CSC425 Lab1

Load libraries

```
#tseries: Time Series Analysis and Computational Finance
library(tseries)
#zoo: S3 Infrastructure for Regular and Irregular Time Series (Z's Ordered Observations)
library(zoo)
```

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

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

About ZOO, An S3 class with methods for totally ordered indexed observations. It is particularly aimed at irregular time series of numeric vectors/matrices and factors. zoo's key design goals are independence of a particular index/date/time class and consistency with ts and base R by providing methods to extend standard generics.

set the working directory and import dataset into a dataframe

```
setwd("~/Desktop/CSC425/week1")
cisco = read.table('cisco_00-10.csv', header = T, sep = ',')
# checking the head
head(cisco)
```

```
##       Date Price
## 1 12/31/10 20.23
## 2 12/30/10 20.23
## 3 12/29/10 20.25
## 4 12/28/10 20.35
## 5 12/27/10 20.16
## 6 12/23/10 19.69
```

create time series for cisco "prices"

```
ciscots = zoo(cisco$Price, as.Date(as.character(cisco$Date), format = "%m/%d/%y"))
```

```
## Warning in strptime(x, format, tz = "GMT"): unknown timezone 'default/
## America/Chicago'
```

```
head(ciscots)
```

```
## 2000-01-03 2000-01-04 2000-01-05 2000-01-06 2000-01-07 2000-01-10
##      54.03      51.00      50.85      50.00      52.94      54.90
```

```
# To retrieve only dates use
head(time(ciscots),10)
```

```
##  [1] "2000-01-03" "2000-01-04" "2000-01-05" "2000-01-06" "2000-01-07"
##  [6] "2000-01-10" "2000-01-11" "2000-01-12" "2000-01-13" "2000-01-14"
```

```
tail(time(ciscots),10)
```

```
##  [1] "2010-12-17" "2010-12-20" "2010-12-21" "2010-12-22" "2010-12-23"
##  [6] "2010-12-27" "2010-12-28" "2010-12-29" "2010-12-30" "2010-12-31"
```

```
# Retrieve start date
start(ciscots)
```

```
## [1] "2000-01-03"
```

```
# Retrieve End date
end(ciscots)
```

```
## [1] "2010-12-31"
```

Sort data in chronological order

```
# set variable Date as time/date variable
cisco$Date = as.Date(as.character(cisco$Date), format = "%m/%d/%y")
cisco = cisco[order(cisco$Date),]
head(cisco)
```

```
##            Date Price
## 2767 2000-01-03 54.03
## 2766 2000-01-04 51.00
## 2765 2000-01-05 50.85
## 2764 2000-01-06 50.00
## 2763 2000-01-07 52.94
## 2762 2000-01-10 54.90
```

Creating new Variables

```
# create lagged series using function lab(tsobject, k==1);
pricelag = lag(ciscots, k = -1);
head(pricelag)
```

```
## 2000-01-04 2000-01-05 2000-01-06 2000-01-07 2000-01-10 2000-01-11
##      54.03      51.00      50.85      50.00      52.94      54.90
```

```
#notice that "2000-01-03" has removed to be lagged
```

diff = p_t - p_(t-1)

```
pricedif = diff(ciscots);
head(pricedif)
```

```
## 2000-01-04 2000-01-05 2000-01-06 2000-01-07 2000-01-10 2000-01-11
##      -3.03      -0.15      -0.85       2.94       1.96      -1.65
```

compute simple returns ret = (p_t - p_(t-1) / p_(t-1))

```
ret = (ciscots - pricelag) / pricelag
head(ret)
```

```
##   2000-01-04   2000-01-05   2000-01-06   2000-01-07   2000-01-10
## -0.056079956 -0.002941176 -0.016715831  0.058800000  0.037023045
##   2000-01-11
## -0.030054645
```

Example of data analysis for cisco dataset

Define Log Returns

"rts" is a time series object since it is created from a TS object

```
rts = diff(log(ciscots))
head(rts)
```

```
##  2000-01-04  2000-01-05  2000-01-06  2000-01-07  2000-01-10  2000-01-11
## -0.05771382 -0.00294551 -0.01685712  0.05713619  0.03635415 -0.03051554
```

to retrieve numerical calues from time series use coredata()

