Chapter 17

Sample Study C — APT Model

The core sample study for Part V Programming as a Contributor is Sun (2011). In this chapter, the underlying statistical model and manuscript and program versions for this study are presented first. The research issue is asymmetric price transmission (APT) between China and Vietnam in the import wooden bed market of the United States. This is closely related to the issue examined in Wan et al. (2010a), i.e., the main sample study for Part IV Programming as a Wrapper. At the stage of proposal and project design, some aspects of designing several projects in one area are discussed. The relevant discussion can be found at Section 5.3.3 Design with challenging models (Sun 2011) on page 66.

The model employed is at the frontier of time series statistics, i.e., nonlinear threshold cointegration analysis. It involves hundreds of linear regressions for even a very small data set. Thus, writing new functions and even preparing a new package are needed to have an efficient data analysis. For this specific model, a new package called `apt` is created. The program version for Sun (2011) is organized with the help of this package, so the whole program has become more concise and readable.

17.1 Manuscript version for Sun (2011)

Recall that an empirical study has three versions: proposal, program, and manuscript. A proposal provides a guide like a road map for setting up the first draft of a manuscript. Like an engine, an R program can generate detailed tables and figures for the final manuscript. For this study, the brief proposal is presented at Section 5.3.3 Design with challenging models (Sun 2011). The R program for this study is presented later in this chapter. The final manuscript version is published as Sun (2011). Below is the very first manuscript version that is developed from the the proposal.

In constructing the first manuscript version for an empirical study, the key components are the tables and figures. The contents should be predicted as much as possible before a researcher works on an R program. The prediction is based on the understanding of the issue, data, model, and literature. The more a researcher can predict at this stage, the more efficient the programming will become. At the end, both the content and format of tables and figures need to be written down in the manuscript draft. For example, the results of Engle-Granger and threshold cointegration tests are reported in combination as Table 3 in Sun (2011). The first draft of these results is presented in Table 17.1. Again, some hypothetical values are put in the columns to provide formatting guides for R programming later.

17.4 Program version for Sun (2011)

When the program for an empirical study is very long (e.g., 30 pages), it may be better to organize it through several documents. The R program for Sun (2011) is five pages long only and it may not be necessary in this particular case. Nevertheless, to demonstrate the benefits of splitting a long program, two R programs are presented below. One is for the main statistical analyses and tables. The other is used to generate three figures.

17.4.1 Program for tables

The main program is listed in R Code 17.1 Main program version for generating tables in Sun (2011). This contains all the statistical analyses and can generate the four tables. Specifically, the data used in this study is pretty simple. It has four time series: import values and prices for China and Vietnam each from January 2002 to January 2010. They are saved as the data object of `daVich` in the `erer` library. The main steps in the program correspond to the study design in the proposal and desired outputs in the manuscript. These include summary statistics (Table 1), Johansen cointegration tests (Table 2), threshold cointegration tests (Table 3), and asymmetric error correction model (Table 4).

As you read along the program, you will notice that a number of new functions have been created and wrapped together in the `apt` package. This is the focus of Part V Programming as a Contributor and will be elaborated gradually. At this point, it should be evident that the program version is well organized with the help of a new package. Except some minor format differences, the tables generated from this R program are highly similar to the final versions reported in Sun (2011).

Some results in Table 3 as published in Sun (2011) were inaccurate because of a mistake made when the data was processed in 2009. The mistake was identified when the `apt` package was build in later 2010. For example, for the consistent MTAR, the coefficient for the positive term was reported as -0.251 (-2.130) in Sun (2011), but it should be -0.106 (-0.764), as calculated from below codes. This is also explained on the help page of `daVich`. The main conclusions from all the analyses are still qualitatively the same.

A large portion of R Code 17.1 is distributed with the `apt` library as an example. A number of users worldwide have raised a similar question to me. The question is simple from my perspective. However, as it occurs repeatedly from time to time, it is worthy of a note here. Briefly, the data used in Sun (2011) are just two single time series. It is tempting for another user to have two new data series imported into R and then run the whole program. Unfortunately, this will generate errors at various stages in the middle. This is because several key choices have to be made in R Code 17.1, e.g., the lag number and threshold values. The choices are based on individual data sets. Thus, one cannot just simply copy the whole R program for another data set.

