Handling Time Series in R

This version: 29 November 2018

Notes for Intermediate Econometrics / Time Series Analysis and Forecasting
Anthony Tay

Base R has a time-series object class ("**ts**"). The most fundamental way to create a time series is to pass a vector of numerics to the **ts** function, specifying the frequency, and the start year and month/quarter. As an example, we create a simple time series object, quarterly frequency, starting 2008Q2, containing the sequence 1 to 20.

```
## Convert vector 1:20 to a quarterly series starting in 2008Q2
q.ts <- ts(1:20, start=c(2008,2), frequency=4)
q.ts
```

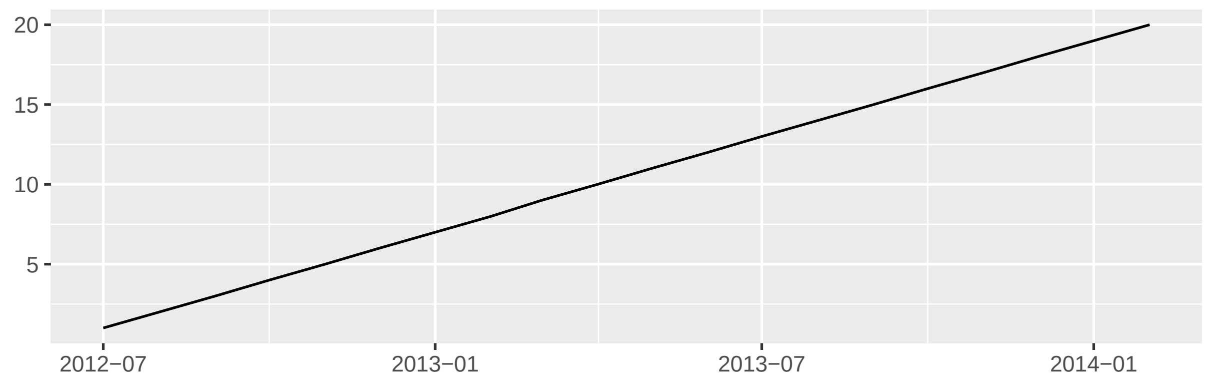
```
##      Qtr1 Qtr2 Qtr3 Qtr4
## 2008         1    2    3
## 2009    4    5    6    7
## 2010    8    9   10   11
## 2011   12   13   14   15
## 2012   16   17   18   19
## 2013   20
```

```
## Convert vector 1:20 to a monthly series starting in 2012M7
m.ts <- ts(1:20, start=c(2012,7), frequency=12)
m.ts
```

```
##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 2012                           1   2   3   4   5   6
## 2013   7   8   9  10  11  12  13  14  15  16  17  18
## 2014  19  20
```

Many functions recognize the "ts" class, and adapt accordingly.

```
library(tidyverse)
library(ggfortify)
autoplot(m.ts)
```



You can create a multiple time series ("**mts**") object, by feeding in a matrix to the ts function.

```
## Convert vector 1:36 to three monthly series, each starting in 2015M3
X.mts <- ts(matrix(1:36, ncol=3), start=c(2015,3), frequency=12)
colnames(X.mts) <- c("X1", "X2", "X3")
X.mts
```

```
##          X1 X2 X3
## Mar 2015  1 13 25
## Apr 2015  2 14 26
## May 2015  3 15 27
## Jun 2015  4 16 28
## Jul 2015  5 17 29
## Aug 2015  6 18 30
## Sep 2015  7 19 31
## Oct 2015  8 20 32
## Nov 2015  9 21 33
## Dec 2015 10 22 34
## Jan 2016 11 23 35
## Feb 2016 12 24 36
```