"rt" is a numerical vector (no date information)

```
rt = coredata(rts)
head(rt)
```

```
## [1] -0.05771382 -0.00294551 -0.01685712  0.05713619  0.03635415 -0.03051554
```

Load Libraries

Load "fBasics" packages into current session

```
library(fBasics)
```

```
## Loading required package: timeDate
## Loading required package: timeSeries
##
## 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
```

Compute Summary Statistics

```
basicStats(rt)
```

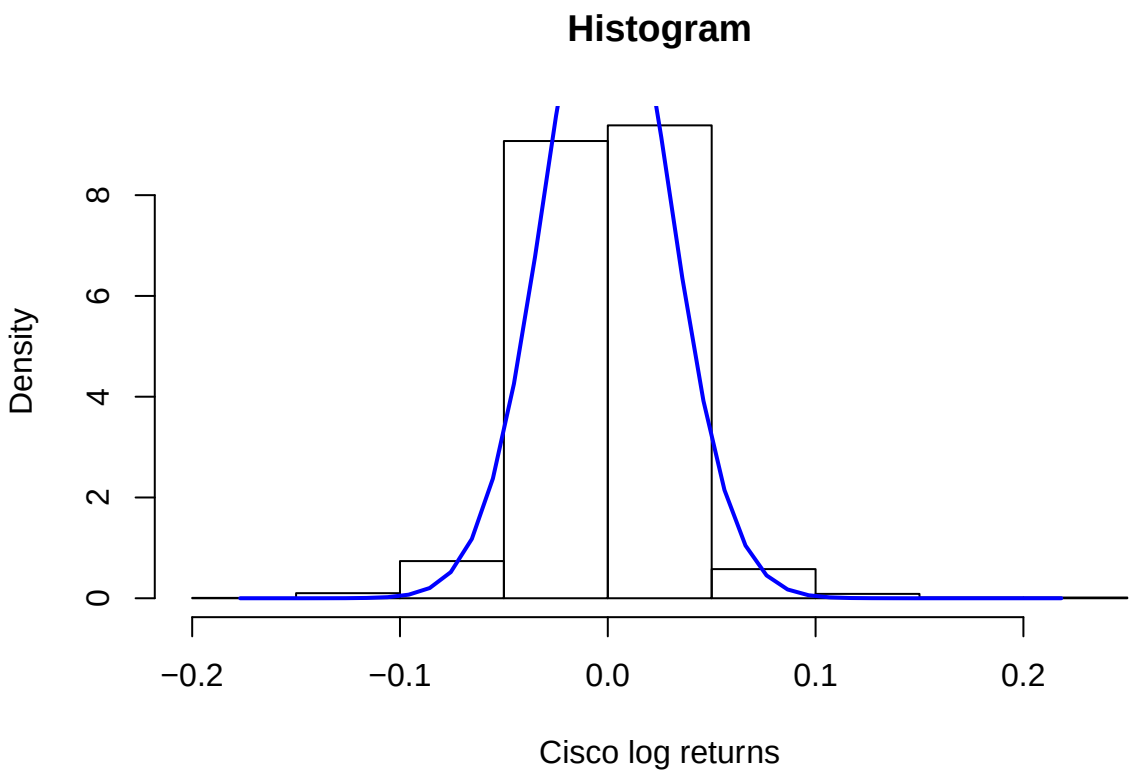
```
##                      rt
## nobs        2766.000000
## NAs            0.000000
## Minimum       -0.176865
## Maximum        0.218239
## 1. Quartile   -0.013890
## 3. Quartile    0.013411
## Mean          -0.000355
## Median         0.000449
## Sum           -0.982373
## SE Mean        0.000560
## LCL Mean      -0.001453
## UCL Mean       0.000742
## Variance       0.000867
## Stdev          0.029437
## Skewness       0.187810
## Kurtosis       6.053895
```