R Code 17.1: Main program version for generating tables in Sun (2011)

```
# Title: R Program for Sun (2011 FPE)
library(apt); library(vars); setwd('C:/aErer')
options(width = 100, stringsAsFactors = FALSE)

# -------------------------------------------------------------------------
# 1. Data and summary statistics
# Price data for China and Vietnam are saved as 'daVich'
data(daVich); head(daVich); tail(daVich); str(daVich)
prVi <- daVich[, 1]; prCh <- daVich[, 2]
```

```
(dog <- t(bsStat(y = daVich, digits = c(3, 3))))
dog2 <- data.frame(item = rownames(dog), CH.level = dog[, 2],
  CH.diff = '__', VI.level = dog[, 1], VI.diff = '__')[2:6, ]
rownames(dog2) <- 1:nrow(dog2); str(dog2); dog2

# -------------------------------------------------------------------------
# 2. Unit root test (Table 1)
ch.t1 <- ur.df(type = 'trend', lags = 3,  y = prCh); slotNames(ch.t1)
ch.d1 <- ur.df(type = 'drift', lags = 3,  y = prCh)
ch.t2 <- ur.df(type = 'trend', lags = 3,  y = diff(prCh))
ch.d2 <- ur.df(type = 'drift', lags = 3,  y = diff(prCh))
vi.t1 <- ur.df(type = 'trend', lags = 12, y = prVi)
vi.d1 <- ur.df(type = 'drift', lags = 11, y = prVi)
vi.t2 <- ur.df(type = 'trend', lags = 10, y = diff(prVi))
vi.d2 <- ur.df(type = 'drift', lags = 10, y = diff(prVi))
dog2[6, ] <- c('ADF with trend',
  paste(round(ch.t1@teststat[1], digits = 3), '[', 3,  ']', sep = ''),
  paste(round(ch.t2@teststat[1], digits = 3), '[', 3,  ']', sep = ''),
  paste(round(vi.t1@teststat[1], digits = 3), '[', 12, ']', sep = ''),
  paste(round(vi.t2@teststat[1], digits = 3), '[', 10, ']', sep = ''))
dog2[7, ] <- c('ADF with drift',
  paste(round(ch.d1@teststat[1], digits = 3), '[', 3,  ']', sep = ''),
  paste(round(ch.d2@teststat[1], digits = 3), '[', 3,  ']', sep = ''),
  paste(round(vi.d1@teststat[1], digits = 3), '[', 11, ']', sep = ''),
  paste(round(vi.d2@teststat[1], digits = 3), '[', 10, ']', sep = ''))
(table.1 <- dog2)

# -------------------------------------------------------------------------
# 3. Johansen-Juselius and Engle-Granger cointegration analyses
# JJ cointegration
VARselect(daVich, lag.max = 12, type = 'const')
summary(VAR(daVich, type = 'const', p = 1))
K <- 5; two <- cbind(prVi, prCh)
summary(j1 <- ca.jo(x = two, type = 'eigen', ecdet = 'trend', K = K))
summary(j2 <- ca.jo(x = two, type = 'eigen', ecdet = 'const', K = K))
summary(j3 <- ca.jo(x = two, type = 'eigen', ecdet = 'none' , K = K))
summary(j4 <- ca.jo(x = two, type = 'trace', ecdet = 'trend', K = K))
summary(j5 <- ca.jo(x = two, type = 'trace', ecdet = 'const', K = K))
summary(j6 <- ca.jo(x = two, type = 'trace', ecdet = 'none' , K = K))
slotNames(j1)
out1 <- cbind('eigen', 'trend', K, round(j1@teststat, digits = 3), j1@cval)
out2 <- cbind('eigen', 'const', K, round(j2@teststat, digits = 3), j2@cval)
out3 <- cbind('eigen', 'none',  K, round(j3@teststat, digits = 3), j3@cval)
out4 <- cbind('trace', 'trend', K, round(j4@teststat, digits = 3), j4@cval)
out5 <- cbind('trace', 'const', K, round(j5@teststat, digits = 3), j5@cval)
out6 <- cbind('trace', 'none',  K, round(j6@teststat, digits = 3), j6@cval)
jjci <- rbind(out1, out2, out3, out4, out5, out6)
colnames(jjci) <- c('test 1', 'test 2', 'lag', 'statistic',
    'c.v 10%', 'c.v 5%', 'c.v 1%')
```

```
rownames(jjci) <- 1:nrow(jjci)
(table.2 <- data.frame(jjci))

# EG cointegration
LR <- lm(formula = prVi ~ prCh); summary(LR)
(LR.coef <- round(summary(LR)$coefficients, digits = 3))
(ry <- ts(data = residuals(LR), start = start(prCh), end = end(prCh),
  frequency = 12))
eg <- ur.df(y = ry, type = c('none'), lags = 1)
eg2 <- ur.df2(y = ry, type = c('none'), lags = 1)
(eg4 <- Box.test(eg@res, lag = 4, type = 'Ljung') )
(eg8 <- Box.test(eg@res, lag = 8, type = 'Ljung') )
(eg12 <- Box.test(eg@res, lag = 12, type = 'Ljung'))
EG.coef <- coefficients(eg@testreg)[1, 1]
EG.tval <- coefficients(eg@testreg)[1, 3]
(res.EG <- round(t(data.frame(EG.coef, EG.tval, eg2$aic, eg2$bic,
  eg4$p.value, eg8$p.value, eg12$p.value)), digits = 3))