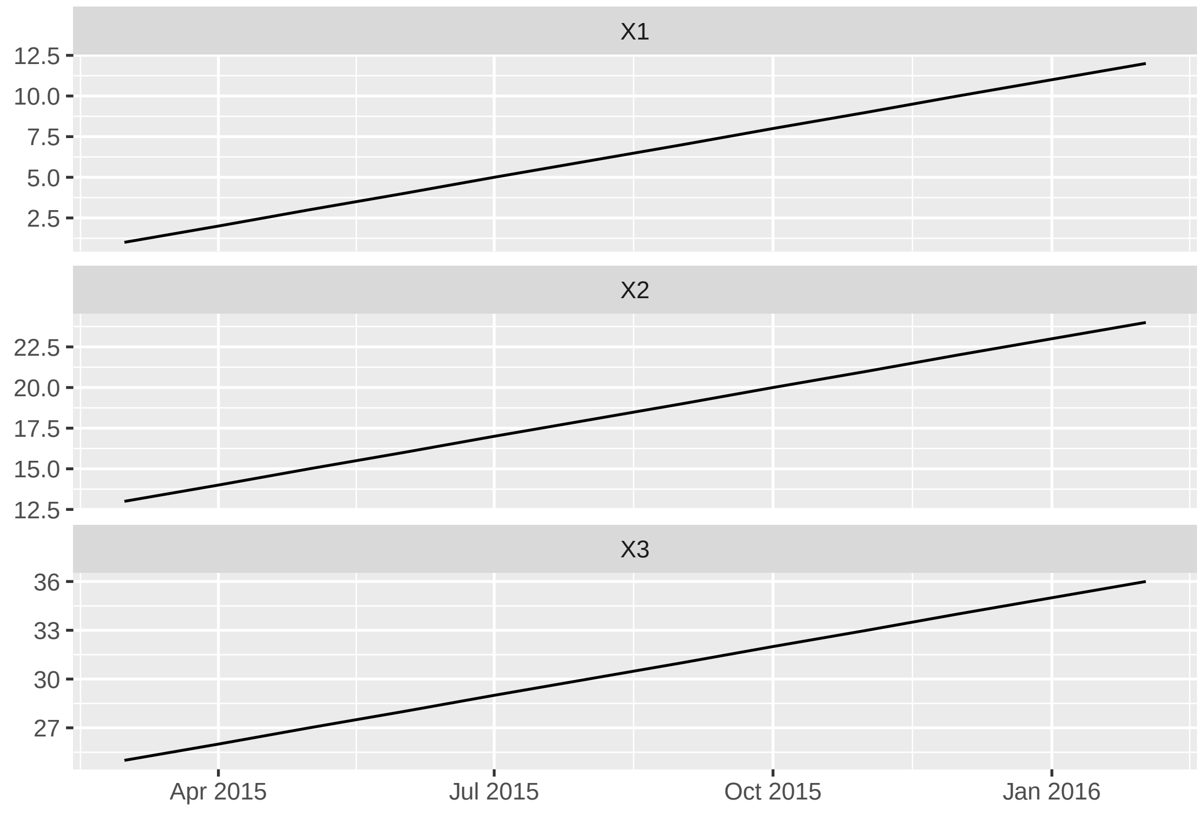
To pull out a single time series from an mts object, use the usual indexing procedures for matrices:

```
X.mts[,"X2"]
```

```
##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 2015          13  14  15  16  17  18  19  20  21  22
## 2016  23  24
```

You can use autoplot on an mts series.

```
autoplot(X.mts)
```



You can get sub-sample of time series object (ts or mts)

```
window(X.mts, start=c(2015,3), end=c(2015,7))
```

```
##          X1 X2 X3
## Mar 2015  1 13 25
## Apr 2015  2 14 26
## May 2015  3 15 27
## Jun 2015  4 16 28
## Jul 2015  5 17 29
```

```
window(X.mts, start=c(2015,5), end=c(2015,5)) <- 0
window(X.mts, start=c(2015,3), end=c(2015,7))
```

```
##          X1 X2 X3
## Mar 2015  1 13 25
## Apr 2015  2 14 26
## May 2015  0  0  0
## Jun 2015  4 16 28
## Jul 2015  5 17 29
```

More sophisticated manipulations, such as selecting only the January observations, are awkward to carry out in base R. We will frequently use other packages and the alternative time series classes they offer to ease these manipulations. Unsurprisingly, a specialized class for working with dates is useful for working with time series.

R date class

R has a **date** object, which can be created from a string using the **ymd** and related functions from the **lubridate** package. The package **xts** will be helpful for working with monthly and quarterly data, and we load it here at the outset.

```
library(lubridate)
library(xts)
d <- ymd("20070210")
d
```

```
## [1] "2007-02-10"
```

```
class(d) # ymd turns a "year month day" string into a "date" object
```

```
## [1] "Date"
```

Multiple formats are possible

```
ymd("1999 Sep 13")
```

```
## [1] "1999-09-13"
```

Other similar functions: **ydm, mdy, myd, dmy, yq**

```
mdy("01162011")
```

```
## [1] "2011-01-16"
```

```
dmy("23 July 2003")
```

```
## [1] "2003-07-23"
```

```
yq("2013-2")
```

```
## [1] "2013-04-01"
```

Some useful tricks with date objects

```
d <- ymd("20070210")
d+4
```

```
## [1] "2007-02-14"
```

```
seq(d, d+8, by="2 days")
```

```
## [1] "2007-02-10" "2007-02-12" "2007-02-14" "2007-02-16" "2007-02-18"
```

```r
seq(d, d+100, by="1 month")
```

```
## [1] "2007-02-10" "2007-03-10" "2007-04-10" "2007-05-10"
```

```r
dmy("3rd July 2008")
```

```
## [1] "2008-07-03"
```

```r
mdy("December 12, 2002")
```

```
## [1] "2002-12-12"
```

You can convert date objects to character using **as.character()**.

```r
d
```

```
## [1] "2007-02-10"
```

```r
class(d)
```

```
## [1] "Date"
```

```r
as.character(d)
```

```
## [1] "2007-02-10"
```

```r
class(as.character(d))
```

```
## [1] "character"
```

You will often wante to specify the format of dates. R uses the following codes:

