Basic R Functions of Handling Financial Data

Zheng Tian

Contents

1 Basic R Functions for Financial Data

1.1 Downloading Financial Data

We can download financial data with the function `getSymbols()` in the package `quantmod` in R from open sources, like Yahoo Finance, Google Finance, and the Federal Reserve Bank of St. Louis (FRED).

```r
library(quantmod, quietly = TRUE)
```

```
##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric

## Version 0.4-0 included new data defaults. See ?getSymbols.
```

```r
options("getSymbols.warning4.0"=FALSE)
getSymbols('AAPL', src = "yahoo", from="2005-01-02", to="2010-12-31")
```

The data object obtained through `getSymbols()` is `xts`, i.e., extensible time series.

```r
class(AAPL)
```

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

We can check the data using `head()` and `tail`. And we can select a subset of the data within a range by calling the row names in the format as `yyyy-mm-dd`.

```r
# Look at the first three observations
head(AAPL, n = 3)
```

```
##            AAPL.Open AAPL.High AAPL.Low AAPL.Close AAPL.Volume
## 2005-01-03     64.78     65.11    62.60      63.29   172998000
## 2005-01-04     63.79     65.47    62.97      63.94   274202600
## 2005-01-05     64.46     65.25    64.05      64.50   170108400
##            AAPL.Adjusted
## 2005-01-03      4.099912
## 2005-01-04      4.142019
## 2005-01-05      4.178296
```

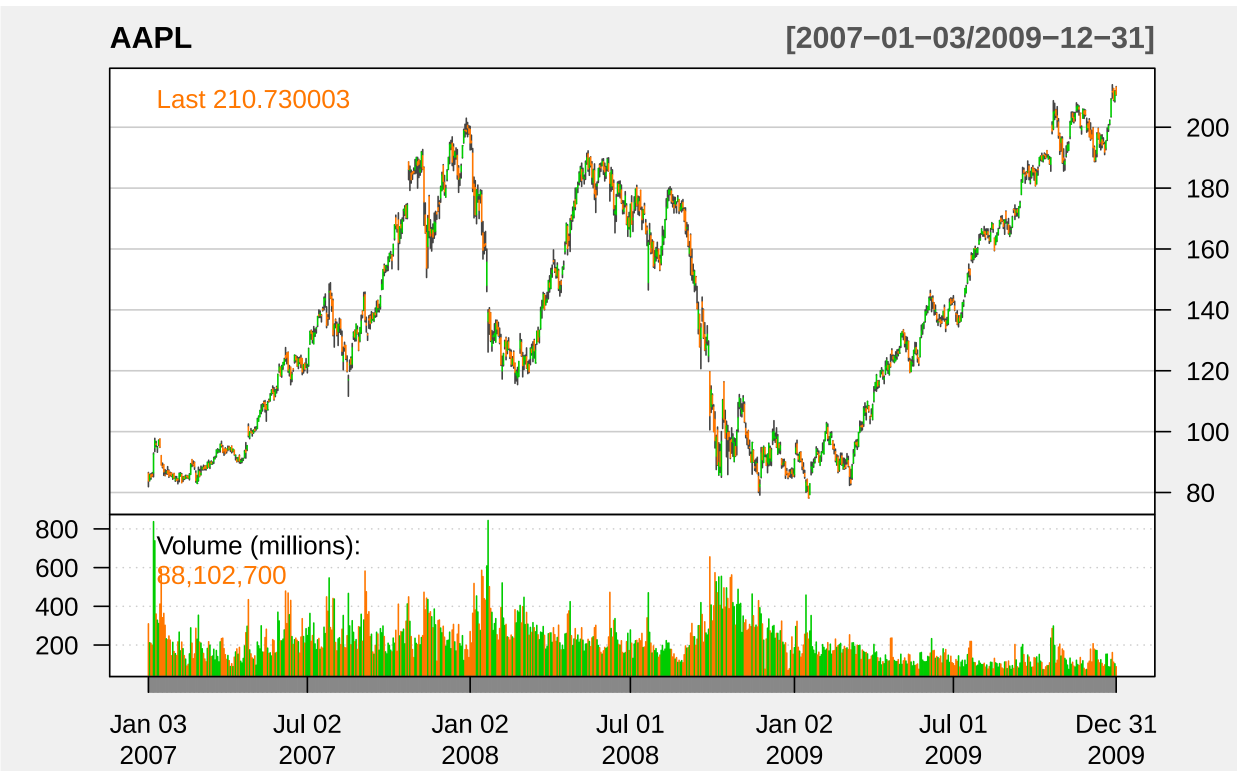
```r
# Look at the observations from December 1st, 2010 to December 7th, 2010
AAPL["2010-12-01::2010-12-07"]
```

```
##            AAPL.Open AAPL.High AAPL.Low AAPL.Close AAPL.Volume
## 2010-12-01    315.27    317.75   315.00     316.40   115437700
## 2010-12-02    317.53    319.00   314.89     318.15   115709300
## 2010-12-03    317.01    318.65   316.34     317.44    85523200
## 2010-12-06    318.64    322.33   318.42     320.15   112120400
## 2010-12-07    323.80    323.99   318.12     318.21    97863500
##            AAPL.Adjusted
## 2010-12-01      40.99264
## 2010-12-02      41.21937
## 2010-12-03      41.12738
## 2010-12-06      41.47849
## 2010-12-07      41.22715
```