# -----------------------------------------------------------------------
# 4. Threshold cointegration
# best threshold
test <- ciTarFit(y = prVi, x = prCh); test; names(test)
t3 <- ciTarThd(y = prVi, x = prCh, model = 'tar', lag = 0); plot(t3)
time.org <- proc.time()
(th.tar <- t3$basic)
for (i in 1:12) {  # about 20 seconds
  t3a <- ciTarThd(y = prVi, x = prCh, model = 'tar', lag = i)
  th.tar[i+2] <- t3a$basic[, 2]
}
th.tar
time.org - proc.time()

t4 <- ciTarThd(y = prVi, x = prCh, model = 'mtar', lag = 0)
(th.mtar <- t4$basic); plot(t4)
for (i in 1:12) {  # about 36 seconds
  t4a <- ciTarThd(y = prVi, x = prCh, model = 'mtar', lag = i)
  th.mtar[i+2] <- t4a$basic[,2]
}
th.mtar

t.tar <- -8.041; t.mtar <- -0.451     # lag = 0 to 4; final choices
# t.tar <- -8.701 ; t.mtar <- -0.451  # lag = 5 to 12

mx <- 12  # lag selection
(g1 <-ciTarLag(y=prVi, x=prCh, model='tar',  maxlag = mx, thresh = 0))
(g2 <-ciTarLag(y=prVi, x=prCh, model='mtar', maxlag = mx, thresh = 0))
(g3 <-ciTarLag(y=prVi, x=prCh, model='tar',  maxlag = mx, thresh = t.tar))
(g4 <-ciTarLag(y=prVi, x=prCh, model='mtar', maxlag = mx, thresh = t.mtar))
plot(g1)
```

```
# Figure of threshold selection: mtar at lag = 3 (Figure 3 data)
(t5 <- ciTarThd(y=prVi, x=prCh, model = 'mtar', lag = 3, th.range = 0.15))
plot(t5)

# Table 3 Results of EG and threshold cointegration combined
vv <- 3
(f1 <- ciTarFit(y=prVi, x=prCh, model = 'tar',  lag = vv, thresh = 0))
(f2 <- ciTarFit(y=prVi, x=prCh, model = 'tar',  lag = vv, thresh = t.tar ))
(f3 <- ciTarFit(y=prVi, x=prCh, model = 'mtar', lag = vv, thresh = 0))
(f4 <- ciTarFit(y=prVi, x=prCh, model = 'mtar', lag = vv, thresh = t.mtar))

r0 <- cbind(summary(f1)$dia, summary(f2)$dia,
            summary(f3)$dia, summary(f4)$dia)
diag <- r0[c(1:4, 6:7, 12:14, 8, 9, 11), c(1, 2, 4, 6, 8)]
rownames(diag) <- 1:nrow(diag); diag

e1 <- summary(f1)$out; e2 <- summary(f2)$out
e3 <- summary(f3)$out; e4 <- summary(f4)$out; rbind(e1, e2, e3, e4)
ee <- list(e1, e2, e3, e4); vect <- NULL
for (i in 1:4) {
  ef <- data.frame(ee[i])
  vect2 <- c(paste(ef[3, 'estimate'], ef[3, 'sign'], sep = ''),
    paste('(', ef[3, 't.value'], ')', sep = ''),
    paste(ef[4, 'estimate'], ef[4, 'sign'], sep = ''),
    paste('(', ef[4, 't.value'], ')', sep = ''))
  vect <- cbind(vect, vect2)
}
item <- c('pos.coeff','pos.t.value', 'neg.coeff','neg.t.value')
ve <- data.frame(cbind(item, vect)); colnames(ve) <- colnames(diag)
(res.CI <- rbind(diag, ve)[c(1:2, 13:16, 3:12), ])
rownames(res.CI) <- 1:nrow(res.CI)
res.CI$Engle <- '__'
res.CI[c(3, 4, 9:13), 'Engle'] <- res.EG[, 1]
res.CI[4, 6] <- paste('(', res.CI[4, 6], ')', sep = '')
(table.3 <- res.CI[, c(1, 6, 2:5)])

# ------------------------------------------------------------------------
# 5. Asymmstric error correction model
(sem <- ecmSymFit(y = prVi, x = prCh, lag = 4)); names(sem)
(aem <- ecmAsyFit(y = prVi, x = prCh, lag = 4, model = 'mtar',
   split = TRUE, thresh = t.mtar))
(ccc <- summary(aem))
coe <- cbind(as.character(ccc[1:19, 2]),
  paste(ccc[1:19, 'estimate'], ccc$signif[1:19], sep = ''),
  ccc[1:19, 't.value'],
  paste(ccc[20:38, 'estimate'], ccc$signif[20:38],sep = ''),
  ccc[20:38, 't.value'])
colnames(coe) <- c('item', 'CH.est', 'CH.t', 'VI.est','VI.t')
```

```

(edia <- ecmDiag(aem, 3)); (ed <- edia[c(1, 6:9), ])
ed2 <- cbind(ed[, 1:2], '_', ed[, 3], '_'); colnames(ed2) <- colnames(coe)
(tes <- ecmAsyTest(aem)$out); (tes2 <- tes[c(2, 3, 5, 11:13, 1), -1])
tes3 <- cbind(as.character(tes2[, 1]),
  paste(tes2[, 2], tes2[, 6], sep = ''),
  paste('[', round(tes2[, 4], digits = 2), ']', sep = ''),
  paste(tes2[, 3], tes2[, 7], sep = ''),
  paste('[', round(tes2[, 5], digits = 2), ']', sep = ''))
colnames(tes3) <- colnames(coe)
(table.4 <- data.frame(rbind(coe, ed2, tes3)))

# -------------------------------------------------------------------------
# 6. Output
(output <- listn(table.1, table.2, table.3, table.4))
write.list(z = output, file = 'OutBedTable.csv')
```

