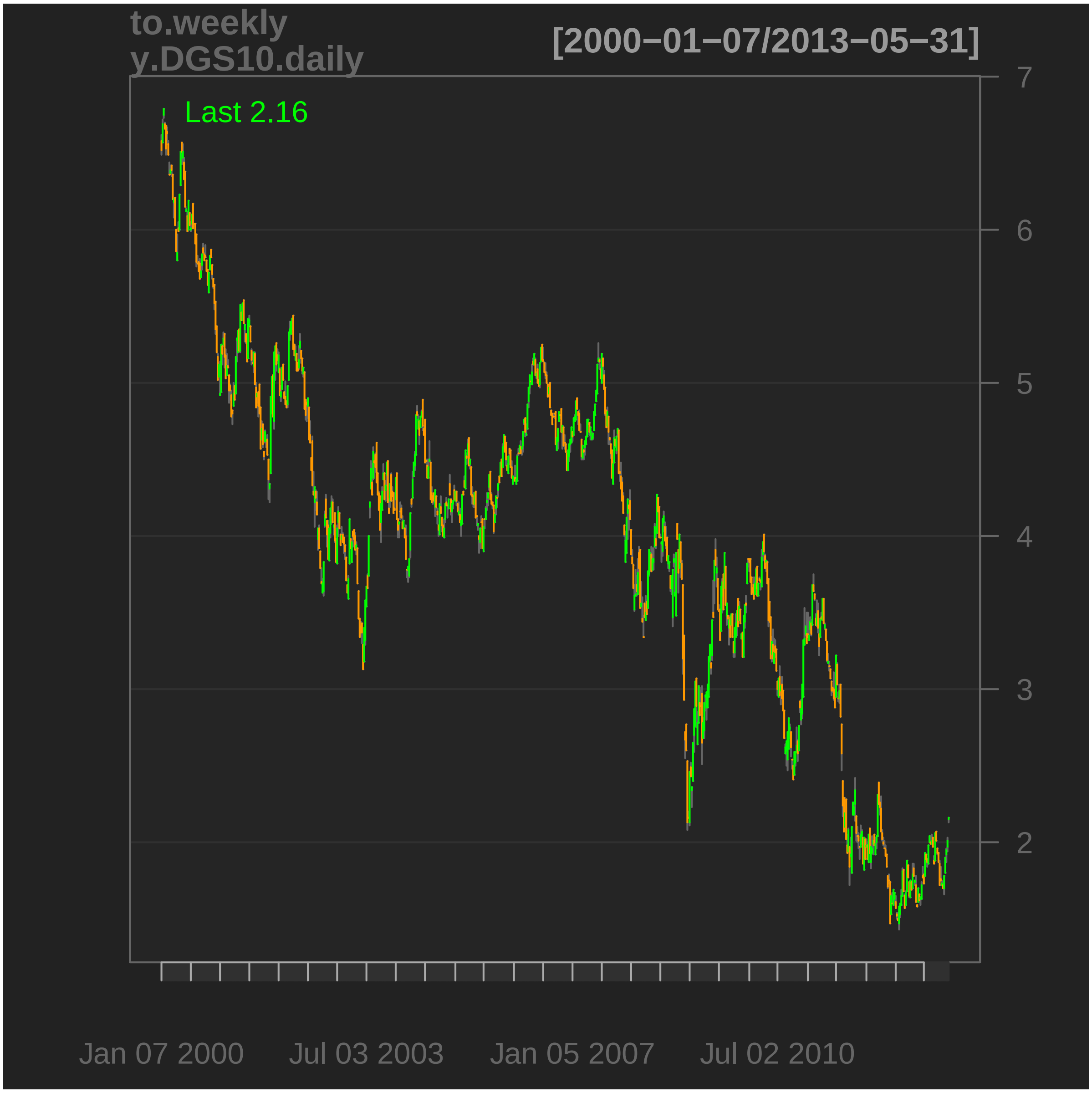
1.2 Create weekly and monthly time series

```
> # The function to.weekly() and to.monthly() converts a time series data object
> # to an Open/High/Low/Close series on a periodicity lower than the input data object.
> head(to.weekly(y.DGS10.daily))

           y.DGS10.daily.Open y.DGS10.daily.High y.DGS10.daily.Low
2000-01-07               6.58               6.62              6.49
2000-01-14               6.57               6.72              6.57
2000-01-21               6.75               6.79              6.73
2000-01-28               6.69               6.70              6.66
2000-02-04               6.68               6.68              6.49
2000-02-11               6.64               6.67              6.56
           y.DGS10.daily.Close
2000-01-07                6.52
2000-01-14                6.69
2000-01-21                6.79
2000-01-28                6.66
2000-02-04                6.53
2000-02-11                6.63

> # Check how the first row of to.weekly(y.DGS10.daily) is consistent with
> # the first row5 rows of y.DGS10.daily
>
> # The function chartSeries()  plots Open/High/Low/Close series
```

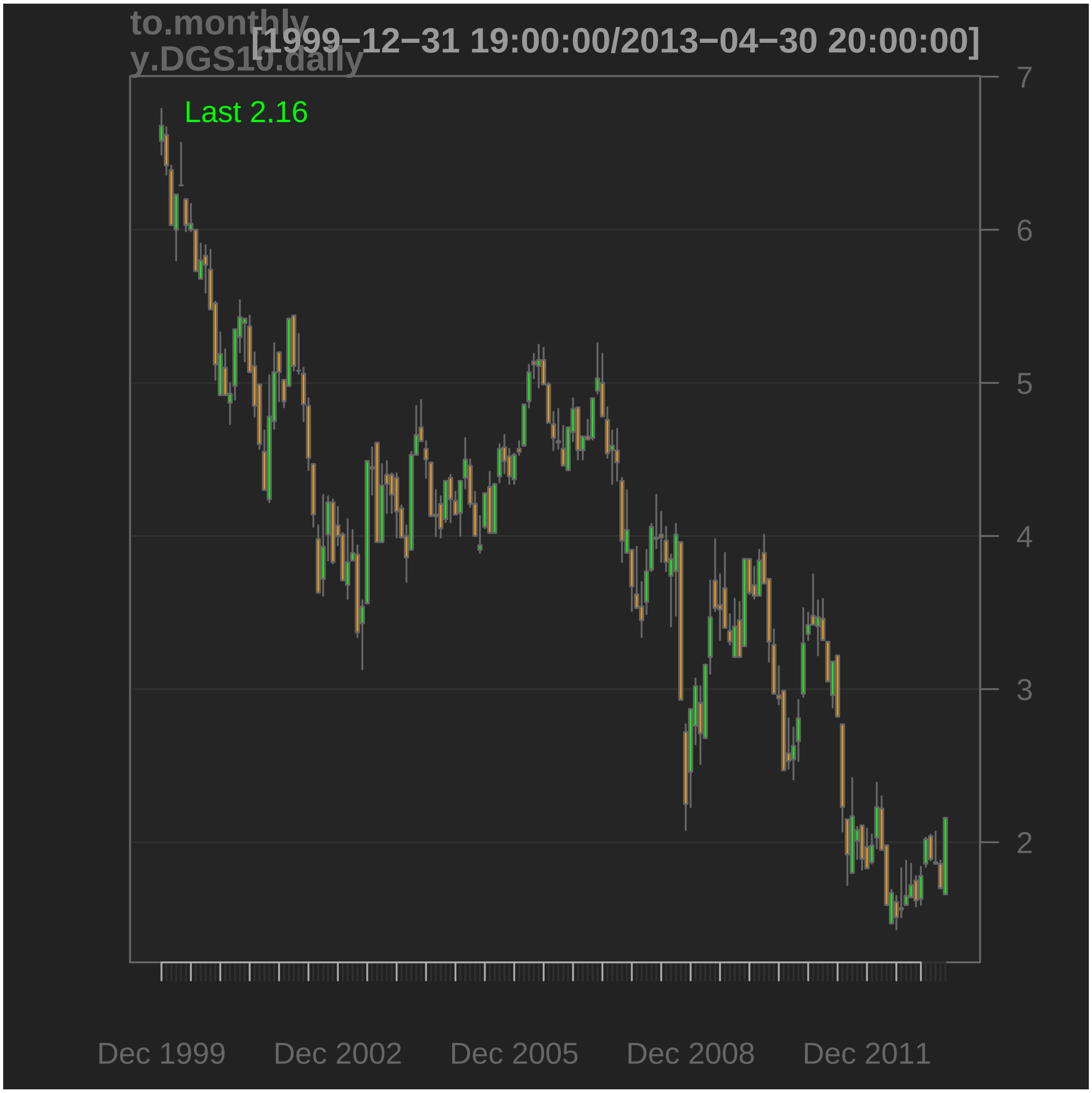
```
> chartSeries(y.DGS10.daily)
>
```



```
> chartSeries(to.weekly(y.DGS10.daily))
```