1.2 Visualizing Financial Data

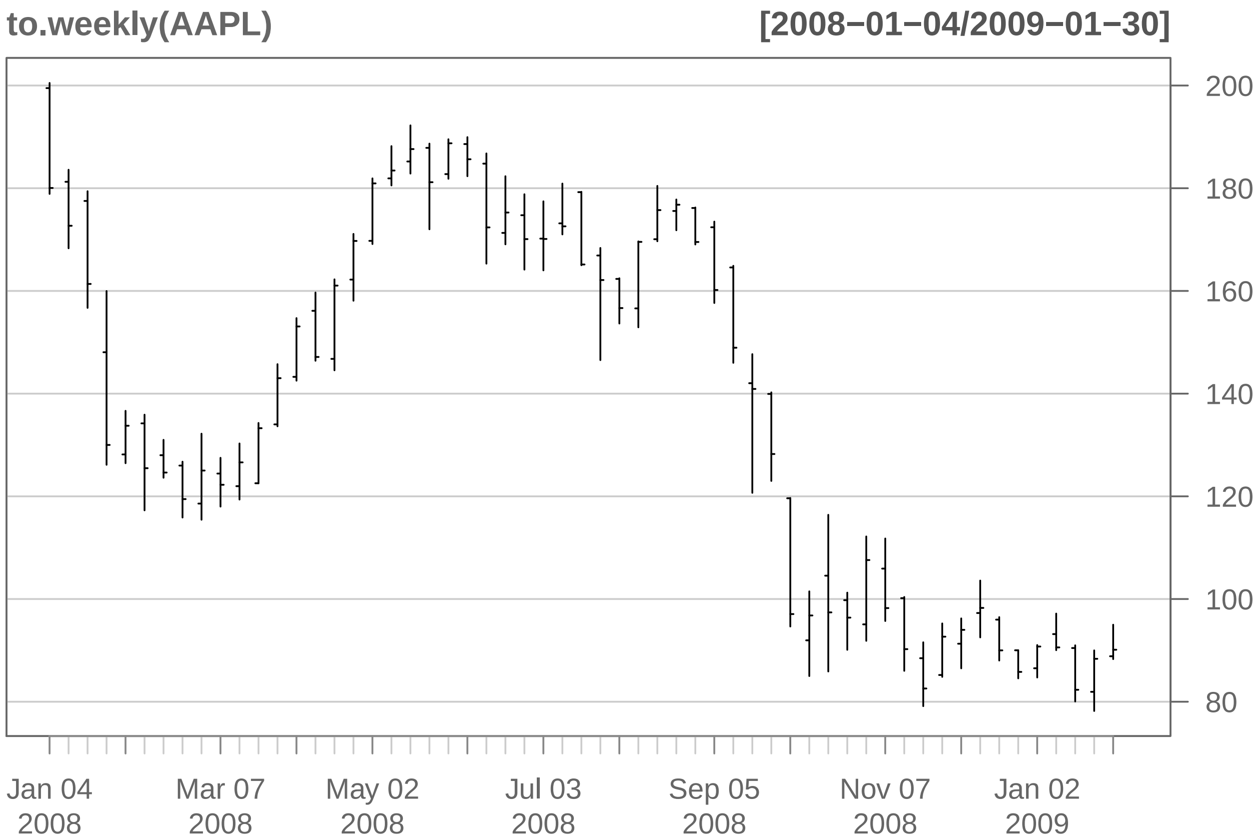
We can visualize the downloaded financial data with the function `chartSeries()`.

```r
chartSeries(AAPL, subset = "2007::2009", theme = "white")
```