Create Histogram

Optional creates 2 by 2 display for 4 plots

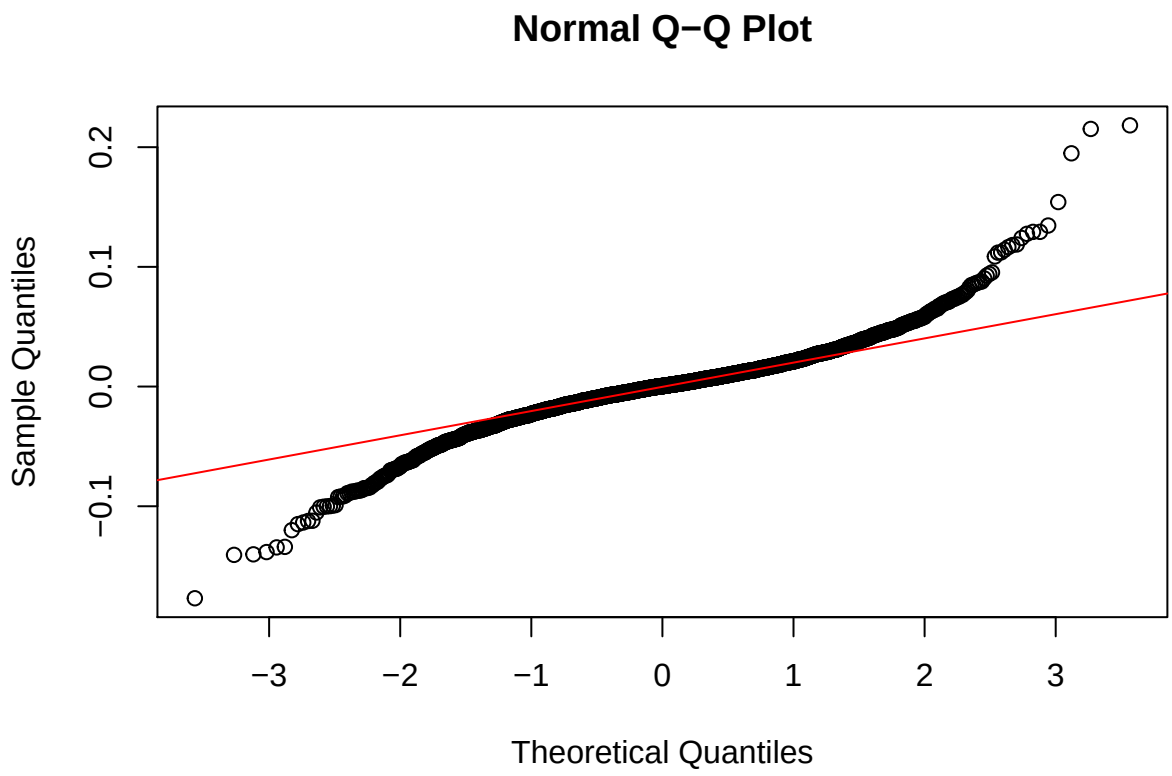
par(mfcol = c(2,2))

```
hist(rt, xlab = "Cisco log returns", prob = TRUE, main = "Histogram")
# add approximating normal density curve
xfit <- seq(min(rt), max(rt), length = 40)
yfit <- dnorm(xfit, mean = mean(rt), sd = sd(rt))
lines(xfit, yfit, col = "blue", lwd = 2)
```