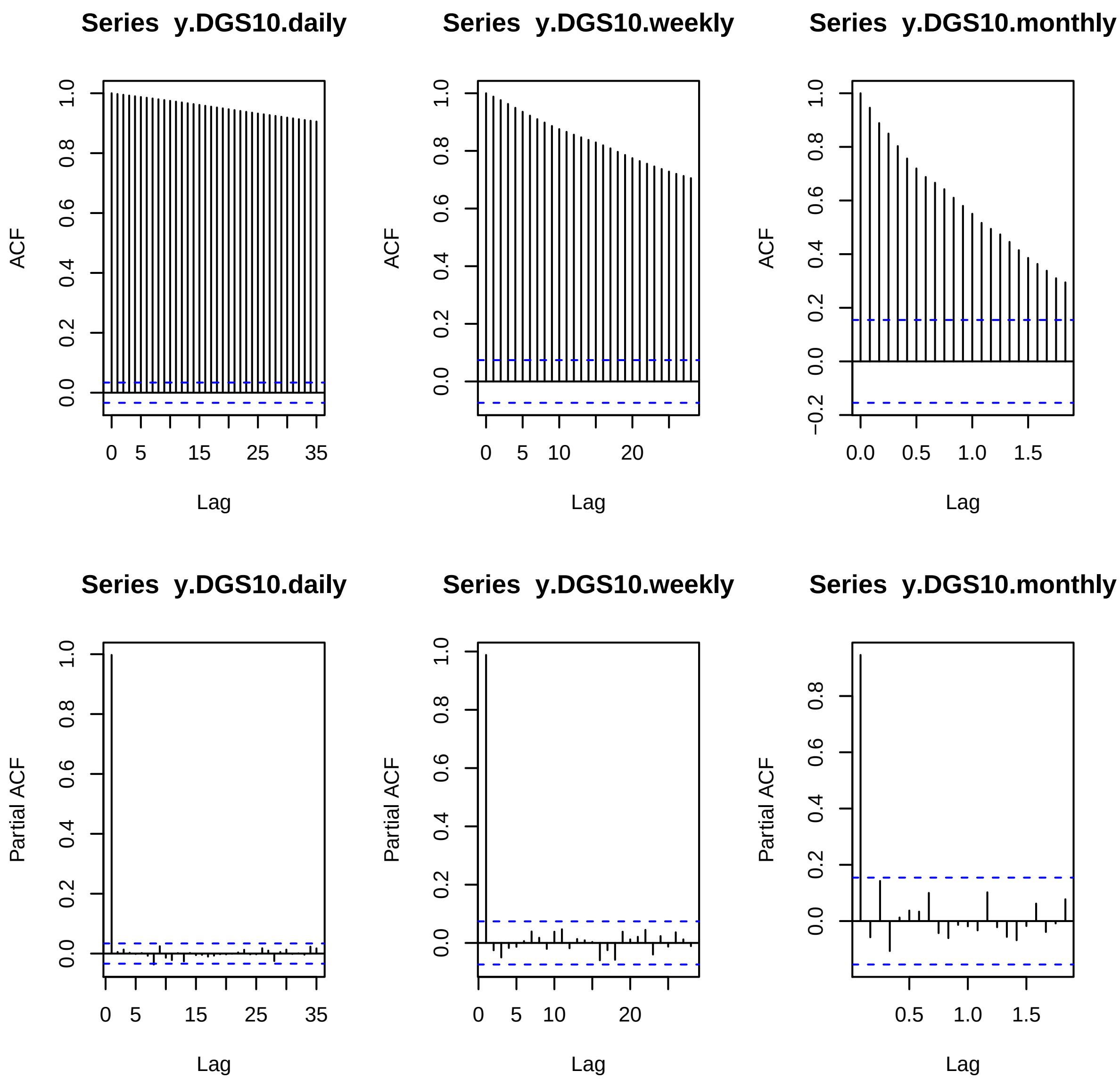
```
> chartSeries(to.monthly(y.DGS10.daily))
```



```
> # See help(chartSeries) for argument options
>
> #   The 4th column of the output from functions to.weekly() and to.monthly()
> #   is the Close corresponding to the periodicity.
>
> # Define the two vector time series of weekly close values and of monthly close values
> y.DGS10.weekly<-to.weekly(y.DGS10.daily)[,4]
> y.DGS10.monthly<-to.monthly(y.DGS10.daily)[,4]
> # Note the dimensions when daily data reduced to weekly and to monthly periods
> dim(y.DGS10.weekly)

[1] 700   1

> dim(y.DGS10.monthly)

[1] 161   1

>
```

1.3 The ACF and PACF for daily, weekly, monthly series

```
>
> # Plot the ACF (auto-correlation function) and PACF (partial auto-correlation function)
> #   for each periodicity

> par(mfcol=c(2,3))
> acf(y.DGS10.daily)
> acf(y.DGS10.daily,type="partial")
> acf(y.DGS10.weekly)
> acf(y.DGS10.weekly,type="partial")
> acf(y.DGS10.monthly)
> acf(y.DGS10.monthly,type="partial")
```




```
> # The high first-order auto-correlation suggests that the
> # time series has a unit root on every periodicity  (daily, weekly and monthly).
> #
```

1.4 Conduct Augmented Dickey-Fuller Test for Unit Roots

```
> # The function adf.test() conducts the Augmented Dickey-Fuller Test
>
```

```
> #   For each periodicity, apply the function adf.test() twice:
> #     1) to the un-differenced series (null hypothesis: input series has a unit root)
> #     2) to the first-differenced series (same null hypothesis about differenced series)
>
> #     help(adf.test) # provides references for the test
>
> #   Results for the un-differenced series:
> adf.test(y.DGS10.daily)

        Augmented Dickey-Fuller Test

data:  y.DGS10.daily
Dickey-Fuller = -3.3765, Lag order = 14, p-value = 0.05745
alternative hypothesis: stationary

> adf.test(y.DGS10.weekly)

        Augmented Dickey-Fuller Test

data:  y.DGS10.weekly
Dickey-Fuller = -3.1191, Lag order = 8, p-value = 0.1046
alternative hypothesis: stationary

> adf.test(y.DGS10.monthly)

        Augmented Dickey-Fuller Test

data:  y.DGS10.monthly
Dickey-Fuller = -2.7287, Lag order = 5, p-value = 0.2724
alternative hypothesis: stationary
```

For each periodicity, the null hypothesis of a unit root for the time series DGS10 is not rejected at the 0.05 level. The p-value for each test does not fall below standard critical values of 0.05 or 0.01.