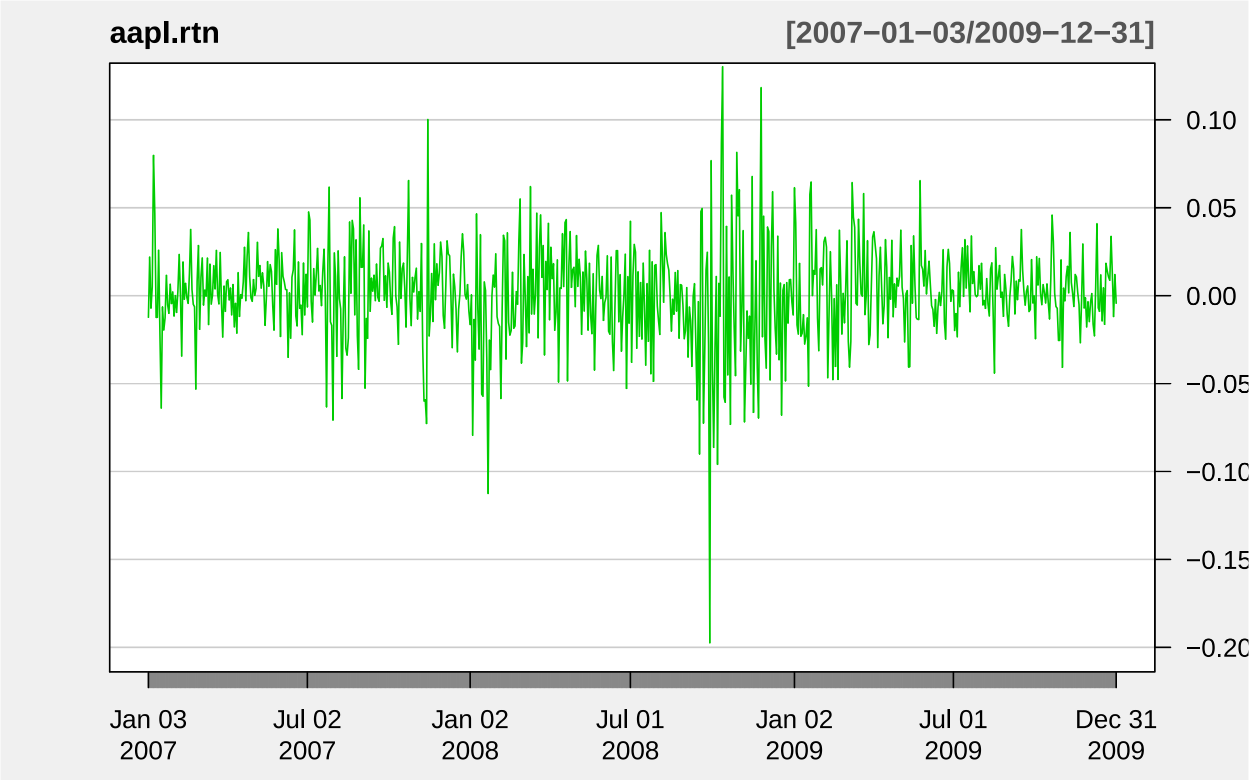
Since the data set comforms with an `OHLC` data format, we can plot the range of stock prices with a trading period with bar charts. In the following chart, we converted the daily stock prices to weekly prices using the function `to.weekly()`, and removed the trade volumn by setting `TA = NULL`.

```r
barChart(to.weekly(AAPL), subset = "2008:01::2009:01",
         theme='white.mono', TA = NULL, bar.type='ohlc')
```



Next we can generate the log return series, and plot it.

```
aapl.rtn <- diff(log(AAPL))
chartSeries(aapl.rtn, subset = "2007::2009", type = 'line',
            TA = NULL, theme = "white")
```



2 Linear Time Series Models with R

We consider the example of the monthly value-weighted return of IBM stock, which uses the data file `m-ibm3dx2608.txt`.

```
ibm_rtn <- read.table('m-ibm3dx2608.txt', header = TRUE)
head(ibm_rtn)
```

```
##       date    ibmrtn     vwrtn     ewrtn     sprtn
## 1 19260130 -0.010381  0.000724  0.023174  0.022472
## 2 19260227 -0.024476 -0.033374 -0.053510 -0.043956
## 3 19260331 -0.115591 -0.064341 -0.096824 -0.059113
## 4 19260430  0.089783  0.038358  0.032946  0.022688
## 5 19260528  0.036932  0.012172  0.001035  0.007679
## 6 19260630  0.068493  0.056888  0.050487  0.043184
```

```
# tail(ibm_rtn)
```

We can convert the first column in `ibm_rtn` to a `Date` object, and convert the third column, which is the monthly value-weighted return, to a `ts` object.

