การสอนโปรแกรม R

เฉลิมพงษ์ คงเจริญ

Why R?

- Powerful and flexible
- Free(open source)
- Extensive add-on software(package)
- Designed for statistical computing
- High level language

Installing R

- Install the latest version from: http//cran.r-project.org
- Install R Studio (Makes R easier) from http://www.rstudio.com

นอกจากนี้ เราสามารถใช้ Rstudio บน internet โดยไม่ต้องลงโปรแกรมผ่าน https://rstudio.cloud/

Working with R

- The R Console 'interprets' whatever you type
  - Calculator
  - Creating variables
  - Applying function
- "Analysis" Script+ Interactive Exploration
  - Static copy of what you did (reproducibility)
  - Try things out interactively, then add to your script
- R revolves around functions
  - Commands that take input, performs computations, and returns results
  - Many come with R Base (installation), but people write external functions you can download and use

R as a calculator

```
2+2
```

```
## [1] 4
```

```
2*4
```

```
## [1] 8
```

```
2^3
```

```
## [1] 8
```

```
2+(2*3)^2
```

```
## [1] 38
```

##R variables - You can create variables from within the R environment and from files on your computer - R uses "=" or "<-" to assign values to a variable name - Variable names are case-sensitive, i.e. X and x are different.

```
a<-2*3
2*a
```

```
## [1] 12
```

The 'combine' function – the function c( ) collects/combines/joins single R objects into a vector of R objects.

```
b<- c(2,3,4)
b
```

```
## [1] 2 3 4
```

#What is a function? The c( ) command is called a function: it takes inputs and gives an output. - In R, functions always go function(input) or name of function, then parentheses. The input can be many different things, such as function(x,y,z)

```
x<-c(5,9,13,4)
mean(x)
```

```
## [1] 7.75
```

```
y<-rnorm(10)
y
```

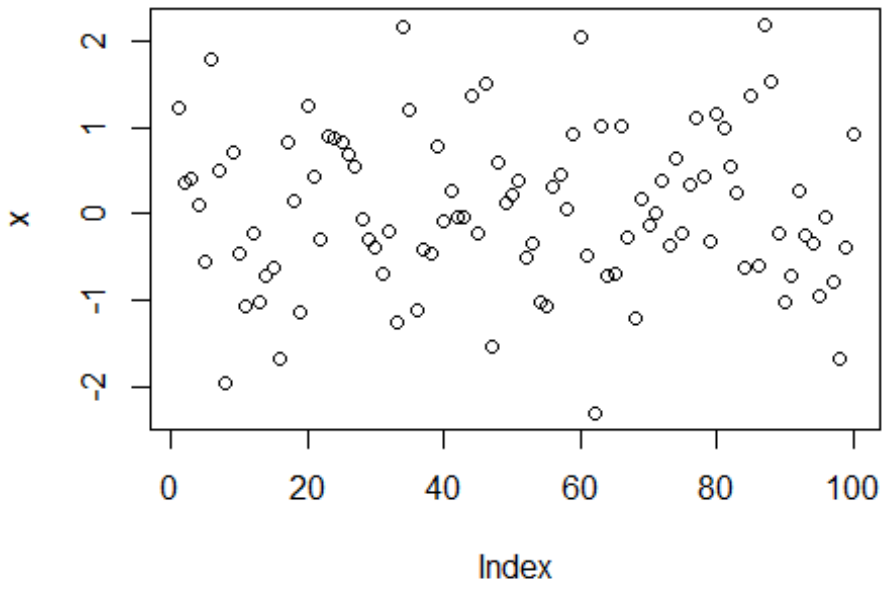
```
##  [1]  0.76862217 -0.25064770 -1.31171143 -1.08164804  1.33823005 -0.419800
77
##  [7]  0.36038086  0.29927606 -0.34704021 -0.09999096
```

```
set.seed(123)
y<-rnorm(10)
y
```

```
##  [1] -0.56047565 -0.23017749  1.55870831  0.07050839  0.12928774  1.715064
99
##  [7]  0.46091621 -1.26506123 -0.68685285 -0.44566197
```