- **%d** - day as two digits,
- **%a** - abbreviated weekday (eg. Wed),
- **%A** - unabbreviated weekday (eg. Monday),
- **%m** - month as two digits,
- **%b** - abbreviated month,
- **%B** - unabbreviated month,
- **%y** - two digit year,
- **%Y** - four digit year.
- **%V** - Week of the year from 01 to 53 as defined in ISO 8601. Monday is first day of the week. The week with the first Thursday of the year is Week 1, with previous week as last week of previous year,
- **%U** - Week of the year from 00 to 53 according to US convention. Sunday is the first day of the week (and typically the first Sunday of the year is day 1 of week 1).

```r
format(dmy("3rd July 2008"), format="%B %d, %Y")
```

```
## [1] "July 03, 2008"
```

```r
d <- ymd("20070101")
d <- seq(d, d+10, by="day")
```

```r
format(d, format="%Y week %U")  # Note 2007/01/01 was a monday
```

```
##  [1] "2007 week 00" "2007 week 00" "2007 week 00" "2007 week 00"
##  [5] "2007 week 00" "2007 week 00" "2007 week 01" "2007 week 01"
##  [9] "2007 week 01" "2007 week 01" "2007 week 01"
```

To work with monthly and quarterly data, the **yearmon** and **yearqtr** class of dates is useful. These are from the **zoo** package, included in **xts**. Use **as.yearmon** amd **as.yearqtr** to convert dates and datestrings to yearmon/yearqtr.

```r
d <- ymd("20070210")
as.yearmon(d)
```

```
## [1] "Feb 2007"
```

```r
as.yearqtr(d)
```

```
## [1] "2007 Q1"
```

```r
class(as.yearqtr(d))  ## This is zoo's yearqtr class, not same as R date
```

```
## [1] "yearqtr"
```

```r
as.yearmon("Jan-83", "%B-%y")
```

```
## [1] "Jan 1983"
```

```r
as.yearqtr("2008Q1")
```

```
## [1] "2008 Q1"
```

```r
as.yearqtr("2008:1", "%Y:%q") # Note the %q code
```

```
## [1] "2008 Q1"
```

You can convert back to character with **as.character()**. Here we use **gsub()** to get rid of the space between the year and the “Q1”.

```r
gsub(" ", "", as.yearqtr(d))
```

```
## [1] "2007Q1"
```

```r
class(gsub(" ", "", as.yearqtr(d))) # removes space, coerces to character
```

```
## [1] "character"
```

R also has a date-time object which we won’t cover here. For daily data, there is another consideration – what if your data is five-day week format (no weekends?) Or five-day week and holidays omitted? We ignore these issues here, except to mention that the package **xts** can deal with many of these situations.

Moving from file to xts, and between ts and xts

We use the dataseries in timeseries_monthly.csv to illustrate importing time series from csv to R, and moving between ts and “xts” type of objects. The ‘xts’ class is a time series class from the xts package, and quite a bit more maneuverable than the ts class.

```
df_m <- read_csv("timeseries_monthly.csv")
glimpse(df_m)
```

```
## Observations: 420
## Variables: 6
## $ DATE        <chr> "Jan-83", "Feb-83", "Mar-83", "Apr-83", "May-83", ...
## $ ELEC_GEN_SG <dbl> 667.3, 586.9, 727.4, 719.0, 727.1, 728.4, 741.1, 7...
## $ TOUR_SG     <int> 232164, 212591, 242272, 226610, 236720, 227843, 24...
## $ IP_SG       <dbl> 14.34, 11.37, 14.50, 12.86, 13.01, 12.62, 13.56, 1...
## $ CPI_US      <dbl> 97.8, 97.9, 97.9, 98.6, 99.2, 99.5, 99.9, 100.2, 1...
## $ DOMEX5_SG   <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA...
```

We can convert the whole data_frame to xts object. To create xts objects, we use the xts() function, and supply the vector/matrix/dataframe to convert, and an appropriate date series. In our case, the data is monthly frequency, and available in df_m, but as character string in %b-%y format. To feed into the xts() function, we convert it to yearmon class.

```
df_m.xts <- xts(select(df_m, -DATE), order.by = as.yearmon(df_m$DATE, "%b-%y"))
head(df_m.xts, 3)
```

```
##          ELEC_GEN_SG TOUR_SG IP_SG CPI_US DOMEX5_SG
## Jan 1983       667.3  232164 14.34   97.8        NA
## Feb 1983       586.9  212591 11.37   97.9        NA
## Mar 1983       727.4  242272 14.50   97.9        NA
```

```
tail(df_m.xts, 3)
```

```
##          ELEC_GEN_SG TOUR_SG  IP_SG  CPI_US DOMEX5_SG
## Oct 2017      4504.5 1402299 117.82 246.663    3565.2
## Nov 2017      4283.3 1397330 115.19 246.669    3581.6
## Dec 2017      4376.5 1569616 120.50 246.524    3682.2
```

```
class(df_m.xts)
```

```
## [1] "xts" "zoo"
```