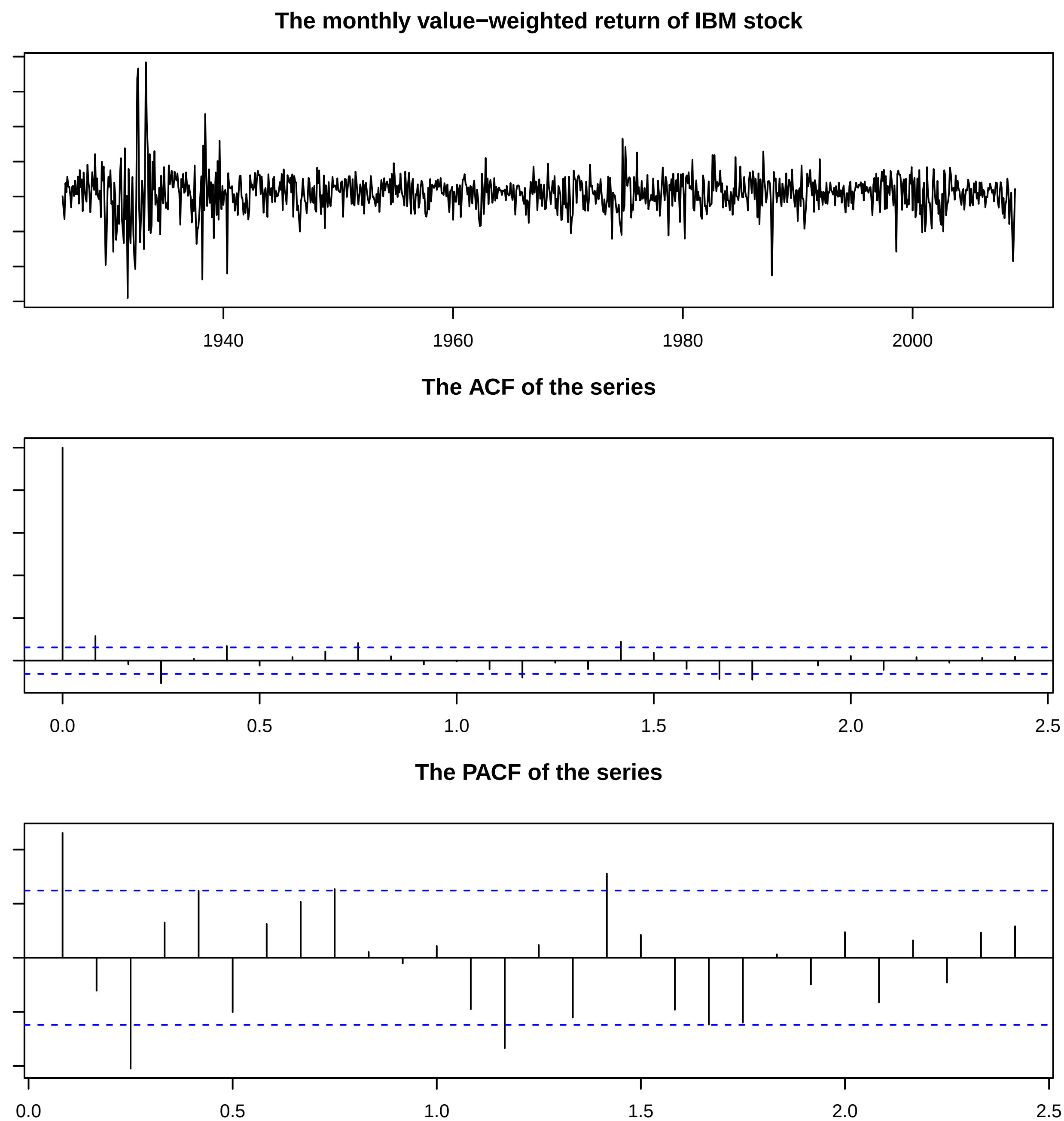
```
ibm_date <- as.Date(as.character(ibm_rtn$date), "%Y%m%d")
class(ibm_date)
```

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

```
vw_rtn <- ts(ibm_rtn[, 3], start=c(1926, 1), frequency = 12)
```

We plot the series and its ACF and PACF.

```
op <- par(mfrow = c(3, 1), mar=c(3, 1, 3, 1), pty = "m")
plot(vw_rtn, main="The monthly value-weighted return of IBM stock")
acf_vw_rtn <- acf(vw_rtn, main="The ACF of the series")
pacf_vw_rtn <- pacf(vw_rtn, main="The PACF of the series")
```




```
par(op)
```