#Graphic

```
x<-rnorm(100)
plot(x)
```



#R Help If you know the name of a function, help(function name)

#Data Classes: One dimensional classes('vectors') - Character - Numeric - Integer - Factor - Logical Two dimensional classes - data.frame: traditional 'Excel' spreadsheet - Matrix: two-dimensional data, composed of rows and columns

##Vector functions

```
z<-1:100
head(z)

## [1] 1 2 3 4 5 6

tail(z)

## [1]  95  96  97  98  99 100

str(z)

##  int [1:100] 1 2 3 4 5 6 7 8 9 10 ...
```

##Data Subsetting

```
x1<- 10:20
length(x1)

## [1] 11

x1[1]

## [1] 10

x1[3:4]
```

```
## [1] 12 13
```

```
x1[c(1,5,7)]
```

```
## [1] 10 14 16
```

##Matrices

```
m<-1:9
m
```

```
## [1] 1 2 3 4 5 6 7 8 9
```

```
mat<-matrix(m,nrow=3)
mat
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
## [3,]    3    6    9
```

```
nrow(mat)
```

```
## [1] 3
```

```
ncol(mat)
```

```
## [1] 3
```

```
dim(mat)
```

```
## [1] 3 3
```

##Data Selection Matrices have tow 'slots' you can use to select data, which represent rows and columns, that are separated by a comma, so the syntax is matrix[row, column]

```
mat[1,1]
```

```
## [1] 1
```

```
mat[1, ]
```

```
## [1] 1 4 7
```

```
mat[ ,1]
```

```
## [1] 1 2 3
```

##Data Frame

```
t<-data.frame(x=c(1,2,3), y=c(10,20,30), z=c(100,200,300))
t
```

```
##   x  y   z
## 1 1 10 100
## 2 2 20 200
## 3 3 30 300
```

```r
mean(t$y)
```

```
## [1] 20
```

```r
mean(t[["z"]])
```

```
## [1] 200
```

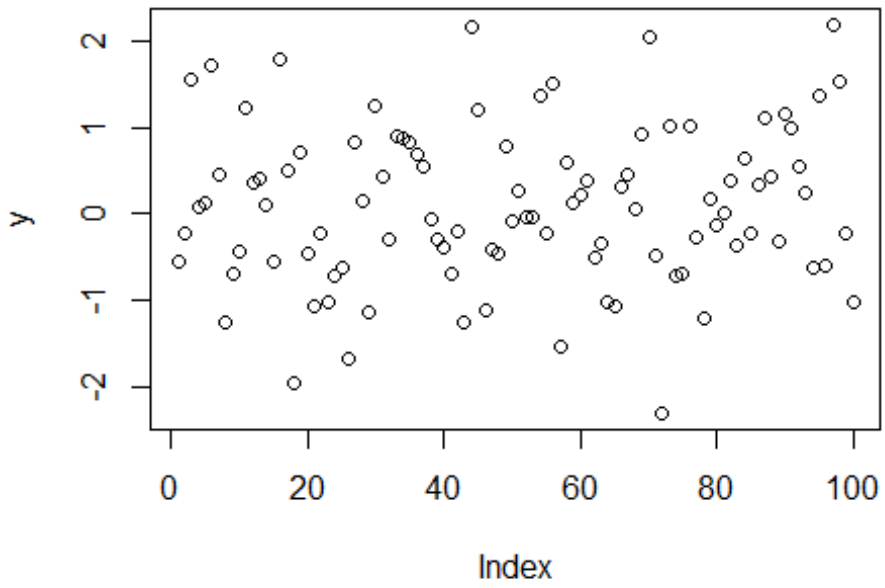
##List

```r
L<-list(one=1, two=c(1,2), five=seq(1,4, length=5))
L
```

