Financial Econometrics

R Commands used in Lecture 2

Prof Hamed Ghoddusi

2019

The First Step

- Install R (or RStudio) on your computer
- Double click on R-icon to start R
- Make sure R is running before coming to the class

Libraries

We will use several libraries in this course.

- Install.packages("NAME of Package") % Installs new packages
  - Example: install.packages("tseries")
- Activate a library
  - library(quantmod) / require(quantmod)
  - require(tseries)
  - library(fBasics)

Data Input Option

- From the Internet (e.g. APIs)
- From an existing file (txt, csv, etc)
- Result of a simulation or calculations
- By the user

Commands: Get Data from Web

- library(quantmod) or require(quantmod)
  - load the package "quantmod"
- getSymbols("AAPL")
  - get daily Apple stock data from Yahoo Finance
- dim(AAPL)
  - find the size of the data downloaded
- head(AAPL)
  - show the first 6 rows of data
- tail(AAPL)
  - show the last 6 rows of data
- chartSeries(AAPL)
  - plot Apple daily closing stock prices with trading volume
  - Daily closing prices do not adjusted for stock split. You can use adjusted closing price.

Commands: Visualize

- chartSeries(AAPL[,6])
  - Column 6 of the object "AAPL" in R.
- chartSeries(AAPL[,6],theme="white")
  - Same as the previous command, but use "white" background for the plot.
- getSymbols("AAPL",from="2005-01-03",to="2013-12-31")
  - specify the data span
- getSymbols("INTC", src="yahoo") % src stands for "source",
- head(INTC)
- getSymbols("INTC", src="google")
- head(INTC)

计算方法不同

% Note the difference between Yahoo and Google Finance data

Commands: Data from FRED

- getSymbols("UNRATE",src="FRED")
  - Load U.S. monthly unemployment rate from Federal Reserve Bank of St Louis. % This command may not work on some versions
- chartSeries(UNRATE)
  - plot the U.S. monthly unemployment rate
- getSymbols("DEXUSEU",src="FRED")
  - Load Dollar verus Euro daily exchange rates from FRED.
- chartSeries(DEXUSEU)
  - plot the daily dollar-euro exchange rates.
- getSymbols("^VIX")
  - load daily VIX index
- getSymbols("^TNX")
  - load interest rate

An Alternative For Downloading FRED Data

***** Alternatively, You can use Quandl to Download FRED Data

```
install.packages("Quandl")
library(Quandl)
mydata = Quandl("FRED/GDP")
mydata = Quandl("FRED/GDP", start_date="2001-12-31”,end_date="2005-12-31")
mydata = Quandl("FRED/GDP", collapse="annual") % transform the data
mydata = Quandl("FRED/GDP", transform="rdiff") % transform the data
```

List of FRED data:

https://www.quandl.com/data/FRED-Federal-Reserve-Economic-Data

Commands: Read Data from Files

- setwd("…")
  - set my working directory.

You should use your own working directory

- library(fBasics)
  - Load the package "fBasics"
- da=read.table("m-ibm6708.txt",header=T)
  - Load data with header into R
  - **header=T** means the data file contains "names" for each column.
  - Use **header=F**, if the data file contains no column names.
- dim(da)
  - Check dimension of the data (row = sample size, col = variables)
- head(da)
  - Print out the first 6 rows of the data object "da".
- tail(da)
  - Print out the last 6 rows of the data object "da".
- ibm=da[,2] or ibm=da$ibm
  - Select the simple returns of IBM stock stored in Column 2.

Commands: Plot and Analyze Returns

- plot(ibm,type='l')
  - Plot the simple returns. Note that type is "ell" not 1.
- basicStats(ibm)
  - Compute the descriptive statistics of simple returns.
- libm=log(ibm+1)
  - Compute the IBM log returns
- basicStats(libm)
  - Compute descriptive statistics of log returns.
- t.test(ibm)
  - Perform t-test for mean being zero.
- t.test(ibm,alternative=c("greater"))
  - Perform one-sided test (Not shown in class)

hist(ibm,nclass=40)

- Obtain histogram of IBM simple returns.