We use an AR(3) model to examine the return series, i.e.,

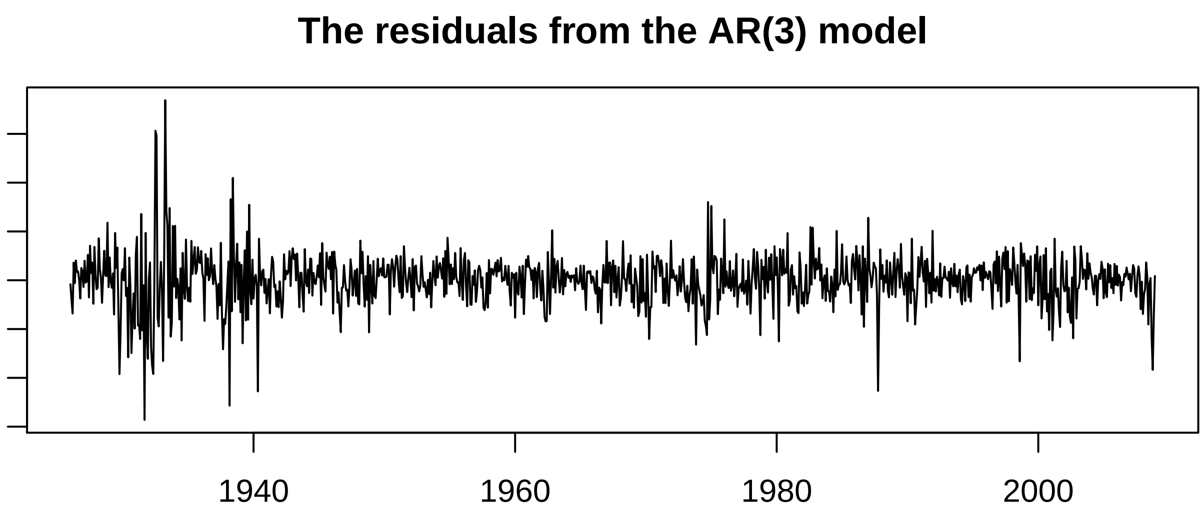
$$r_t = \phi_0 + \phi_1 r_{t-1} + \phi_2 r_{t-2} + \phi_3 r_{t-3} + a_t$$

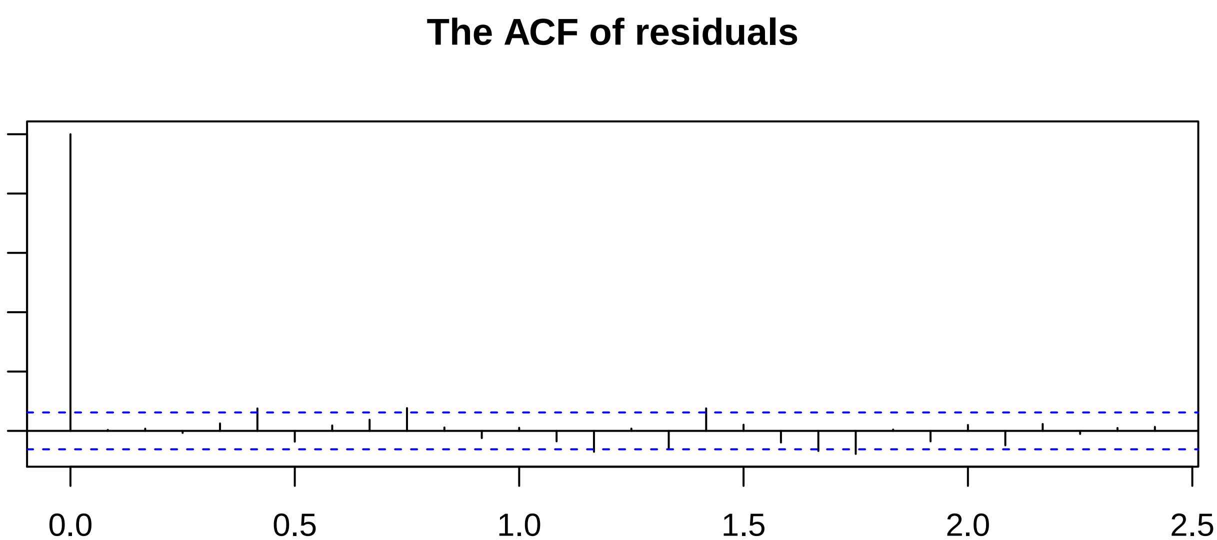
```r
vw_ar3 <- arima(vw_rtn, order=c(3,0,0))
vw_ar3
```

```
##
## Call:
## arima(x = vw_rtn, order = c(3, 0, 0))
##
## Coefficients:
##          ar1      ar2      ar3  intercept
##       0.1158  -0.0187  -0.1042     0.0089
## s.e.  0.0315   0.0317   0.0317     0.0017
##
## sigma^2 estimated as 0.002875:  log likelihood = 1500.86,  aic = -2991.73
```

We can check the model adequacy by testing whether the residual series follows a white noise series. First, we plot the ACF of the residuals and then do a Ljung-Box test.

```r
vw_ar3_resid <- resid(vw_ar3)
op <- par(mfrow = c(2, 1), mar=c(3, 1, 3, 1))
plot(vw_ar3_resid, main="The residuals from the AR(3) model")
acf(vw_ar3_resid, main="The ACF of residuals")
```






```
par(op)
```