You can refer to specific subsets

```
df_m.xts["2006-05"] # May 2006
```

```
##          ELEC_GEN_SG TOUR_SG IP_SG CPI_US DOMEX5_SG
## May 2006      3403.8  766181 66.39  202.5    4434.6
```

```
df_m.xts["2006-01/2006-05"] # From Jan 2006 to May 2006
```

```
##          ELEC_GEN_SG TOUR_SG IP_SG CPI_US DOMEX5_SG
## Jan 2006      3096.0  767908 63.10  198.3    4364.6
## Feb 2006      2947.5  728158 64.82  198.7    4520.7
## Mar 2006      3407.6  819985 77.60  199.8    5079.6
## Apr 2006      3193.2  816654 62.24  201.5    4437.6
## May 2006      3403.8  766181 66.39  202.5    4434.6
```

```
temp <- df_m.xts["2006"]  # all 2006
head(temp,4)
```

```
##          ELEC_GEN_SG TOUR_SG IP_SG CPI_US DOMEX5_SG
## Jan 2006      3096.0  767908 63.10  198.3    4364.6
## Feb 2006      2947.5  728158 64.82  198.7    4520.7
## Mar 2006      3407.6  819985 77.60  199.8    5079.6
## Apr 2006      3193.2  816654 62.24  201.5    4437.6
```

```
tail(temp,4)
```

```
##          ELEC_GEN_SG TOUR_SG IP_SG CPI_US DOMEX5_SG
## Sep 2006      3281.0  715527 80.73  202.9    4889.3
## Oct 2006      3439.7  857821 78.46  201.8    5103.2
## Nov 2006      3303.9  800732 82.14  201.5    4783.9
## Dec 2006      3271.6  903578 82.54  201.8    4618.1
```

```
temp <- df_m.xts[month(df_m.xts)==2] # All February data
head(temp,4)                 #   The month() functions is
```

```
##          ELEC_GEN_SG TOUR_SG IP_SG CPI_US DOMEX5_SG
## Feb 1983       586.9  212591 11.37   97.9        NA
## Feb 1984       669.4  221968 13.17  102.4        NA
## Feb 1985       725.3  230830 12.54  106.0        NA
## Feb 1986       727.4  236979 11.68  109.3        NA
```

```
tail(temp,4)                 # from the lubridate package
```

```
##          ELEC_GEN_SG TOUR_SG IP_SG  CPI_US DOMEX5_SG
## Feb 2014      3667.9 1238239 94.92 234.781    2890.6
## Feb 2015      3666.9 1188818 91.81 234.722    2502.9
## Feb 2016      3849.2 1335099 88.73 237.111    2670.6
## Feb 2017      3892.2 1361010 97.71 243.603    3213.7
```

We create a dummy variable for March:

```
df_m.xts$d1 <- 0
df_m.xts$d1[month(df_m.xts)==3] <- 1
df_m.xts["2006/2007"]
```

```
##          ELEC_GEN_SG TOUR_SG IP_SG  CPI_US DOMEX5_SG d1
## Jan 2006      3096.0  767908 63.10 198.300    4364.6  0
## Feb 2006      2947.5  728158 64.82 198.700    4520.7  0
## Mar 2006      3407.6  819985 77.60 199.800    5079.6  1
## Apr 2006      3193.2  816654 62.24 201.500    4437.6  0
## May 2006      3403.8  766181 66.39 202.500    4434.6  0
## Jun 2006      3272.1  788955 84.10 202.900    4536.4  0
## Jul 2006      3420.1  914879 75.01 203.500    4691.2  0
## Aug 2006      3405.5  870763 70.78 203.900    4498.3  0
## Sep 2006      3281.0  715527 80.73 202.900    4889.3  0
## Oct 2006      3439.7  857821 78.46 201.800    5103.2  0
## Nov 2006      3303.9  800732 82.14 201.500    4783.9  0
## Dec 2006      3271.6  903578 82.54 201.800    4618.1  0
## Jan 2007      3338.3  825919 72.90 202.416    4630.4  0
## Feb 2007      3005.3  758341 68.27 203.499    4000.7  0
## Mar 2007      3490.3  858661 77.60 205.352    4668.1  1
## Apr 2007      3397.3  819505 75.95 206.686    4411.5  0
## May 2007      3554.3  821248 77.44 207.949    4252.4  0
## Jun 2007      3457.0  850514 75.17 208.352    4514.7  0
## Jul 2007      3517.3  952836 93.67 208.299    4735.4  0
## Aug 2007      3523.8  913181 81.20 207.917    4843.9  0
## Sep 2007      3443.4  771453 78.30 208.490    4646.0  0
## Oct 2007      3626.0  917427 81.13 208.936    4994.8  0
## Nov 2007      3402.3  841648 79.79 210.177    4460.4  0
## Dec 2007      3382.4  953812 79.24 210.036    4305.8  0
```

Working with quarterly data is similar, though there are some things to note when subsetting.

```
df_q <- read_csv("timeseries_quarterly.csv")
glimpse(df_q)
```

```
## Observations: 128
## Variables: 3
## $ Period <chr> "1980Q1", "1980Q2", "1980Q3", "1980Q4", "1981Q1", "1981...
## $ Y      <dbl> 2.665478, 3.406533, 3.486567, 3.842664, 3.627703, 3.354...
## $ Y1     <dbl> 2.665478, 3.406533, 3.486567, 3.842664, 3.627703, 3.354...
```

```
df_q.xts <- xts(select(df_q, -Period), order.by = as.yearqtr(df_q$Period))
head(df_q.xts,3)
```

```
##                Y       Y1
## 1980 Q1 2.665478 2.665478
## 1980 Q2 3.406533 3.406533
## 1980 Q3 3.486567 3.486567
```

```
tail(df_q.xts,3)
```

```
##                Y       Y1
## 2011 Q2 4.148819 4.424495
## 2011 Q3 5.086940 8.176982
## 2011 Q4 4.466191 5.693984
```

```
df_q.xts["1985-01"]
```

```
##                Y       Y1
## 1985 Q1 3.297786 3.297786
```

```
df_q.xts["1985-02"]
```

```
##      Y Y1
```

```
df_q.xts["1985-03"]
```

```
##      Y Y1
```

```
df_q.xts["1985-04"]
```

```
##                Y       Y1
## 1985 Q2 3.508998 3.508998
```