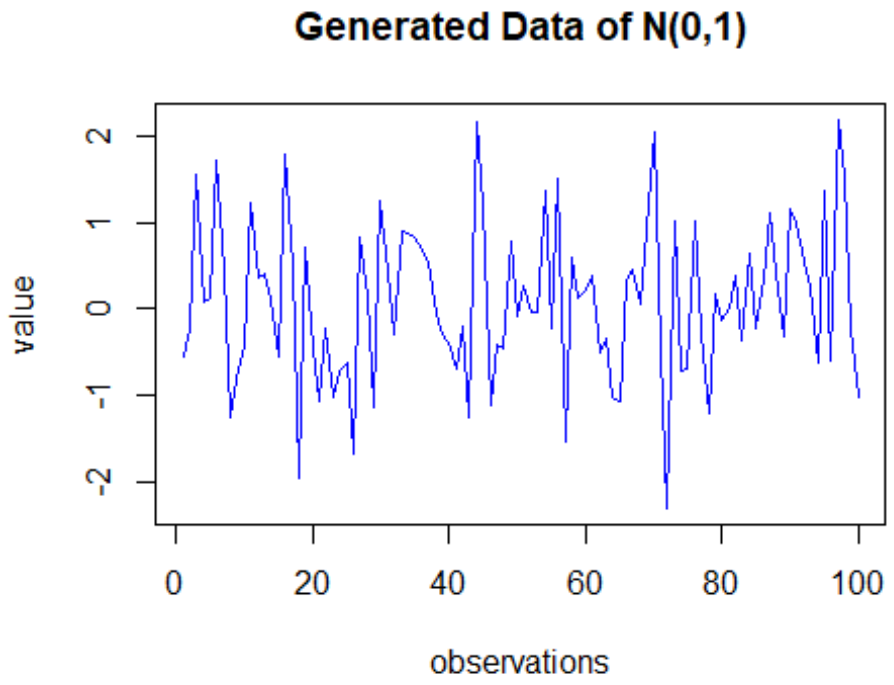
```
## $one
## [1] 1
##
## $two
## [1] 1 2
##
## $five
## [1] 1.00 1.75 2.50 3.25 4.00
```

##Graphic

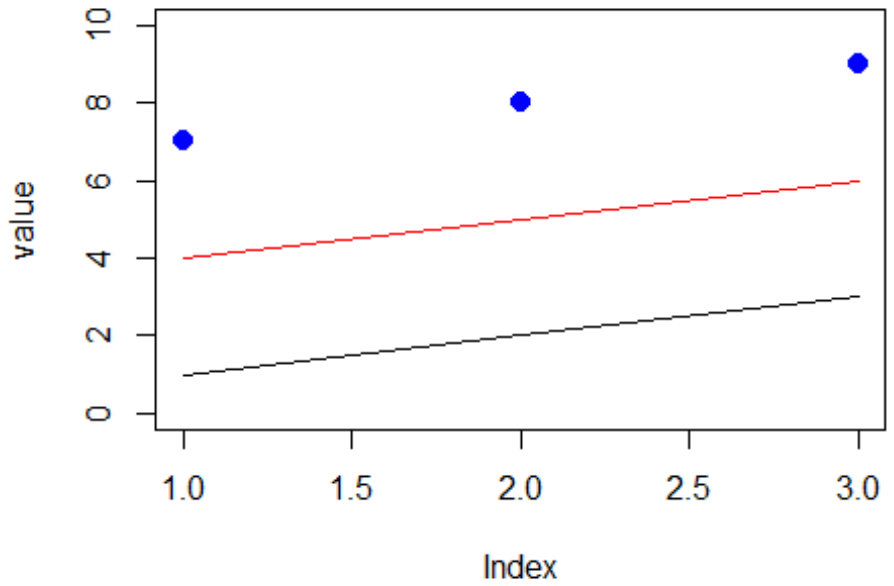
```r
set.seed(123)
y<-rnorm(100)
plot(y)
```




```r
plot(y, type="l", col="blue", ylab="value", xlab="observations", main="Genera
ted Data of N(0,1)")
```




```r
t<-data.frame(x=c(1,2,3), y=c(4,5,6), z=c(7,8,9))
plot(t$x, type="l",ylim=c(0,10), ylab="value")
lines(t$y, col="red")
points(t$z, pch=20, cex=2, col="blue")
```



```
ptt <- read.csv("https://raw.githubusercontent.com/chaleampong/EC435/master/ptt_d_02_19.csv", stringsAsFactors=FALSE)
class(ptt)
```

```
## [1] "data.frame"
```

```
names(ptt)
```