Note: Major functions used in R Code 17.1 are: `ur.df()`, `ca.jo()`, `VAR()`, `ciTarThd()`, `ciTarLag()`, `ciTarFit()`, `ecmSymFit()`, `ecmAsyFit()`, `ecmDiag()`, `bsStat()`, `Box.test()`, and `lm()`.

```
# Selected results from R Code 17.1
> table.1
            item  CH.level   CH.diff   VI.level    VI.diff
1           mean   148.791        __    115.526         __
2           stde    11.461        __      9.882         __
3           mini   119.618        __     99.335         __
4           maxi   177.675        __    150.721         __
5           obno        97        __         97         __
6 ADF with trend -2.956[3] -7.394[3] -2.936[12] -5.777[10]
7 ADF with drift -2.422[3] -7.195[3] -1.161[11]  -5.74[10]

> table.2
   test.1 test.2 lag statistic c.v.10. c.v.5. c.v.1.
1   eigen  trend   5    10.001   10.49  12.25  16.26
2   eigen  trend   5    20.253   16.85  18.96  23.65
3   eigen  const   5     4.461    7.52   9.24  12.97
4   eigen  const   5    14.304   13.75  15.67   20.2
5   eigen   none   5     4.438     6.5   8.18  11.65
6   eigen   none   5      14.3   12.91   14.9  19.19
7   trace  trend   5    10.001   10.49  12.25  16.26
8   trace  trend   5    30.254   22.76  25.32  30.45
9   trace  const   5     4.461    7.52   9.24  12.97
10  trace  const   5    18.765   17.85  19.96   24.6
11  trace   none   5     4.438     6.5   8.18  11.65
12  trace   none   5    18.738   15.66  17.95  23.52

> table.3
          item    Engle       tar     c.tar      mtar    c.mtar
1          lag       __         3         3         3         3
```

```
2       thresh       __         0    -8.041         0    -0.451
3    pos.coeff   -0.407  -0.328**   -0.28**    -0.116    -0.106
4  pos.t.value (-4.173)  (-2.523)  (-2.306)  (-0.824)  (-0.764)
5    neg.coeff       __ -0.515*** -0.721*** -0.658*** -0.677***
6  neg.t.value       __  (-3.119)  (-3.942)  (-4.754)  (-4.888)
7    total obs       __        97        97        97        97
8    coint obs       __        93        93        93        93
9          aic  669.627   658.998   654.863   650.612   649.495
10         bic  677.351   674.193   670.059   665.808    664.69
11  LB test(4)    0.773     0.961     0.879     0.988     0.987
12  LB test(8)    0.919     0.992     0.964     0.999     0.998
13 LB test(12)    0.239     0.122     0.084     0.289     0.333
14   H1: no CI       __     6.539     8.836    11.307    11.976
15  H2: no APT       __     1.033     5.081     9.435    10.612
16 H2: p.value       __     0.312     0.027     0.003     0.002

> table.4[1:7, ]
                 item    CH.est   CH.t  VI.est   VI.t
1         (Intercept)    -0.146 -0.052 -3.853* -1.777
2 X.diff.prCh.t_1.pos -0.622*** -2.755 -0.155  -0.897
3 X.diff.prCh.t_2.pos     0.082  0.344 -0.144  -0.795
4 X.diff.prCh.t_3.pos    -0.282 -1.264  0.146   0.854
5 X.diff.prCh.t_4.pos    -0.324 -1.403 -0.193  -1.091
6 X.diff.prCh.t_1.neg   -0.314. -1.464 -0.105  -0.641
7 X.diff.prCh.t_2.neg -0.584*** -2.651  0.085   0.508
```