Quarters are referenced by months 1, 4, 7, and 10.

```
df_q.xts["1985-01/1986-04"]
```

```
##                Y       Y1
## 1985 Q1 3.297786 3.297786
## 1985 Q2 3.508998 3.508998
## 1985 Q3 3.898615 3.898615
## 1985 Q4 3.453972 3.453972
## 1986 Q1 4.091660 4.091660
## 1986 Q2 4.626595 4.626595
```

```
df_q.xts["1985-02/1986-05"]
```

```
##                Y       Y1
## 1985 Q2 3.508998 3.508998
## 1985 Q3 3.898615 3.898615
## 1985 Q4 3.453972 3.453972
## 1986 Q1 4.091660 4.091660
## 1986 Q2 4.626595 4.626595
```

To reference all Q2 observations, use the quarter() function

```
temp <- df_q.xts[quarter(df_q.xts)==2]
head(temp, 3)
```

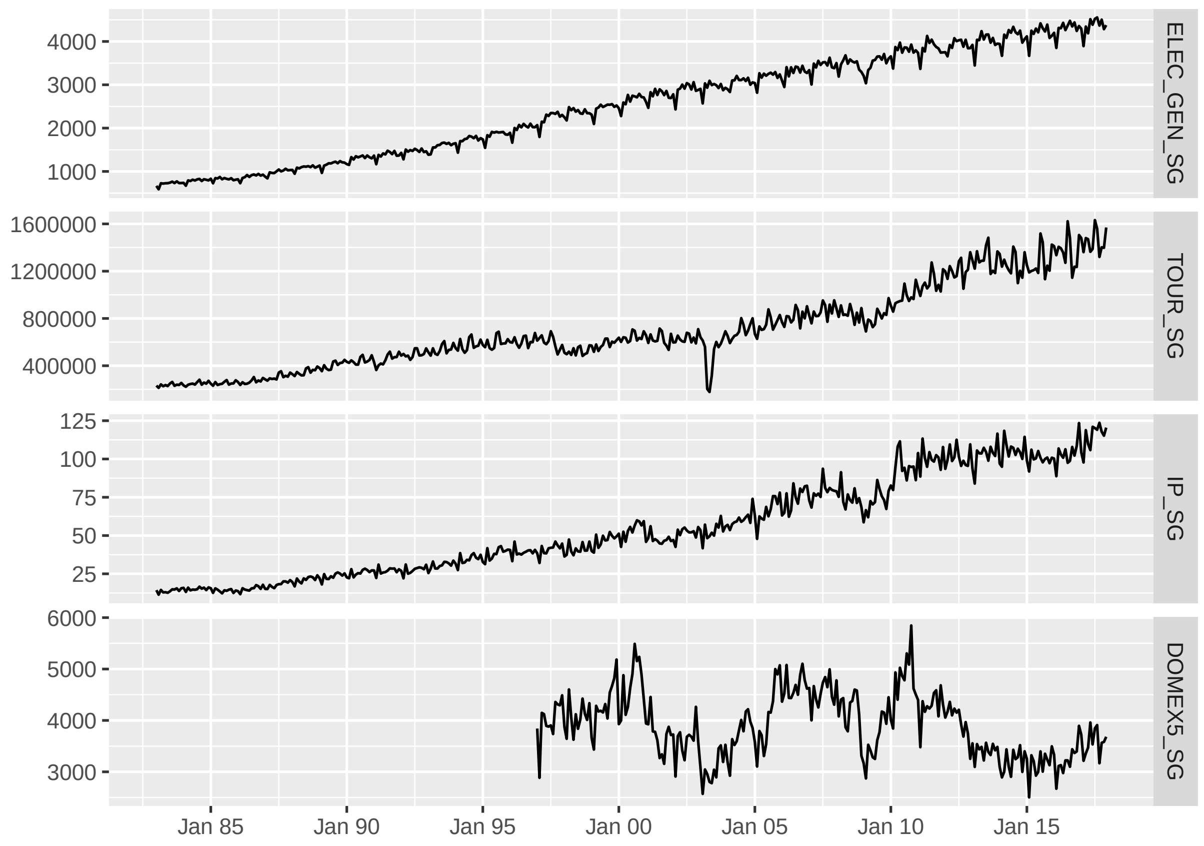
```
##                Y       Y1
## 1980 Q2 3.406533 3.406533
## 1981 Q2 3.354770 3.354770
## 1982 Q2 2.745151 2.745151
```

```
tail(temp, 3)
```

```
##                Y       Y1
## 2009 Q2 4.261111 4.873664
## 2010 Q2 4.519543 5.907393
## 2011 Q2 4.148819 4.424495
```

Plotting selected columns from an xts objects using **autoplot.zoo**.

```
sel_col <- c( "ELEC_GEN_SG","TOUR_SG","IP_SG","DOMEX5_SG")
autoplot.zoo(df_m.xts[,sel_col]) +
  facet_free(scales="free_y") +   # otw y axis limits forced to be identical
  scale_x_yearmon(format="%b %y", n=10) + xlab("")
```



You can also use autoplot.zoo on ts or mts objects (the format is sometimes a bit nicer than autoplot). You cannot use autoplot on xts objects.