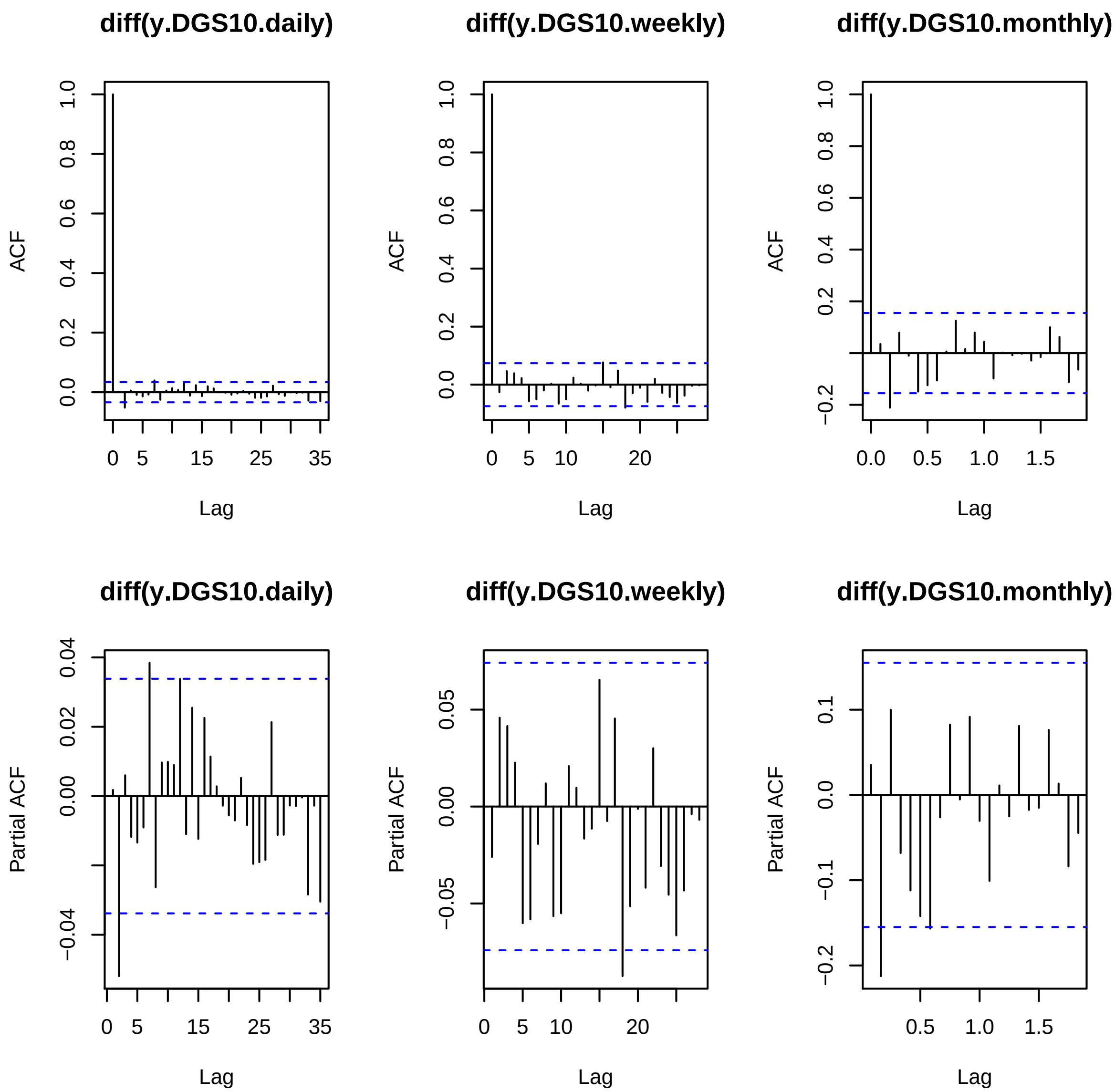
The p-value is the probability (assuming the null hypothesis is true) of realizing a test statistic as extreme as that computed for the input series. Smaller values (i.e., lower probabilities) provide stronger evidence against the null hyptohesis.

The p-value decreases as the periodicity of the data shortens. This suggests that the time-series structure in the series DGS10 may be stronger at higher frequencies.

```
> #
>
> #   Results for the first-differenced series:
> adf.test(na.omit(diff(y.DGS10.daily)))
```

```
        Augmented Dickey-Fuller Test

data:  na.omit(diff(y.DGS10.daily))
Dickey-Fuller = -14.3427, Lag order = 14, p-value = 0.01
alternative hypothesis: stationary

> adf.test(na.omit(diff(y.DGS10.weekly)))

        Augmented Dickey-Fuller Test

data:  na.omit(diff(y.DGS10.weekly))
Dickey-Fuller = -9.5629, Lag order = 8, p-value = 0.01
alternative hypothesis: stationary

> adf.test(na.omit(diff(y.DGS10.monthly)))

        Augmented Dickey-Fuller Test

data:  na.omit(diff(y.DGS10.monthly))
Dickey-Fuller = -6.6049, Lag order = 5, p-value = 0.01
alternative hypothesis: stationary

>
> # For each of the three time periodicities, the ADF test rejects the
> # null hypothesis that a unit root is present for the first-differenced series
>
```

1.5 The ACF and PACF for the differenced series of each periodicity

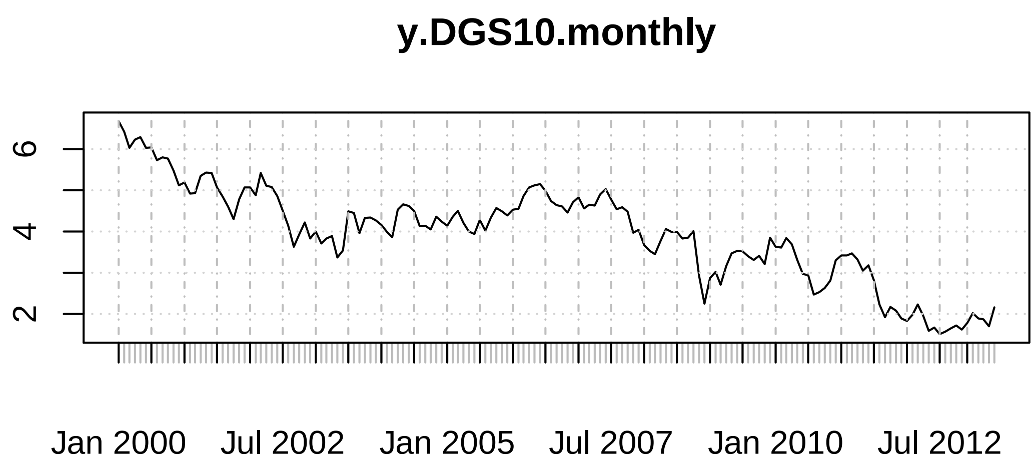


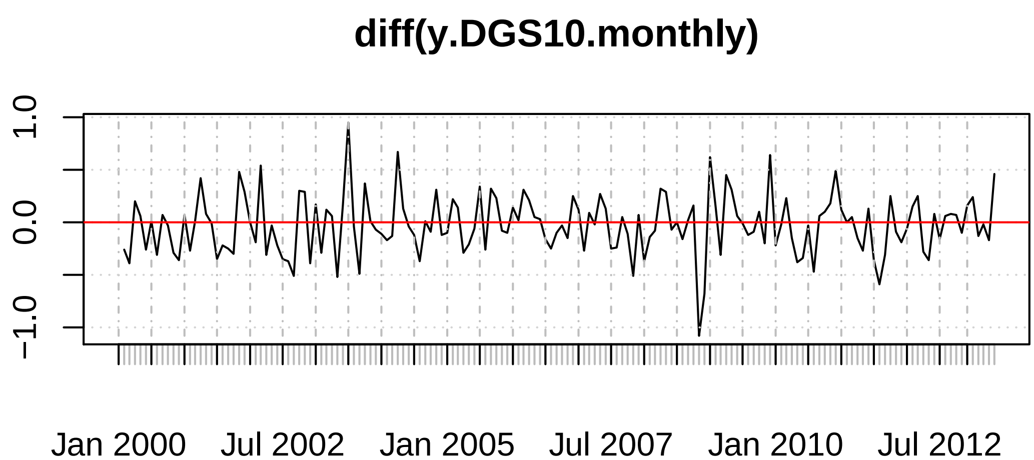

```
> # see:  help(acf)
```

```
The apparent time series structure of DGS10 varies with the periodicity:

Daily:
      strong negative order-2 autocorrelation and partial autocorrelation
      strong positive order-7 autocorrelation and partial autocorrelation
```

```
Weekly:
      weak time series structure; possible significant correlations
      at lag 15 (simple) and lag 18 (partial)

Monthly:
      strong negative order-2 autocorrelation (both simple and partial)

The autocorrelations are modestly larger as the periodicity increases
from daily to weekly to monthly.
```

```
> #length(y.DGS10.monthly)
> par(mfcol=c(2,1))
> plot(y.DGS10.monthly)
> plot(diff(y.DGS10.monthly)      )
> abline(h=0, col=2)
```