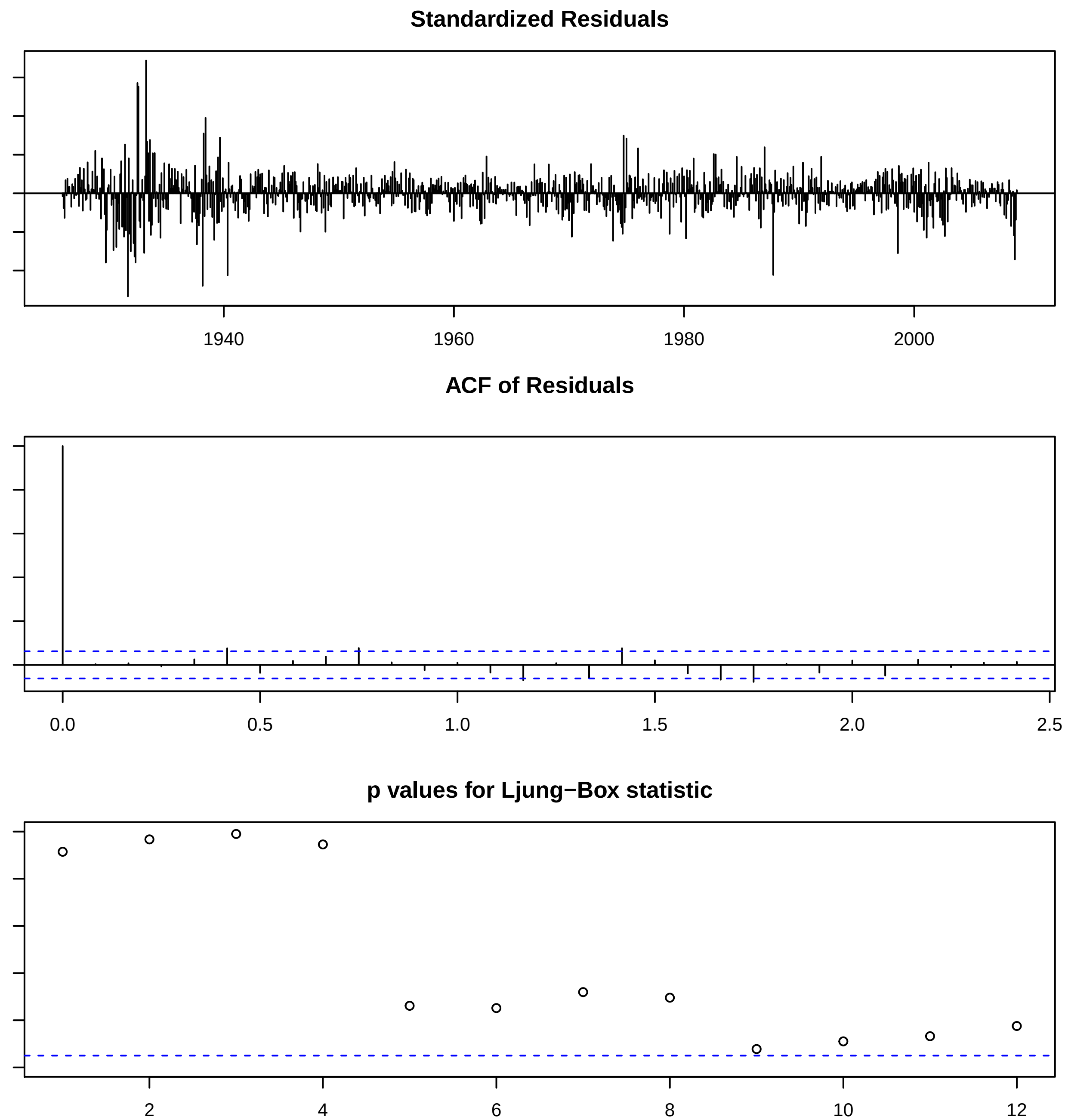
The textbook suggest using the chi-squared distribution with the degree-of-freedom adjustment for the Ljung-Box statisitcs.

```
m <- 12
g <- length(coef(vw_ar3)) - 1
vw_ar3_lbtest <- Box.test(vw_ar3_resid, lag = m, type = "Ljung")
pv_ar3 <- 1 - pchisq(vw_ar3_lbtest$statistic, m-g)
```

The p-value is 0.0599 greater than the 5% significant level so that we cannot reject the null hypothesis that the residuals are white noises, implying that the AR(3) model is adequate.