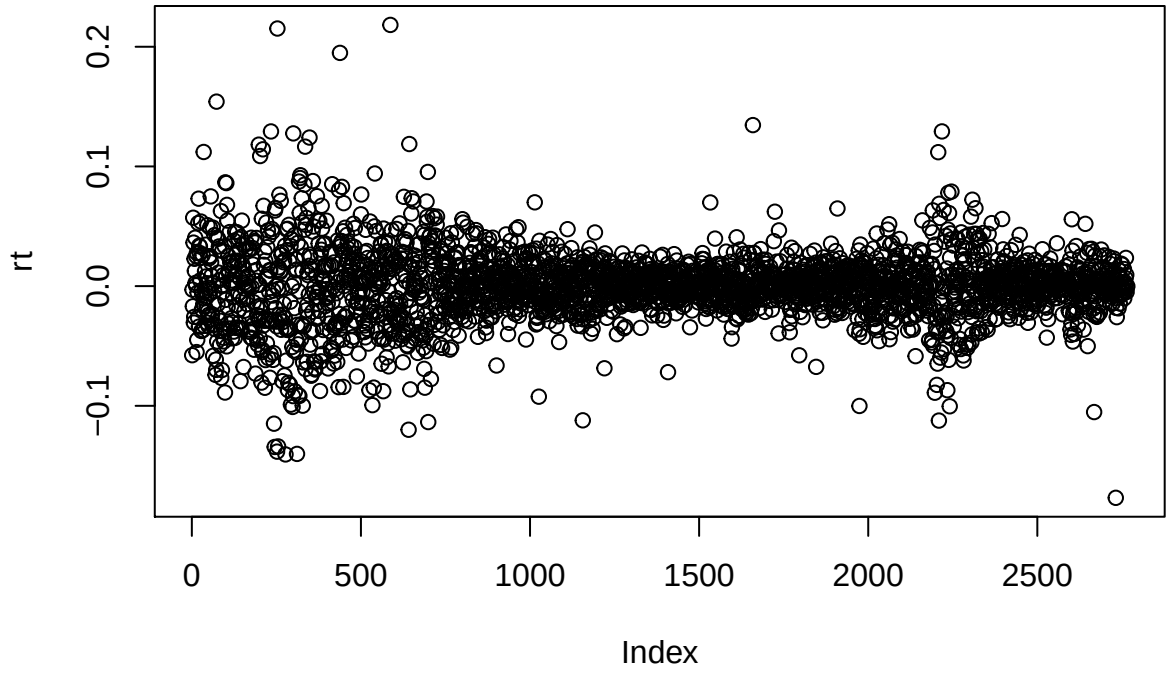
Create Normal Probabilty Plot

```
qqnorm(rt)
qqline(rt, col = 2)
```