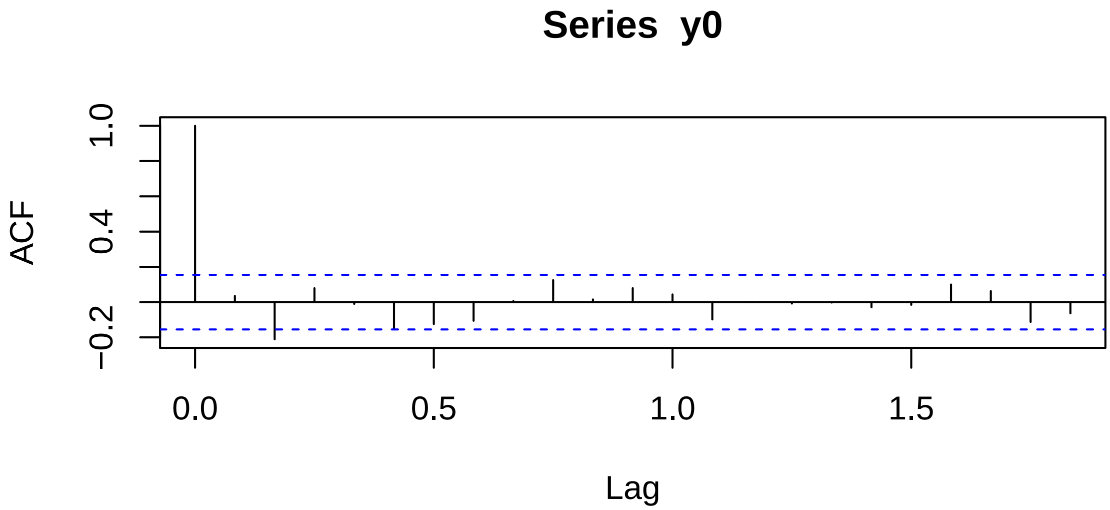


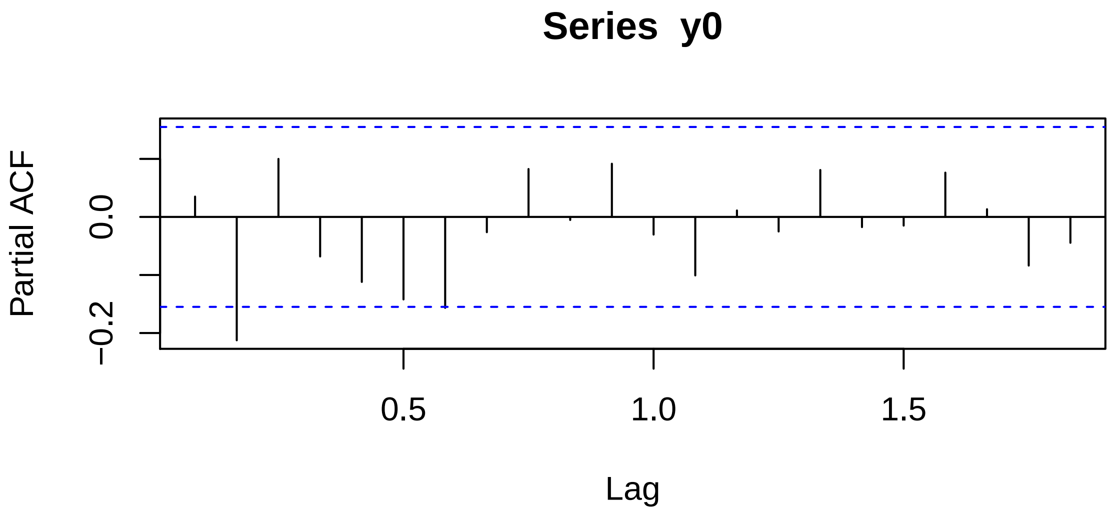
The differenced series $diff(y.DGS10.monthly)$ crosses the level 0.0 many times over the historical period. There does not appear to be a tendency for the differenced series to stay below (or above) the zero level. The series appears consistent with covariance-stationary time series structure but whether the structure is other than white noise can be evaluated by evaluating $AR(p)$ models for $p = 0, 1, 2, ...$ and determining whether an $AR(p)$ model for $p > 0$ is identified as better than an $AR(0)$, i.e., white noise.

Before fitting $AR(p)$ models we demonstrate the interpretation of partial autocorrelation coefficients.

1.6 Understanding partial autocorrelation coefficients

```
> y0<-na.omit(diff(y.DGS10.monthly) )
> # Print out the arguments of acf()
> args(acf)

function (x, lag.max = NULL, type = c("correlation", "covariance",
    "partial"), plot = TRUE, na.action = na.fail, demean = TRUE,
    ...)
NULL

> # By default, plot=TRUE, and lag.max=20.

> par(mfcol=c(2,1))
> y0.acf<-acf(y0, type="correlation")
> y0.pacf<-acf(y0, type="partial")
```






```
> # Create a table of the first 10 acf values and pacf values:
> y0.acf<-acf(y0,type="correlation", plot=FALSE,lag.max=10)
> y0.pacf<-acf(y0,type="partial", plot=FALSE,lag.max=10)
> # The output of acf() is an object with class=="acf"
> class(y0.acf)

[1] "acf"

> class(y0.pacf)
```

```
[1] "acf"

> # The length of the acf vector is 11 for the simple acf
> # and 10 for the partial acf.
> # The simple acf includes the zero-th order autocorrelation which is 1.0
>
> # Apply the function cbind() to bind together columns into a matrix
> # (use as.matrix() to coerce an n-vector into an (nx1) matrix)
>
> tab.acf_pacf<-cbind(
+   as.matrix(y0.acf$acf[-1]),
+   as.matrix(y0.pacf$acf))
> ####
> # set names of rows and columns:
> dimnames(tab.acf_pacf)<-list(c(1:10), c("acf","pacf"))
> print(tab.acf_pacf)

            acf        pacf
1   0.035097728  0.03509773
2  -0.211023180 -0.21251682
3   0.078906622  0.09997889
4  -0.010014333 -0.06810702
5  -0.148622951 -0.11198836
6  -0.124088553 -0.14212626
7  -0.105725073 -0.15669884
8   0.005975397 -0.02630317
9   0.124659317  0.08252488
10  0.015409594 -0.00527197

> # Consider the auto-regression models where
> #   y0 is the dependent variables
> #   lags of y0 are the independent variables
>
> y0.lag1<-lag(y0,k=1)
> y0.lag2<-lag(y0,k=2)
> y0.lag3<-lag(y0,k=3)
> y0.lag4<-lag(y0,k=4)
> #   The r function lm() fits the linear model by least squares
> #   The r function summary.lm() summarizes a fitted model (output from lm())
> options(show.signif.stars=FALSE)
> summary.lm(lm(y0 ~ y0.lag1))

Call:
lm(formula = y0 ~ y0.lag1)

Residuals:
     Min       1Q   Median       3Q      Max
```

```
-1.06007 -0.18752  0.00678  0.15692  0.96958

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.02567    0.02242  -1.145    0.254
y0.lag1      0.03584    0.08036   0.446    0.656

Residual standard error: 0.281 on 157 degrees of freedom
  (1 observation deleted due to missingness)
Multiple R-squared: 0.001266,        Adjusted R-squared: -0.005096
F-statistic: 0.199 on 1 and 157 DF,  p-value: 0.6562

> summary.lm(lm(y0 ~ y0.lag1 + y0.lag2))

Call:
lm(formula = y0 ~ y0.lag1 + y0.lag2)

Residuals:
     Min       1Q   Median       3Q      Max
-1.05181 -0.17109  0.01447  0.15639  0.86064

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.02996    0.02210  -1.356  0.17713
y0.lag1      0.03815    0.07880   0.484  0.62891
y0.lag2     -0.21698    0.07869  -2.757  0.00653

Residual standard error: 0.2747 on 155 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared: 0.04756,        Adjusted R-squared: 0.03527
F-statistic:  3.87 on 2 and 155 DF,  p-value: 0.0229

> summary.lm(lm(y0 ~ y0.lag1 + y0.lag2 + y0.lag3))

Call:
lm(formula = y0 ~ y0.lag1 + y0.lag2 + y0.lag3)

Residuals:
     Min       1Q   Median       3Q      Max
-1.04238 -0.17994  0.01624  0.14743  0.84592

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.02725    0.02231  -1.221  0.22384
y0.lag1      0.06667    0.08113   0.822  0.41253
y0.lag2     -0.21874    0.07890  -2.772  0.00626
y0.lag3      0.10414    0.08061   1.292  0.19835
```

```
Residual standard error: 0.2745 on 153 degrees of freedom
  (3 observations deleted due to missingness)
Multiple R-squared: 0.05687,        Adjusted R-squared: 0.03837
F-statistic: 3.075 on 3 and 153 DF,  p-value: 0.02947

> summary.lm(lm(y0 ~ y0.lag1 + y0.lag2 + y0.lag3 + y0.lag4))

Call:
lm(formula = y0 ~ y0.lag1 + y0.lag2 + y0.lag3 + y0.lag4)

Residuals:
    Min      1Q  Median      3Q     Max
-1.0401 -0.1715  0.0181  0.1565  0.8469

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.02954    0.02259  -1.308   0.1930
y0.lag1      0.07443    0.08210   0.907   0.3660
y0.lag2     -0.23466    0.08178  -2.869   0.0047
y0.lag3      0.10965    0.08125   1.350   0.1791
y0.lag4     -0.07204    0.08149  -0.884   0.3781

Residual standard error: 0.2756 on 151 degrees of freedom
  (4 observations deleted due to missingness)
Multiple R-squared: 0.06117,        Adjusted R-squared: 0.0363
F-statistic:  2.46 on 4 and 151 DF,  p-value: 0.04785

> # Compare the last regression coefficient of each model with
> # the second column (pacf) values:
>
> tab.acf_pacf[1:4,]

          acf        pacf
1  0.03509773  0.03509773
2 -0.21102318 -0.21251682
3  0.07890662  0.09997889
4 -0.01001433 -0.06810702
```