Also, there is a function called `tsdiag` to check the residuals from a model estimation.

```
op <- par(mfrow = c(3, 1), mar=c(3, 1, 3, 1), pty = "m")
tsdiag(vw_ar3, 12)
```

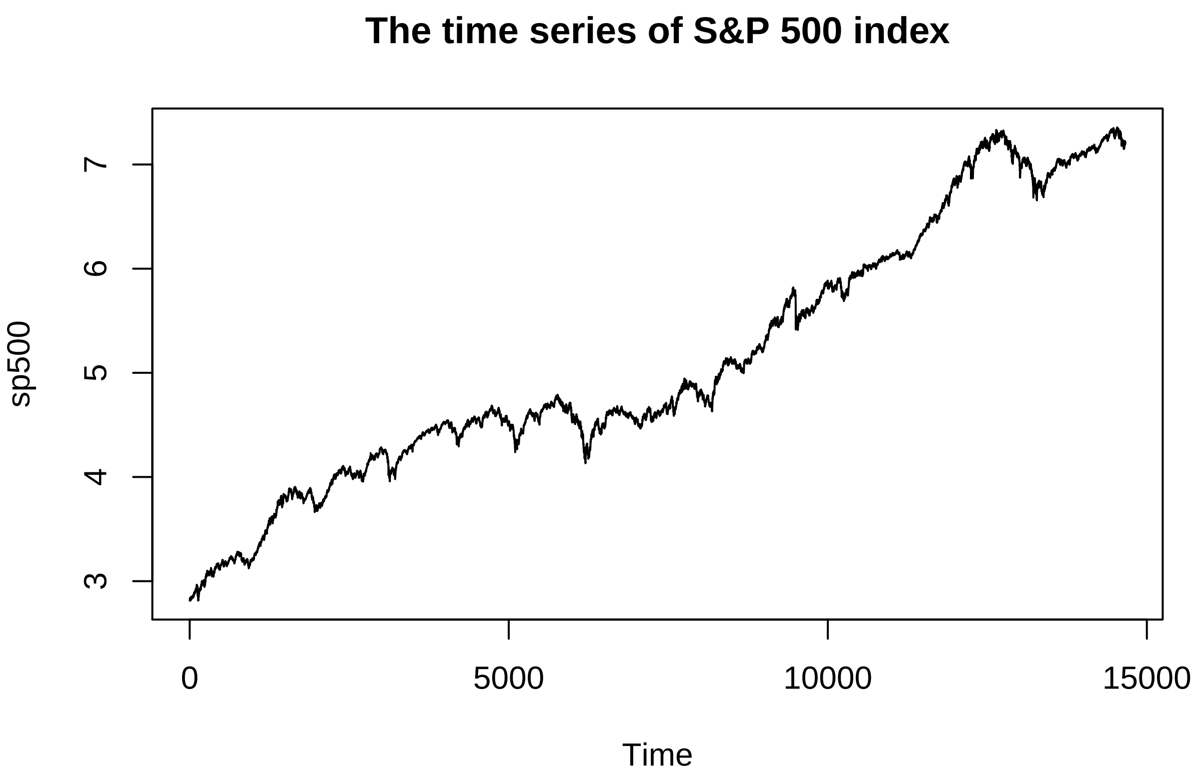



```
par(op)
```

3 Unit-Root Non-Stationarity

Finally, we demonstrate how to use R to examine a unit-root test with the example of the S&P 500 index.

```
sp_data <- read.table("d-sp55008.txt", header=TRUE)
sp500 <- ts(log(sp_data[, 7]))
plot(sp500, main="The time series of S&P 500 index")
```



The series appears to be non-stationary and have a time trend. We can use the augmented Dickey-Fuller test to check the existence of unit-roots, and model the series with a time trend.

```
library(fUnitRoots, quietly = TRUE)
```

```
##
## Attaching package: 'timeSeries'

## The following object is masked from 'package:zoo':
##
##     time<-

##

## Rmetrics Package fBasics

## Analysing Markets and calculating Basic Statistics

## Copyright (C) 2005-2014 Rmetrics Association Zurich

## Educational Software for Financial Engineering and Computational Science

## Rmetrics is free software and comes with ABSOLUTELY NO WARRANTY.

## https://www.rmetrics.org --- Mail to: info@rmetrics.org

##
## Attaching package: 'fBasics'

## The following object is masked from 'package:TTR':
##
##     volatility

##
## Attaching package: 'fUnitRoots'
```

```
## The following objects are masked from 'package:urca':
##
##     punitroot, qunitroot, unitrootTable
```

```r
# Carry out the Dickey-Fuller test with a constant and time trend
adfTest(sp500, lags=2, type=("ct"))
```

```
##
## Title:
##  Augmented Dickey-Fuller Test
##
## Test Results:
##   PARAMETER:
##     Lag Order: 2
##   STATISTIC:
##     Dickey-Fuller: -2.0179
##   P VALUE:
##     0.5708
##
## Description:
##  Mon Apr 17 11:03:18 2017 by user:
```

```r
# Model the series with time trend
dsp500 <- diff(sp500)
tdx <- 1:length(dsp500)
m3 <- arima(dsp500, order=c(2,0,0), xreg=tdx)
m3
```