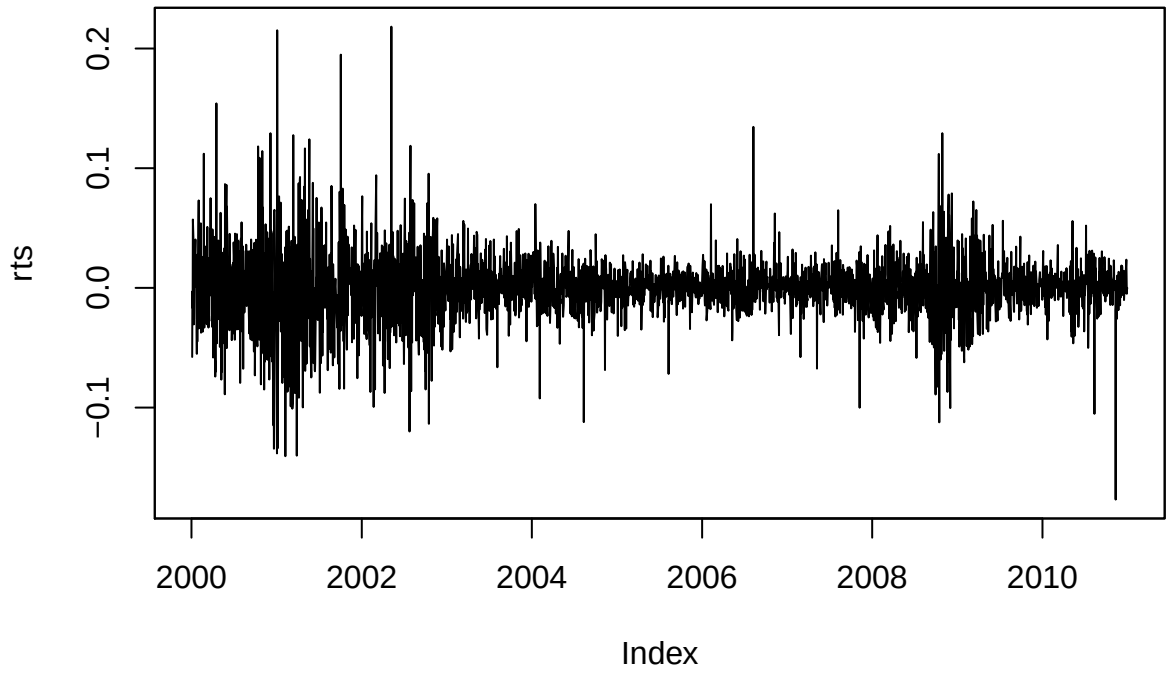


Create Time Plot

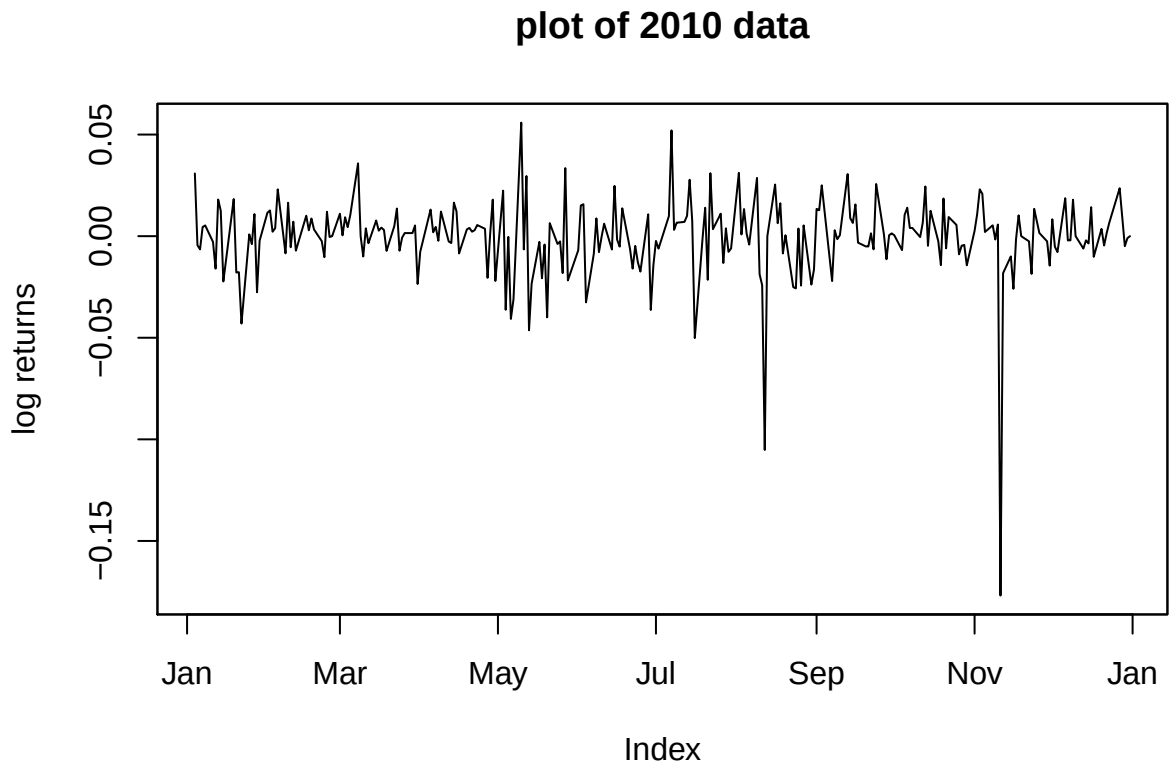
```
#simple plot where x-axis is not labeled with time
plot(rt)
```




```
#use time series object "rts" to draw time plot indexed with time
plot(rts)
```




```r
#creates subsets of data for a certain period of time
rts_10 = window(rts, start = as.Date("2010-01-01"), end = as.Date("2010-12-31"))
#plot the new subset
plot(rts_10, type = 'l', ylab = "log returns", main = "plot of 2010 data")
```



Normality Tests

```
#perform Jarque-Bera normality test
normalTest(rts, method = c("jb"))
```