These values should be essentially equal. The small differences are due to the fact that the auto-regression model of order k conditions on the first k cases to estimate all the parameters. The acf/pacf function uses sample estimates of the auto-correlations of different orders, conditioning on j cases for order-j autocorrelations, which are different when $j < k$.

The lag-p coefficient estimate is significant for only the AR(p=2) model.

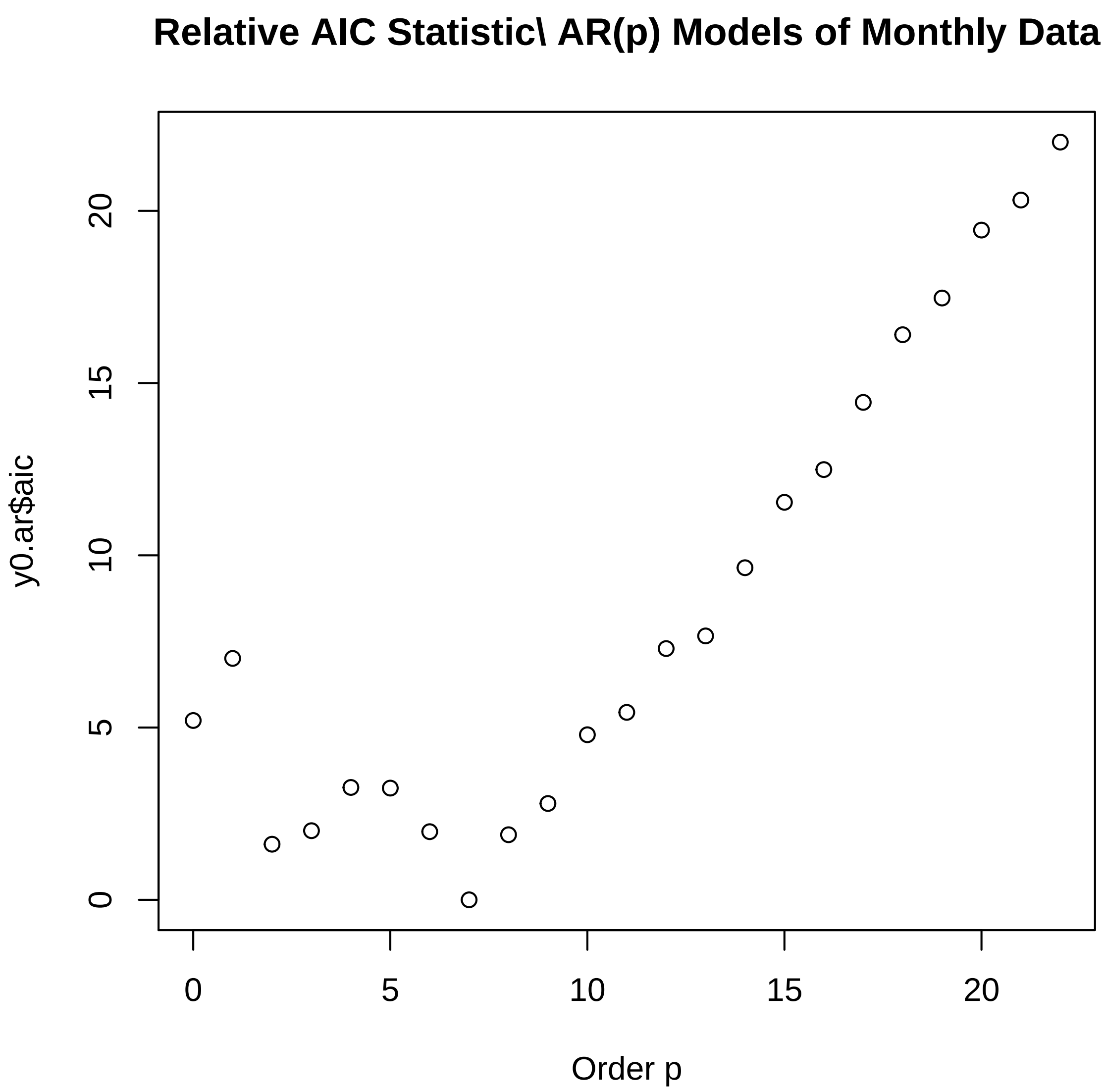
1.7 Evaluating the stationarity and cyclicality of the fitted AR(2) model to monthy data

```
> # we fit the AR(2) model and evaluate the roots of the characteristic polynomial
>
> # The AR(2) model has a p-value 0.0229 which is statistically significant
>
> lmfit0<-lm(y0 ~ y0.lag1 + y0.lag2)
> summary.lm(lmfit0)

Call:
lm(formula = y0 ~ y0.lag1 + y0.lag2)

Residuals:
     Min       1Q   Median       3Q      Max
-1.05181 -0.17109  0.01447  0.15639  0.86064

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.02996    0.02210  -1.356  0.17713
y0.lag1      0.03815    0.07880   0.484  0.62891
y0.lag2     -0.21698    0.07869  -2.757  0.00653

Residual standard error: 0.2747 on 155 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared: 0.04756,        Adjusted R-squared: 0.03527
F-statistic:  3.87 on 2 and 155 DF,  p-value: 0.0229

> lmfit0$coefficients

(Intercept)     y0.lag1     y0.lag2
-0.02995942  0.03815488 -0.21698103

> # Extract AR(2) coefficients phi1 and phi2 as named elements of
> # output list element $coefficients
> lmfit0.phi1<-lmfit0$coefficients["y0.lag1"]
> lmfit0.phi2<-lmfit0$coefficients["y0.lag2"]
> # polyroot(z) returns complex roots of polynomial with coefficients z
> char.roots<-polyroot(c(1,-1.*lmfit0.phi1, -1.*lmfit0.phi2))
> print(char.roots)

[1] 0.087922+2.144987i 0.087922-2.144987i

> print(Conj(char.roots)*char.roots )

[1] 4.608698+0i 4.608698+0i
```

```
> # These roots are complex and outside the unit circle so the fitted model is stationary
>
> # With complex roots, there is evidence of cyclicality in the series
> # The following computation computes the period as it is determined by the
> # coefficients of the characteristic polynomial.
> twopif0=acos( abs(lmfit0.phi1)/(2*sqrt(-lmfit0.phi2)))
> f0=twopif0/(8*atan(1))
> period0<-1/f0
> print(as.numeric(period0) )

[1] 4.107114

>
> # The data are consistent with cycle of period just over 4 months.
>
```