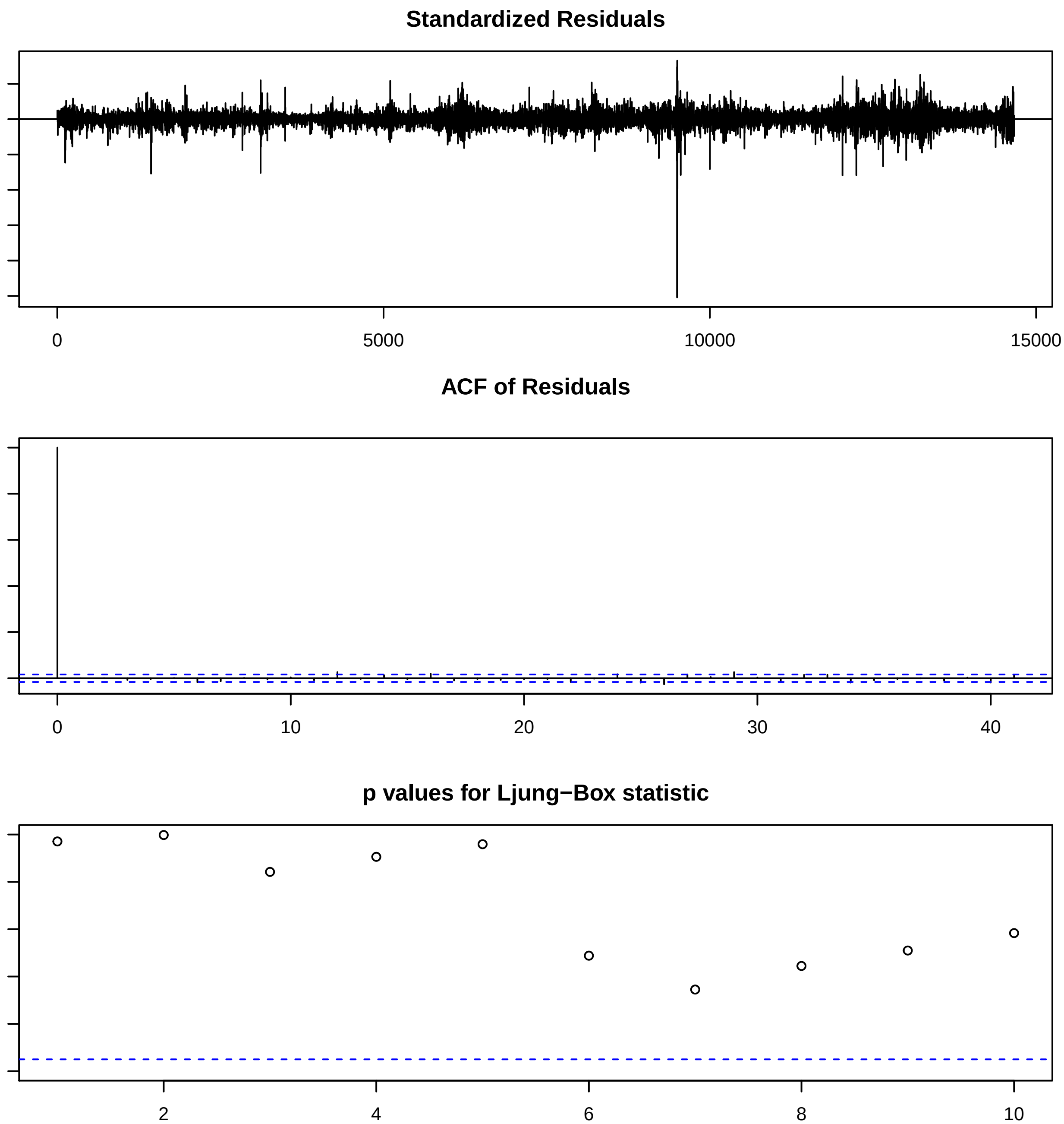
```
##
## Call:
## arima(x = dsp500, order = c(2, 0, 0), xreg = tdx)
##
## Coefficients:
##          ar1      ar2  intercept  tdx
##       0.0721  -0.0387      4e-04    0
## s.e.  0.0083   0.0083      2e-04    0
##
## sigma^2 estimated as 8.068e-05:  log likelihood = 48286.95,  aic = -96563.91
```

We then check the model adequacy with t statistics and the Ljung-Box test for the residuals.

```r
# check the t statistics
tratio <- m3$coef / sqrt(diag(m3$var.coef))# compute t-ratio
tratio
```

```
##          ar1          ar2    intercept          tdx
##  8.683863146 -4.669296791  2.285809057 -0.000858162
```

```r
# check the residuals
op <- par(mfrow = c(3, 1), mar=c(3, 1, 3, 1), pty = "m")
tsdiag(m3)
```


Standardized Residuals
0
5000
10000
15000
ACF of Residuals
0
10
20
30
40
p values for Ljung−Box statistic
2
4
6
8
10

```r
par(op)
```