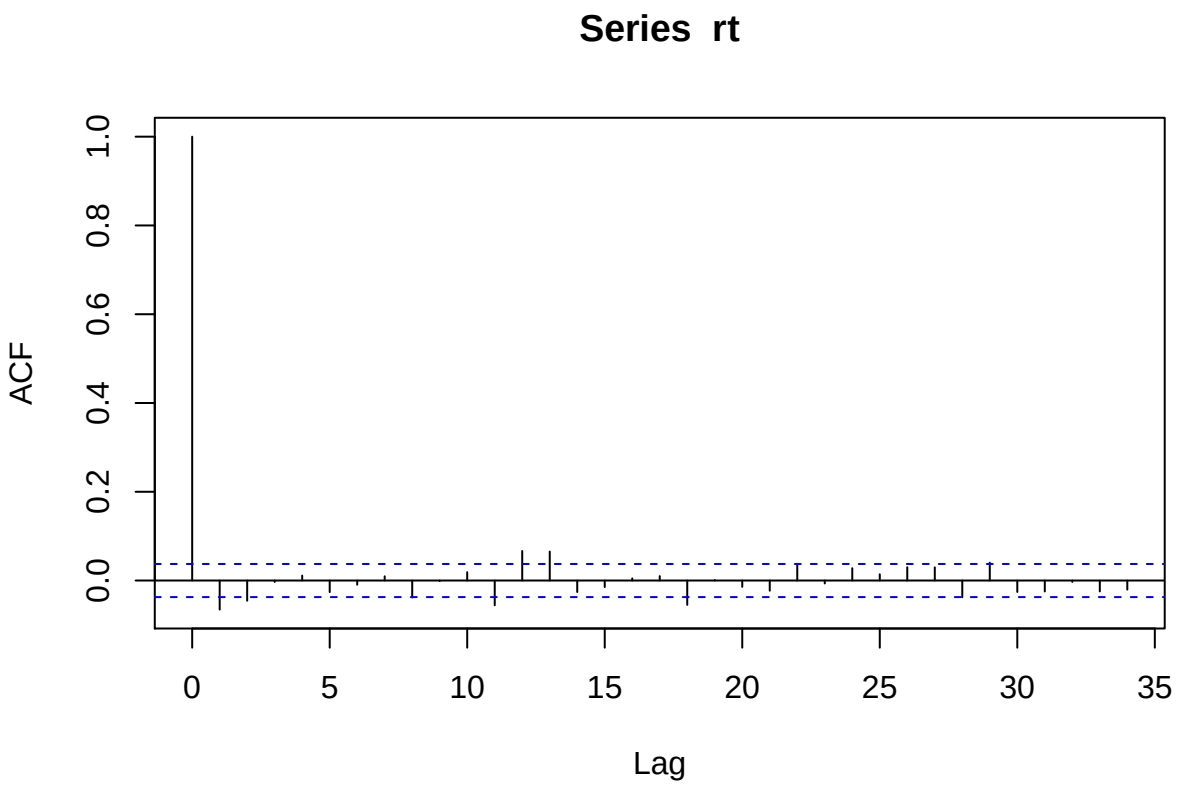
```
##
## Title:
##  Jarque - Bera Normalality Test
##
## Test Results:
##   STATISTIC:
##     X-squared: 4249.2945
##   P VALUE:
##     Asymptotic p Value: < 2.2e-16
##
## Description:
##  Sun Nov 19 20:52:19 2017 by user:
```

Compute ACF and Plot Correlogram

```
#prints acf values to console
acf(rt, plot = F)
```

```
##
## Autocorrelations of series 'rt', by lag
##
##      0      1      2      3      4      5      6      7      8      9
##  1.000 -0.065 -0.046 -0.003  0.011 -0.026 -0.009  0.010 -0.038 -0.001
##     10     11     12     13     14     15     16     17     18     19
##  0.019 -0.056  0.067  0.065 -0.026 -0.015  0.005  0.010 -0.055  0.001
##     20     21     22     23     24     25     26     27     28     29
## -0.014 -0.023  0.037 -0.007  0.028  0.014  0.030  0.030 -0.038  0.040
##     30     31     32     33     34
## -0.026 -0.025 -0.003 -0.025 -0.020
```

```
#plot acf values on graph (correlogram)
acf(rt, plot = T)
```



Compute LJUNG-BOX TEST for WHITE NOISE (NO AUTOCORRELATION)

```
# to lag 6
Box.test(rts, lag = 6, type = 'Ljung')
```

```
##
##  Box-Ljung test
##
## data:  rts
## X-squared = 38.638, df = 6, p-value = 8.427e-07
```

```
# to lag 12
Box.test(rts, lag = 12, type = 'Ljung')
```

```
##
##  Box-Ljung test
##
## data:  rts
## X-squared = 45.989, df = 12, p-value = 6.969e-06
```