From xts to ts/mts

```
df_m.mts <- as.ts(df_m.xts[,sel_col], start=start(df_m.xts))
class(df_m.mts)
```

```
## [1] "mts"    "ts"     "matrix"
```

```
summary(df_m.mts)
```

```
##   ELEC_GEN_SG        TOUR_SG            IP_SG           DOMEX5_SG
##  Min.   : 586.9   Min.   : 177808   Min.   : 11.37   Min.   :2503
##  1st Qu.:1392.5   1st Qu.: 472929   1st Qu.: 27.39   1st Qu.:3416
##  Median :2580.6   Median : 613014   Median : 47.27   Median :3870
##  Mean   :2525.7   Mean   : 701625   Mean   : 54.99   Mean   :3898
##  3rd Qu.:3545.7   3rd Qu.: 885968   3rd Qu.: 80.50   3rd Qu.:4362
##  Max.   :4555.1   Max.   :1632135   Max.   :123.65   Max.   :5848
##                                                      NA's   :168
```

From ts/mts to xts

```r
df_m.xts2 <- as.xts(df_m.mts, dateFormat="yearmon")
head(df_m.xts2, 3)
```

```
##          ELEC_GEN_SG TOUR_SG IP_SG DOMEX5_SG
## Jan 1983       667.3  232164 14.34        NA
## Feb 1983       586.9  212591 11.37        NA
## Mar 1983       727.4  242272 14.50        NA
```

```r
tail(df_m.xts2, 3)
```

```
##          ELEC_GEN_SG TOUR_SG  IP_SG DOMEX5_SG
## Oct 2017      4504.5 1402299 117.82    3565.2
## Nov 2017      4283.3 1397330 115.19    3581.6
## Dec 2017      4376.5 1569616 120.50    3682.2
```

Dynamic Regressions

A convenient package for dynamic regressions is **dynlm**. This package allows you to easily include differences, lags, trend, seasonalities into the regression.

```r
library(dynlm)
```

We run the regression

$$\Delta Y_t = \beta_0 + \beta_1 \Delta Y_{t-1} + ... + \beta_4 \Delta Y_{t-4} + \beta_5 \Delta X_t + \beta_6 \Delta X_{t-1} + \text{seasonal dummies} + \epsilon_t$$

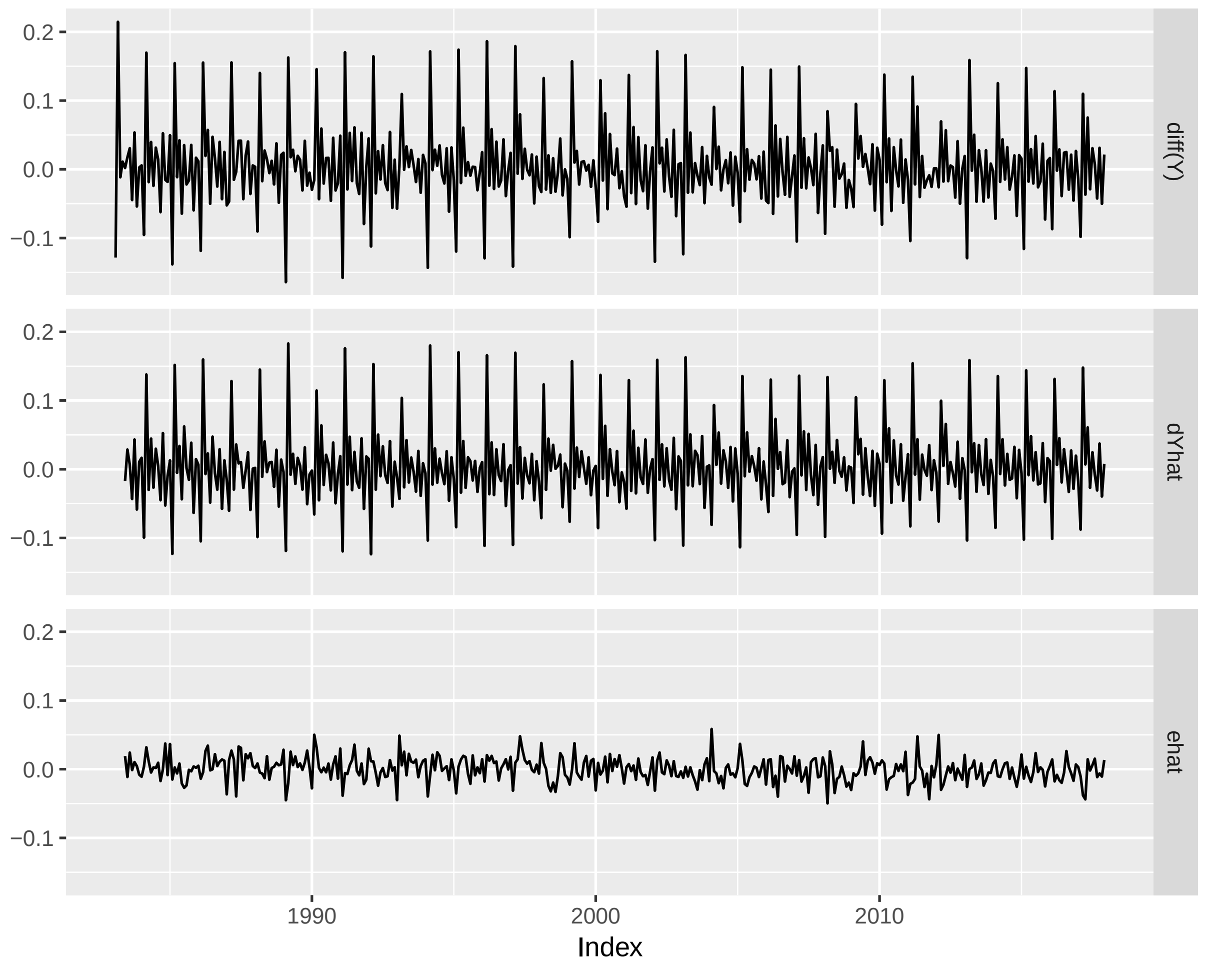
The specification is for illustration only.

```r
Y <- log(df_m.mts[,"ELEC_GEN_SG"])
X <- log(df_m.mts[,"IP_SG"])
mdl <- dynlm(d(Y) ~ L(d(Y), 1:4) +
               d(X) + L(d(X)) +
               season(Y))
summary(mdl)
```