1.8 The best AR(p) model using the AIC criterion

```
> #   8.1 Apply function ar() to identify best AR(K) model by the AIC criterion ----
>
> #   see help(ar) for details of the function
> y0.ar<-ar(y0)
> # The output object is a list with named elements:
> names(y0.ar)

 [1] "order"        "ar"           "var.pred"     "x.mean"       "aic"
 [6] "n.used"       "order.max"    "partialacf"   "resid"        "method"
[11] "series"       "frequency"    "call"         "asy.var.coef"

> # The output element $order is the the AR(p) order p which has minimum AIC statistic
> y0.ar$order

[1] 7

> y0.ar$order.max

[1] 22

> y0.ar$aic

        0         1         2         3         4         5         6         7
 5.205038  7.007820  1.613412  2.006041  3.262144  3.242832  1.977763  0.000000
        8         9        10        11        12        13        14        15
 1.889265  2.795880  4.791433  5.441619  7.293497  7.660392  9.640479 11.539722
       16        17        18        19        20        21        22
12.489406 14.440227 16.404343 17.469800 19.441335 20.313834 21.995912
```




```
> # The documentation detailed in help(ar) indicates that the aic statistic
> # is the differences in AIC between each model and the best-fitting model.
>
> #   8.2 Using ar() and lm() to specify/summarize AR(p) fitted models ----
> y0.ar.7<-ar(y0, aic=FALSE, order.max=7)
> y0.ar.7
```

```
Call:
ar(x = y0, aic = FALSE, order.max = 7)

Coefficients:
      1        2        3        4        5        6        7
 0.0248  -0.2446   0.0752  -0.0774  -0.1388  -0.1348  -0.1567

Order selected 7  sigma^2 estimated as  0.07273

> # The function ar() gives coefficient estimates but does not summarize
> # the autoregression model with the regression detail of the function
> # summary.lm()
>
> # Summarize the fit the AR(7) model using lm() with lagged variables:
>
> y0.lag5<-lag(y0,k=5)
> y0.lag6<-lag(y0,k=6)
> y0.lag7<-lag(y0,k=7)
> summary.lm(lmfit0<-lm(y0 ~ y0.lag1 + y0.lag2 + y0.lag3 + y0.lag4 +
+                 y0.lag5 + y0.lag6 + y0.lag7, x=TRUE, y=TRUE))

Call:
lm(formula = y0 ~ y0.lag1 + y0.lag2 + y0.lag3 + y0.lag4 + y0.lag5 +
    y0.lag6 + y0.lag7, x = TRUE, y = TRUE)

Residuals:
     Min       1Q   Median       3Q      Max
-0.93770 -0.17136  0.02131  0.14171  0.77909

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.04381    0.02300  -1.905  0.05874
y0.lag1      0.02218    0.08241   0.269  0.78821
y0.lag2     -0.25319    0.08174  -3.098  0.00234
y0.lag3      0.07422    0.08340   0.890  0.37501
y0.lag4     -0.07659    0.08336  -0.919  0.35972
y0.lag5     -0.15302    0.08361  -1.830  0.06928
y0.lag6     -0.14731    0.08139  -1.810  0.07238
y0.lag7     -0.16589    0.08208  -2.021  0.04513

Residual standard error: 0.2703 on 145 degrees of freedom
  (7 observations deleted due to missingness)
Multiple R-squared: 0.1232,        Adjusted R-squared: 0.08092
F-statistic: 2.912 on 7 and 145 DF,  p-value: 0.007035

>
> # Note the statistical significance (p-value < .05) of the order-7 lag coefficient.
```

```
>
> #   8.3 Evaluating the stationarity of the best AR(p) model ----
> # Again, we can check the stationarity of the order-7 autoregression using
> # polyroot(z) which returns complex roots of polynomial with coefficients z
```

$$p(x) = z[1] + z[2]x + \cdots z[n]x^{n-1}$$

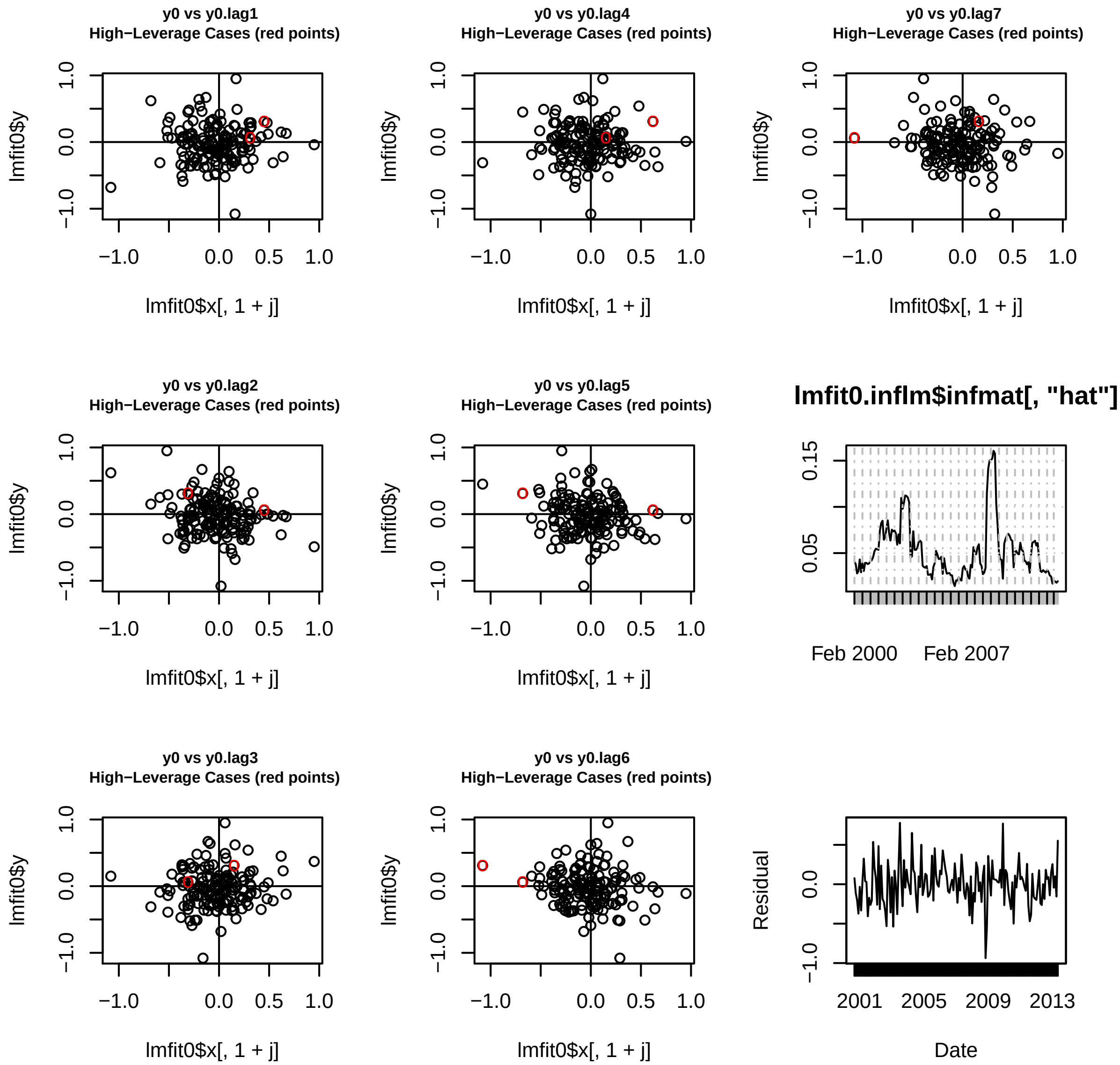
```
> char.roots.DGS10<-polyroot(c(1,-1*y0.ar$ar))
> char.roots.DGS10.modsq<-(Conj(char.roots.DGS10)*char.roots.DGS10)
> char.roots.DGS10.modsq0<- sqrt(( Re(char.roots.DGS10.modsq)) ^2 +
+                                   (Im(char.roots.DGS10.modsq))^2)
> print(char.roots.DGS10)

[1]  1.007731+0.622265i -0.771108+1.104052i -0.771108-1.104052i
[4]  1.007731-0.622265i  0.085798+1.288306i -1.504777-0.000000i
[7]  0.085798-1.288306i

> print(char.roots.DGS10.modsq0)

[1] 1.402736 1.813537 1.813537 1.402736 1.667093 2.264353 1.667093

> print(min(char.roots.DGS10.modsq0))

[1] 1.402736

> # The smallest root modulus is 1.4027 which is outside the complex unit circle.
>
>
> # 8.4 Compute/evaluate influence measures / case-deletion statistics ----
>
>   # Compute lmfit0, the fit of the AR(7) model and print out its summary
>
> summary.lm(lmfit0<-lm(y0 ~ y0.lag1 + y0.lag2 + y0.lag3 + y0.lag4 +
+                       y0.lag5 + y0.lag6 + y0.lag7,
+                       x=TRUE, y=TRUE, weights=1*(is.na(y0.lag7)==FALSE)))

Call:
lm(formula = y0 ~ y0.lag1 + y0.lag2 + y0.lag3 + y0.lag4 + y0.lag5 +
    y0.lag6 + y0.lag7, weights = 1 * (is.na(y0.lag7) == FALSE),
    x = TRUE, y = TRUE)

Residuals:
     Min       1Q   Median       3Q      Max
-0.93770 -0.17136  0.02131  0.14171  0.77909

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
```

```
(Intercept) -0.04381    0.02300  -1.905  0.05874
y0.lag1      0.02218    0.08241   0.269  0.78821
y0.lag2     -0.25319    0.08174  -3.098  0.00234
y0.lag3      0.07422    0.08340   0.890  0.37501
y0.lag4     -0.07659    0.08336  -0.919  0.35972
y0.lag5     -0.15302    0.08361  -1.830  0.06928
y0.lag6     -0.14731    0.08139  -1.810  0.07238
y0.lag7     -0.16589    0.08208  -2.021  0.04513

Residual standard error: 0.2703 on 145 degrees of freedom
  (7 observations deleted due to missingness)
Multiple R-squared: 0.1232,        Adjusted R-squared: 0.08092
F-statistic: 2.912 on 7 and 145 DF,  p-value: 0.007035

> # The input arguments x=TRUE, y=TRUE result in output list elements that
> # are adjusted by eliminating rows with  0-valued weights
> names(lmfit0)

 [1] "coefficients"  "residuals"     "fitted.values" "effects"
 [5] "weights"       "rank"          "assign"        "qr"
 [9] "df.residual"   "na.action"     "xlevels"       "call"
[13] "terms"         "model"         "x"             "y"

> dim(lmfit0$x)

[1] 153   8

> dim(lmfit0$y)

NULL

> length(y0)

[1] 160

> # Compute influence measures (case-deletion statistics) of the fitted model
>
> lmfit0.inflm<-influence.measures(lmfit0)
> names(lmfit0.inflm)

[1] "infmat" "is.inf" "call"

> # Show the dimensions and first rows of the 12-column output list element $infmat
> dim(lmfit0.inflm$infmat)

[1] 153  12

> head(lmfit0.inflm$infmat)
```