17.4.2 Program for figures

The three figures reported in Sun (2011) can be created by base R graphics or the `ggplot2` package. These codes for graphs are organized separately as a document to increase readability, and they are all presented in R Code 17.2. When the codes for figure generation is long, this can make the main program more concise.

There are several ways to connect individual programs for a specific empirical study, e.g., R Codes 17.1 and 17.2 for this sample study. First, the main program can be called by the `source()` function and all data will become available for another program. Alternatively, if it takes a long time to run the main program each time or the data used in another program is small, then the relevant data can be copied or generated directly. This is exactly true for the relation between the two programs here. In general, figures use fewer data than statistical analyses. A threshold cointegration analysis often takes quite some time to finish. Thus, at the beginning of R Code 17.2, the value data for Figure 1, price data for Figure 2, and sum of squared errors for Figure 3 are generated directly, without calling the main program.

Figure 17.1 is generated from traditional graphics system, and Figure 17.2 is from `ggplot2`. The main difference is that the ggplot version has a gray background and grid lines. Which version is more attractive is largely a personal choice. The codes used for the ggplot version is generally longer than these for the base R version. One can also customize the appearance of the ggplot version and make it very similar to the version from base R. This is left as Exercise 17.6.1 on page 398.

In Sun (2011), Figure 1 is monthly import values for China and Vietnam, as shown in Figure 17.2, and Figure 2 is their monthly import prices. Both the figures can be created