- d1=density(libm)
  - Compute density function of ibm log returns

Commands

- names(d1)
  - Find out the output variables of the command "density".
- plot(d1$x,d1$y,type='l')
  - Plot the sample density of log returns
- mu=mean(libm); s1 = sd(libm)
  - compute the sample mean and standard deviation of IBM log returns.
- x=seq(-0.4,0.4,0.01)
  - create a sequence of real numbers from -0.4 to 0.4 with increment 0.01.
- y=dnorm(x,mean=mu,sd=s1)
  - obtain normal density with mean mu and standard deviation s1.
- lines(x,y,lty=2)
  - impose a dashed line on the density plot for comparison with normal density.
  - you can also use different colors in the plot. For example,
- lines(x,y,col="red") %will plot a red curve.
- normalTest(libm,method="jb")
  - Perform normality test.

Correlation, Kendall's tau and Spearman's rho

```
x = rnorm(1000)
```

- Generate 1000 N(0,1) random numbers

```
cor(x,x)
cor(x,exp(x))
cor(x,exp(x),method="kendall")
cor(x,exp(x),method="spearman")
cor(x,exp(20*x))
cor(x,exp(20*x),method="kendall")
cor(x,exp(20*x),method="spearman")
```

Downloading Energy Data from EIA

```
library('EIAdata')
% You need a private key to access EIA API
Register here:
https://www.eia.gov/opendata/

key <- "your_key" % This is your personal unique key; keep it safe

getEIA(ID, key) % Downloads data from EIA
```

Example: Downloads Monthly WTI Spot Prices, Calculate Returns, and Plot them

```
WTI=getEIA("PET.RWTC.M",key)  % Downloads monthly WTI prices
WTI_Price=WTI$PET.RWTC.M % Define a new variable
WTI_R=diff(log(WTI_Price)) % Calculate log returns
WTI_R=diff(log(WTI))
WTI_R1=na.omit(WTI_R)
plot(WTI_R,type='l') % Plot
 myts <- ts(WTI, start=c(2009, 1), end=c(2014, 12), frequency=12) % Selects a subsample
myts2 <- window(myts, start=c(2014, 6), end=c(2014, 12))
```

https://www.eia.gov/opendata/qb.php?category=714757 % You can get the list of IDs for different petroleum spot prices from here

https://www.eia.gov/opendata/qb.php?category=241335

https://cran.r-project.org/web/packages/EIAdata/EIAdata.pdf % Reference manual

Example: Downloads Monthly WTI Spot Prices, Calculate Returns, and Plot them

```
getSymbols("AAPL")


Y1=AAPL$AAPL.Close
m1 <- ar(Y1)
Y2=diff(log(Y1))
acf(Y2[-1])
```

Commands: Distribution of Returns

- jarque.bera.test(WTI_R)

- x <- rnorm(100) % Obvious case
- jarque.bera.test(x)

- x <- runif(100) % % Obvious case
- jarque.bera.test(x))

Commands: Autocorrelation of Returns

- acf(WTI) % autocorrelation of returns
- acf(WTI_R1) % autocorrelation of returns
- pacf(WTI) % partial autocorrelation

Commands: Autocorrelation of Returns

```
> da=read.table("m-ibm6708.txt",header=T)
> par(mfcol=c(2,1))  <== Put two plots on a sinlge page
   c(2,1) means two rows and one column.
> ts.plot(da$ibm)
> ts.plot(da$sprtn)
> tdx=(2+c(1:502))/12+1967 <== create a calendar time index
    <== The data start at March 1967 so that the index is from 3:505
```

Commands: Ljung-Box Test

```
Box.test(x, lag = 1, type = c("Box-Pierce", "Ljung-Box"), fitdf = 0)
```

Arguments

x	a numeric vector or univariate time series.
lag	the statistic will be based on lag autocorrelation coefficients.
type	test to be performed: partial matching is used.
fitdf	number of degrees of freedom to be subtracted if x is a series of residuals.

Commands: Ljung-Box Test

```
Y=rnorm(1000)

plot(y)
 Box.test(y,lag=1,type="Ljung")

 Box.test(y,lag=3,type="Ljung")
```

Commands: Autocorrelation of Returns

```
> par(mfcol=c(1,1)) <== return to one plot per page
> plot(tdx,da$ibm,type='l', xlab='year',ylab='ibm',main="Monthly IBM
returns")
> acf(da$ibm)  <== Compute sample ACF
> acf(da$ibm,lag=20)  <== specify the number of ACF to compute
> Box.test(da$ibm,lag=10,type="Ljung") <== Compute Ljung-Box Q(m) statistics


> gnp = scan(file="dgnp82.txt")  <== load the gnp growth rates into R
                <== "scan" is another way to load data. It only works for
                    a single variable in the data file.
> ts.plot(gnp)
> acf(gnp)
```