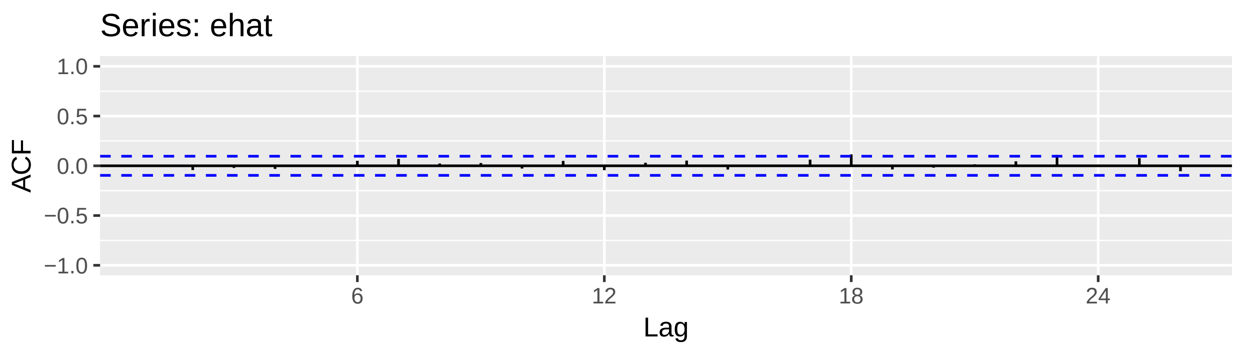
```
##
## Time series regression with "ts" data:
## Start = 1983(6), End = 2017(12)
##
## Call:
## dynlm(formula = d(Y) ~ L(d(Y), 1:4) + d(X) + L(d(X)) + season(Y))
##
## Residuals:
```

```
##       Min        1Q    Median        3Q       Max 
## -0.049768 -0.010900  0.000559  0.011738  0.058600 
## 
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)    
## (Intercept)    0.001147   0.003884   0.295  0.76793    
## L(d(Y), 1:4)1 -0.526401   0.050210 -10.484  < 2e-16 ***
## L(d(Y), 1:4)2 -0.234874   0.053989  -4.350 1.73e-05 ***
## L(d(Y), 1:4)3 -0.007456   0.053238  -0.140  0.88869    
## L(d(Y), 1:4)4  0.016891   0.045733   0.369  0.71207    
## d(X)           0.110408   0.013510   8.172 4.10e-15 ***
## L(d(X))        0.031192   0.014533   2.146  0.03246 *  
## season(Y)Feb  -0.078441   0.005885 -13.328  < 2e-16 ***
## season(Y)Mar   0.073010   0.006707  10.885  < 2e-16 ***
## season(Y)Apr   0.042698   0.007122   5.995 4.58e-09 ***
## season(Y)May   0.072258   0.009303   7.767 6.91e-14 ***
## season(Y)Jun  -0.008519   0.008578  -0.993  0.32128    
## season(Y)Jul   0.020397   0.009426   2.164  0.03106 *  
## season(Y)Aug   0.002606   0.005240   0.497  0.61923    
## season(Y)Sep  -0.026363   0.006089  -4.329 1.90e-05 ***
## season(Y)Oct   0.020479   0.004782   4.282 2.32e-05 ***
## season(Y)Nov  -0.030713   0.005308  -5.786 1.46e-08 ***
## season(Y)Dec  -0.016994   0.005739  -2.961  0.00325 ** 
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.01811 on 397 degrees of freedom
## Multiple R-squared:  0.9097, Adjusted R-squared:  0.9058 
## F-statistic: 235.2 on 17 and 397 DF,  p-value: < 2.2e-16
```

```r
dYhat <- fitted(mdl)
ehat  <- residuals(mdl)
df_plot <- cbind(diff(Y), dYhat, ehat)
autoplot.zoo(df_plot)
```



```
library(forecast)
ggAcf(ehat) + scale_y_continuous(limits=c(-1,1))
```