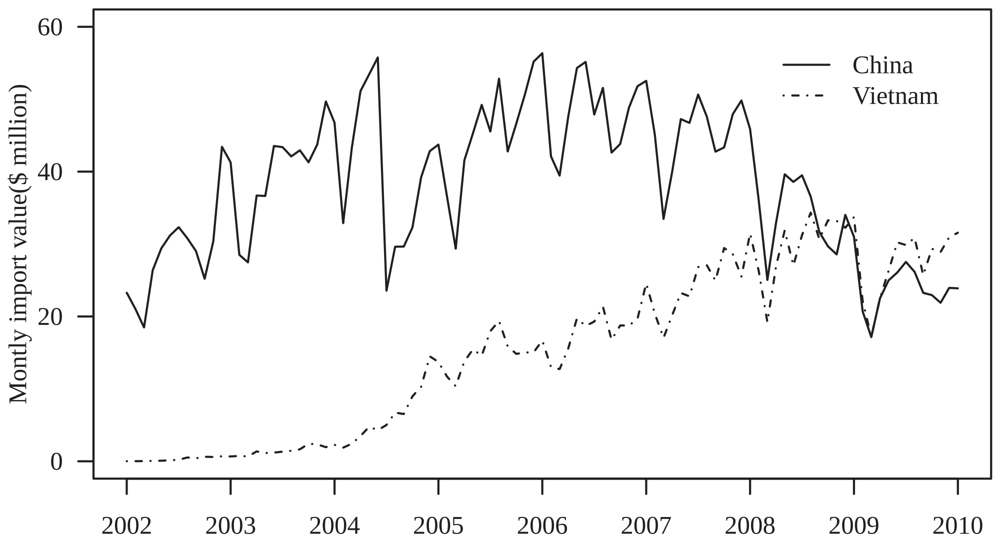

Figure 17.1: Monthly import value of beds from China and Vietnam (base R)

with the `ggplot2` package. Recall that `%+%` is defined in `ggplot2` to replace one data frame with another one. It is tempting to use this operator to generate Figure 2 with a substitution of the underlying data frame. However, the value and price data are quite different in scale. As a result, it is faster in this case to copy the whole block of codes for Figure 1 and then revise them for Figure 2.

R Code 17.2: Graph program version for generating figures in Sun (2011)

```
# Title: Graph codes for Sun (2011 FPE)
library(apt); library(ggplot2); setwd('C:/aErer'); data(daVich)

# -------------------------------------------------------------------------
# A. Data for graphs: value, price, and t5$path
prVi <- daVich[, 1]; prCh <- daVich[, 2]
vaVi <- daVich[, 3]; vaCh <- daVich[, 4]
(date <- as.Date(time(daVich), format = '%Y/%m/%d'))
(value <- data.frame(date, vaCh, vaVi))
(price <- data.frame(date, prVi, prCh))
(t5 <- ciTarThd(y=prVi, x=prCh, model = 'mtar', lag = 3, th.range = 0.15))

# -------------------------------------------------------------------------
# B. Traditonal graphics
# Figure 1 Import values from China and Vietnam
win.graph(width = 5, height = 2.8, pointsize = 9); bringToTop(stay = TRUE)
par(mai = c(0.4, 0.5, 0.1, 0.1), mgp = c(2, 1, 0), family = 'serif')
plot(x = vaCh, lty = 1, lwd = 1, ylim = c(0, 60), xlab = '',
  ylab = 'Montly import value($ million)', axes = FALSE)
box(); axis(side = 1, at = 2002:2010)
axis(side = 2, at = c(0, 20, 40, 60), las = 1)
lines(x = vaVi, lty = 4, lwd = 1)
```