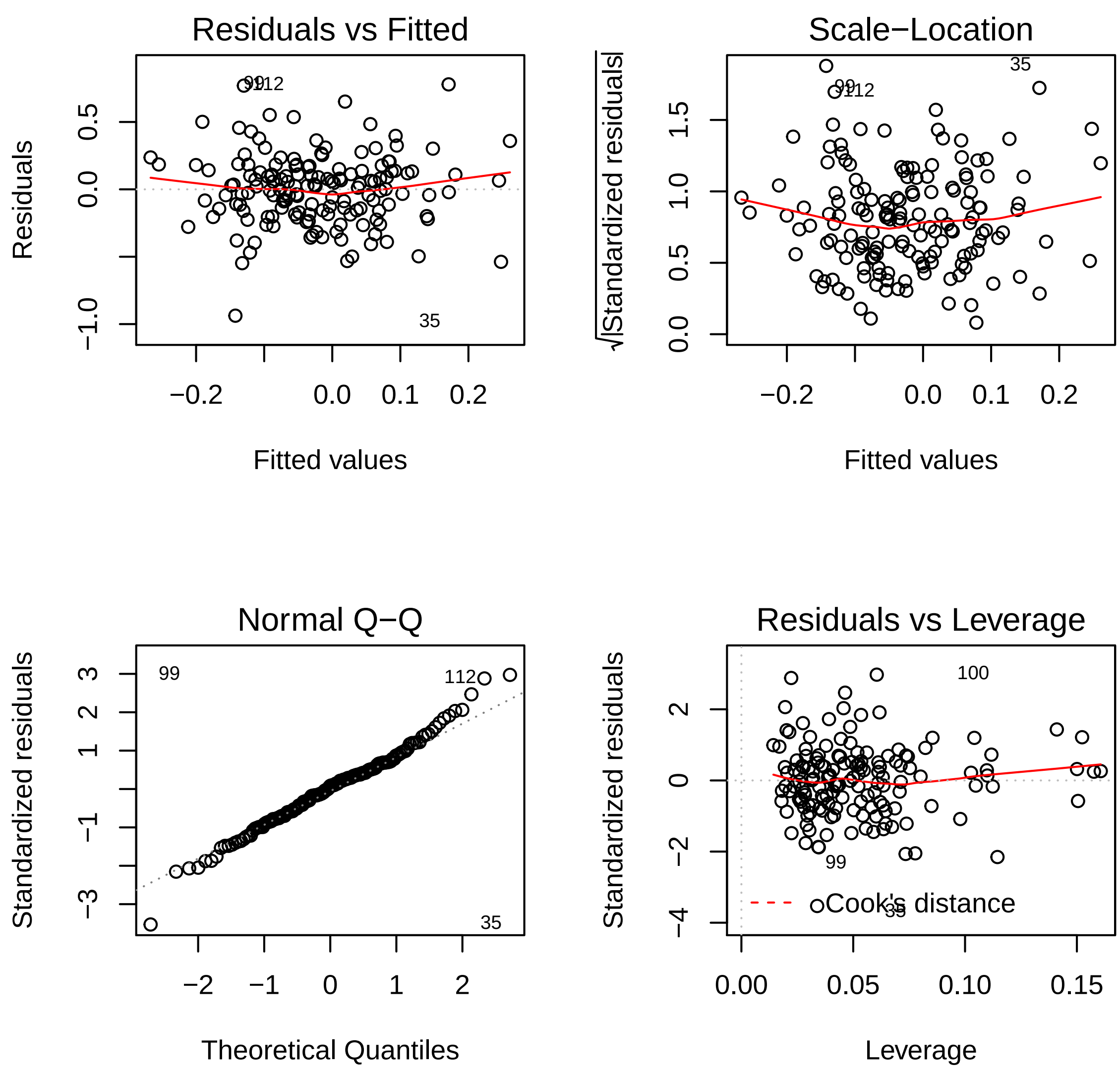
```
               dfb.1_    dfb.y0.1     dfb.y0.2     dfb.y0.3     dfb.y0.4
Feb 2000  0.012771167 -0.03552904  0.001996455 -0.021923949  0.005963799
Mar 2000 -0.026467192 -0.01681370  0.049272013 -0.010171051  0.030827019
Apr 2000 -0.057148073  0.01352185 -0.042691741  0.080183007 -0.027640297
May 2000 -0.081919426  0.09489009  0.030292449 -0.050163698  0.147613118
Jun 2000 -0.004322787  0.01510187  0.008397776  0.006559631 -0.005435013
Jul 2000 -0.069046679 -0.01698180  0.130693058  0.060647592  0.051796152
             dfb.y0.5     dfb.y0.6    dfb.y0.7       dffit     cov.r
Feb 2000  0.008968353 -0.031217183 -0.02024820  0.06000838 1.0968777
Mar 2000 -0.008064587 -0.023095694  0.04426137 -0.08458675 1.0878636
Apr 2000  0.064200046 -0.032426934 -0.03429039 -0.12834827 1.0534402
May 2000 -0.047664397  0.147065812 -0.04983750 -0.24900093 0.9773037
Jun 2000  0.014364254 -0.001720396  0.01501715 -0.02652188 1.1036187
Jul 2000 -0.040307118  0.105724000 -0.01243697 -0.21734606 0.9983007
               cook.d        hat
Feb 2000 4.529862e-04 0.04092938
Mar 2000 8.994483e-04 0.03812787
Apr 2000 2.065259e-03 0.02800237
May 2000 7.698226e-03 0.03038337
Jun 2000 8.852726e-05 0.04313951
Jul 2000 5.881815e-03 0.02921991

> # Show the dimensions and first rows of the 12-column output list element $is.inf
> dim(lmfit0.inflm$is.inf)

[1] 153  12

> head(lmfit0.inflm$is.inf)

         dfb.1_ dfb.y0.1 dfb.y0.2 dfb.y0.3 dfb.y0.4 dfb.y0.5 dfb.y0.6 dfb.y0.7
Feb 2000  FALSE    FALSE    FALSE    FALSE    FALSE    FALSE    FALSE    FALSE
Mar 2000  FALSE    FALSE    FALSE    FALSE    FALSE    FALSE    FALSE    FALSE
Apr 2000  FALSE    FALSE    FALSE    FALSE    FALSE    FALSE    FALSE    FALSE
May 2000  FALSE    FALSE    FALSE    FALSE    FALSE    FALSE    FALSE    FALSE
Jun 2000  FALSE    FALSE    FALSE    FALSE    FALSE    FALSE    FALSE    FALSE
Jul 2000  FALSE    FALSE    FALSE    FALSE    FALSE    FALSE    FALSE    FALSE
         dffit cov.r cook.d   hat
Feb 2000 FALSE FALSE  FALSE FALSE
Mar 2000 FALSE FALSE  FALSE FALSE
Apr 2000 FALSE FALSE  FALSE FALSE
May 2000 FALSE FALSE  FALSE FALSE
Jun 2000 FALSE FALSE  FALSE FALSE
Jul 2000 FALSE FALSE  FALSE FALSE

> # The $is.inf elements are TRUE if the magnitude of the respective influence
> # measure in $infmat exceeds nominal cutoffs; see help(influence.measures)
> #
```

```
> # Count of influential cases by column/influence-measure
> apply(lmfit0.inflm$is.inf,2,sum)

  dfb.1_ dfb.y0.1 dfb.y0.2 dfb.y0.3 dfb.y0.4 dfb.y0.5 dfb.y0.6 dfb.y0.7
       0        0        0        0        0        0        0        0
   dffit    cov.r   cook.d      hat
       2       13        0        2

> # Table counts of influential/non-influential cases
> # as measured by the hat/leverage statistic.
> table(lmfit0.inflm$is.inf[,"hat"])

FALSE  TRUE
  151     2

> # Plot dependent variable vs each independent variable
> #     and selectively highlight influential cases
> #     (Use output elements lmfit0$x and lmfit0$y rather than the input argument variables)
> head(lmfit0$x)

   (Intercept) y0.lag1 y0.lag2 y0.lag3 y0.lag4 y0.lag5 y0.lag6 y0.lag7
8            1   -0.31    0.01   -0.26    0.06    0.20   -0.39   -0.26
9            1    0.07   -0.31    0.01   -0.26    0.06    0.20   -0.39
10           1   -0.03    0.07   -0.31    0.01   -0.26    0.06    0.20
11           1   -0.29   -0.03    0.07   -0.31    0.01   -0.26    0.06
12           1   -0.36   -0.29   -0.03    0.07   -0.31    0.01   -0.26
13           1    0.07   -0.36   -0.29   -0.03    0.07   -0.31    0.01

> par(mfcol=c(3,3))
> for (j in c(1:7)){
+   plot(lmfit0$x[,1+j], lmfit0$y,
+      main=paste("y0 vs y0.lag",as.character(j)," \n High-Leverage Cases (red points)",sep=
+        cex.main=0.8)
+ abline(h=0,v=0)
+ #abline(lmfit0, col=3, lwd=3)
+
+ # Plot cases with high leverage as red (col=2) "o"s
+ index.inf.hat<-which(lmfit0.inflm$is.inf[,"hat"]==TRUE)
+ points(lmfit0$x[index.inf.hat,j+1], lmfit0$y[index.inf.hat],
+        col=2, pch="o")
+ }
> # Plot leverage of cases (diagonals of hat matrix)
> plot(lmfit0.inflm$infmat[,"hat"])
> print(time(lmfit0.inflm$infmat)[index.inf.hat])

[1] "Oct 2008" "Nov 2008"
```

```
> # Note the 2 cases are time points in the heart of the financial crisis of 2008.
>
> # Time series plot of residuals, applying the function zoo() to create
> # the residual time series object (the $residuals element of the lm() output
> # has data and time attributes that are not the same length due to
> # incorrect handling of missing data/rows
> names(lmfit0)

 [1] "coefficients"  "residuals"     "fitted.values" "effects"
 [5] "weights"       "rank"          "assign"        "qr"
 [9] "df.residual"   "na.action"     "xlevels"       "call"
[13] "terms"         "model"         "x"             "y"

> length(lmfit0$residuals)

[1] 153

> length(time(lmfit0$residuals))

[1] 160

> length(coredata(lmfit0$residuals))

[1] 153

> lmfit0$residuals<-zoo(as.numeric(lmfit0$residuals), order.by=time(lmfit0$residuals)[-(1:7)
> plot(lmfit0$residuals,
+       ylab="Residual", xlab="Date")
```