For hypothesis testing, the **lmtest** and **car** packages are useful. The **sandwich** operator can be used to compute Heteroskedasticity-Consistent (HC) and Heteroskedasticity and Autocorrelation Consistent (HAC) variance covariance matrices:

Example: Presenting regression output with HC standard errors.

```
library(sandwich)
library(lmtest)
library(car)
mdl <- dynlm(d(Y) ~ L(d(Y), 1:4) + d(X) + L(d(X)) + season(Y))
coeftest(mdl, vcov = vcovHC(mdl, type = "HC3"))
```

```
##
## t test of coefficients:
##
##                  Estimate Std. Error  t value  Pr(>|t|)
## (Intercept)     0.0011468  0.0046188   0.2483  0.804032
## L(d(Y), 1:4)1 -0.5264013  0.0558415  -9.4267 < 2.2e-16 ***
## L(d(Y), 1:4)2 -0.2348744  0.0575595  -4.0806 5.433e-05 ***
## L(d(Y), 1:4)3 -0.0074561  0.0520483  -0.1433  0.886163
## L(d(Y), 1:4)4  0.0168910  0.0459085   0.3679  0.713124
## d(X)           0.1104083  0.0167709   6.5833 1.459e-10 ***
## L(d(X))        0.0311915  0.0148276   2.1036  0.036040 *
## season(Y)Feb  -0.0784412  0.0076376 -10.2704 < 2.2e-16 ***
## season(Y)Mar   0.0730096  0.0080029   9.1229 < 2.2e-16 ***
## season(Y)Apr   0.0426979  0.0080257   5.3201 1.736e-07 ***
## season(Y)May   0.0722580  0.0100236   7.2088 2.871e-12 ***
## season(Y)Jun  -0.0085189  0.0097240  -0.8761  0.381521
## season(Y)Jul   0.0203974  0.0096403   2.1159  0.034980 *
## season(Y)Aug   0.0026058  0.0058706   0.4439  0.657381
## season(Y)Sep  -0.0263633  0.0061800  -4.2659 2.492e-05 ***
## season(Y)Oct   0.0204785  0.0050978   4.0171 7.048e-05 ***
## season(Y)Nov  -0.0307130  0.0055834  -5.5008 6.787e-08 ***
## season(Y)Dec  -0.0169941  0.0059598  -2.8515  0.004579 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Example: Testing single linear hypothesis. We test $\beta_1 + ... + \beta_4 = -1$

```
linearHypothesis(mdl, c("L(d(Y), 1:4)1 + L(d(Y), 1:4)2 +
                         L(d(Y), 1:4)3 + L(d(Y), 1:4)4 = -1"),
                 vcov = vcovHC(mdl, type = "HC3"), test="F") # can report test="Chisq"
```

```
## Linear hypothesis test
##
## Hypothesis:
```

```
## L(d(Y),4)1  + L(d(Y),4)2  + L(d(Y),4)3  + L(d(Y),4)4 = - 1
##
## Model 1: restricted model
## Model 2: d(Y) ~ L(d(Y), 1:4) + d(X) + L(d(X)) + season(Y)
##
## Note: Coefficient covariance matrix supplied.
##
##   Res.Df Df      F  Pr(>F)
## 1    398
## 2    397  1 2.9046 0.08911 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Example: Testing multiple hypotheses. We test if the seasonal factors are significant. We use Newey-West HAC estimators for illustration.

```
linearHypothesis(mdl, c("season(Y)Feb=0", "season(Y)Mar=0", "season(Y)Apr=0",
                        "season(Y)May=0", "season(Y)Jun=0", "season(Y)Jul=0",
                        "season(Y)Aug=0", "season(Y)Sep=0", "season(Y)Oct=0",
                        "season(Y)Nov=0", "season(Y)Dec=0"),
                 vcov=NeweyWest(mdl), test="Chisq")
```

```
## Linear hypothesis test
##
## Hypothesis:
## season(Y)Feb = 0
## season(Y)Mar = 0
## season(Y)Apr = 0
## season(Y)May = 0
## season(Y)Jun = 0
## season(Y)Jul = 0
## season(Y)Aug = 0
## season(Y)Sep = 0
## season(Y)Oct = 0
## season(Y)Nov = 0
## season(Y)Dec = 0
##
## Model 1: restricted model
## Model 2: d(Y) ~ L(d(Y), 1:4) + d(X) + L(d(X)) + season(Y)
##
## Note: Coefficient covariance matrix supplied.
```

```
##
##   Res.Df Df  Chisq Pr(>Chisq)
## 1    408
## 2    397 11 1875.6  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```