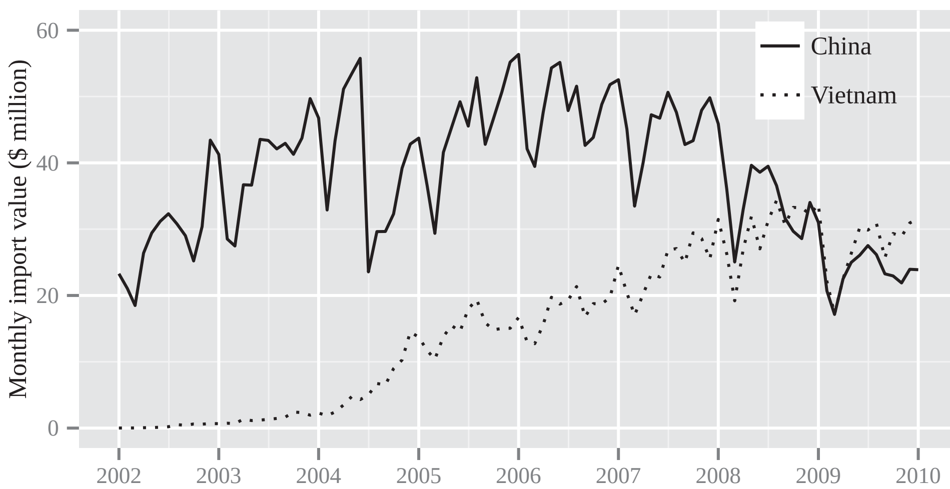

Figure 17.2: Monthly import value of beds from China and Vietnam (`ggplot2`)

```
legend(x = 2008.1, y = 59, legend = c('China', 'Vietnam'),
  lty = c(1, 4), box.lty = 0)
fig1.base <- recordPlot()

# Figure 2 Import prices from China and Vietnam
win.graph(width = 5, height = 2.8, pointsize = 9)
par(mai = c(0.4, 0.5, 0.1, 0.1), mgp = c(2, 1, 0), family = 'serif')
plot(x = prCh, lty = 1, type = 'l', lwd = 1, ylim = range(prCh, prVi),
  xlab = '', ylab = 'Monthly import price ($/piece)' )
lines(x = prVi, lty = 3, type = 'l', lwd = 1)
legend(x = 2008.5, y = 175, legend = c('China', 'Vietnam'),
  lty = c(1, 3), box.lty = 0)

# Figure 3 Sum of dquared errors by threshold value from MTAR
win.graph(width = 5.1, height = 3.3, pointsize = 9)
par(mai = c(0.5, 0.5, 0.1, 0.1),  mgp = c(2.2, 1, 0), family = 'serif')
plot(formula = path.sse ~ path.thr, data = t5$path, type = 'l',
     ylab = 'Sum of Squared Errors', xlab = 'Threshold value')

# ----------------------------------------------------------------------
# C. ggplot for three figures
pp <- theme(axis.text   = element_text(size = 8, family = 'serif')) +
      theme(axis.title  = element_text(size = 9, family = 'serif')) +
      theme(legend.text = element_text(size = 9, family = 'serif')) +
      theme(legend.position = c(0.85, 0.9) ) +
      theme(legend.key = element_rect(fill = 'white', color = NA)) +
      theme(legend.background = element_rect(fill = NA, color = NA))

fig1 <- ggplot(data = value, aes(x = date)) +
```

```
  geom_line(aes(y = vaCh, linetype = 'China')) +
  geom_line(aes(y = vaVi, linetype = 'Vietnam')) +
  scale_linetype_manual(name = '', values = c(1, 3)) +
  scale_x_date(name = '', labels = as.character(2002:2010), breaks =
    as.Date(paste(2002:2010, '-1-1', sep = ''), format = '%Y-%m-%d')) +
  scale_y_continuous(limits = c(0, 60),
    name = 'Monthly import value ($ million)') + pp

fig2 <- ggplot(data = price, aes(x = date)) +
  geom_line(aes(y = prCh, linetype = 'China')) +
  geom_line(aes(y = prVi, linetype = 'Vietnam')) +
  scale_linetype_manual(name = '', values = c(1, 3))+
  scale_x_date(name = '', labels = as.character(2002:2010), breaks =
    as.Date(paste(2002:2010, '-1-1', sep = ''), format = '%Y-%m-%d')) +
  scale_y_continuous(limits = c(98, 180),
    name = 'Monthly import price ($/piece)') + pp

fig3 <- ggplot(data = t5$path) +
  geom_line(aes(x = path.thr, y = path.sse)) +
  labs(x = 'Threshold value', y = 'Sum of squared errors') +
  scale_y_continuous(limits = c(5000, 5700)) +
  scale_x_continuous(breaks = c(-10:7)) +
  theme(axis.text  = element_text(size = 8, family = 'serif')) +
  theme(axis.title = element_text(size = 9, family = 'serif'))

# -------------------------------------------------------------------------
# D. Show on screen devices or save on file devices
pdf(file = 'OutBedFig1base.pdf', width = 5, height = 2.8, pointsize = 9)
replayPlot(fig1.base); dev.off()
windows(width = 5, height = 2.8); fig1
windows(width = 5, height = 2.8); fig2
windows(width = 5, height = 2.8); fig3
ggsave(fig1, file = 'OutBedFig1ggplot.pdf', width = 5, height = 2.8)
ggsave(fig2, file = 'OutBedFig2ggplot.pdf', width = 5, height = 2.8)
ggsave(fig3, file = 'OutBedFig3ggplot.pdf', width = 5, height = 2.8)
```

17.5 Road map: how to create a package (Part V)

Two large parts for R programming have been presented so far in this book. In Part III Programming as a Beginner, basic R concepts and data manipulations are elaborated. Using existing functions for specific analyses is emphasized. In Part IV Programming as a Wrapper, the structure of an R function is examined and how to write new functions is demonstrated through various applications. Assuming you have learned these techniques well, we now reach the final stage of the growing-up process: creating a new package for a statistical model or research issue.

In general, the materials in the Part for beginner are more difficult than these in the Part for wrapper. The materials in the present Part, i.e., V Programming as a Contributor, are the easiest. The main challenge for creating a new package is to design the structure