y0 vs y0.lag1
High−Leverage Cases (red points)
y0 vs y0.lag4
High−Leverage Cases (red points)
y0 vs y0.lag7
High−Leverage Cases (red points)
y0 vs y0.lag2
High−Leverage Cases (red points)
y0 vs y0.lag5
High−Leverage Cases (red points)
lmfit0.inflm$infmat[, "hat"]
y0 vs y0.lag3
High−Leverage Cases (red points)
y0 vs y0.lag6
High−Leverage Cases (red points)
lmfit0$y
lmfit0$x[, 1 + j]
−1.0
0.0
0.5
1.0
0.05
0.15
Feb 2000
Feb 2007
Residual
2001
2005
2009
2013
Date

```
> #The R function $plot.lm()$ generates a useful 2x2 display
> # of plots for various regression diagnostic statistics:
>
> layout(matrix(c(1,2,3,4),2,2)) # optional 4 graphs/page
> plot(lmfit0)
>
```



- The top left panel plots the residuals vs fitted values. This plot should show no correlation between the model error/residual and the magnitude of the fitted value. A smoothed estimate of the residual is graphed in red to compare with the zero-line.

- The top right panel plots the absolute standardized residual vs the fitted values. This plot is useful to identify heteroscedasticity (unequal residual variances/standard deviations) which might vary with the magnitude of the fitted value. There is some curvature in the smoothed estimate of the absolute residual standard deviation suggesting that cases toward the extremes (high/low) of fitted values have higher standard deviations.

- The bottom left panel is the q-q plot of the standardized residuals. The ordered, standardized residuals are plotted on the vertical axis against the expected ordered values from a sample from a $N(0, 1)$ distribution. If the standardized residuals were normally distributed the points would closely follow a line with intercept equal to the mean/expected value and slope equal to the standard deviation (square-root of the variance) of the residual distribution. Note that the regression model residuals have unequal variances (they are proportional to $[1 - H_{i,i}]$ where H is the hat matrix). Standardizing the residuals makes them have equal variances, equal to the error variance in the model. See the R function $qqnorm()$ for more details about the q-q plot

- The bottom right panel is the residual vs leverage (diagonals of the hat matrix) plot. This plot highlights the influence of high-leverage cases in pulling the linear regression model toward cases. in terms of making the residual small.