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

```
str(ptt)
```

```
## 'data.frame':    4390 obs. of  2 variables:
##  $ Date : chr  "1/2/2002" "1/4/2002" "1/7/2002" "1/8/2002" ...
##  $ Price: num  3.42 3.5 3.48 3.5 3.48 ...
```

```
head(ptt$Price)
```

```
## [1] 3.425 3.500 3.475 3.500 3.475 3.500
```

#More on Packages Packages are add-ons that are commonly written by users comprised of functions, data and vignettes. - Use library( ) or require( ) to load the package into memory so you can use its functions. - Install packages using install.packages("PackageName") . - Use help(package="PackageName") to see what contents the package has.

#Import Data In RStudio, we can import data files CSV, Excel, SPSS, SAS, Stata

#Saving R data It's very useful to be able to save collections of R objects for future analyses. Save(..., file="name.rda") When you close R, you might notice the prompt about saving your workspace in ".Rdata".

##Saving and Importing Data from text file

```
#Saving Data
d<-data.frame(a=c(3,4,5), b=c(12,13,14))
d
```

```
##   a  b
## 1 3 12
## 2 4 13
## 3 5 14
```

```
write.table(d, file="ts0.txt", row.names=FALSE)
#Importing Data
d2<-read.table(file="ts0.txt", header=TRUE)
d2
```

```
##   a  b
## 1 3 12
## 2 4 13
## 3 5 14
```

#การคำนวณผลได้ตอบแทน

```
ptt<-read.csv("G:/My Drive/teaching/ec435/book_project/data/ptt_d_02_19.csv")
head(ptt)
```

```
##        Date Price
## 1  1/2/2002 3.425
## 2  1/4/2002 3.500
## 3  1/7/2002 3.475
## 4  1/8/2002 3.500
## 5  1/9/2002 3.475
## 6 1/10/2002 3.500
```

```
ptt.lret<-diff(log(ptt$Price))
n<-length(ptt$Price)
ptt.sret<-(ptt$Price[2:n]-ptt$Price[1:n-1])/(ptt$Price[1:n-1])
ptt$sret<-c(NA,ptt.sret)
head(ptt)
```

```
##        Date Price         sret
## 1  1/2/2002 3.425           NA
## 2  1/4/2002 3.500  0.021897810
## 3  1/7/2002 3.475 -0.007142857
## 4  1/8/2002 3.500  0.007194245
## 5  1/9/2002 3.475 -0.007142857
## 6 1/10/2002 3.500  0.007194245
```

```
write.csv(ptt,file="ptt_ret.csv", row.names=FALSE)
```

##Adding to data frames

```
ptt$lret<-c(NA,ptt.lret)
head(ptt)
```

```
##        Date Price         sret         lret
## 1  1/2/2002 3.425           NA           NA
## 2  1/4/2002 3.500  0.021897810  0.021661497
## 3  1/7/2002 3.475 -0.007142857 -0.007168489
## 4  1/8/2002 3.500  0.007194245  0.007168489
## 5  1/9/2002 3.475 -0.007142857 -0.007168489
## 6 1/10/2002 3.500  0.007194245  0.007168489
```

#Adding rows and columns

```
m1<-matrix(1:9, nrow=3)
m2<-matrix(10:18, nrow=3)
cbind(m1,m2)
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    4    7   10   13   16
## [2,]    2    5    8   11   14   17
## [3,]    3    6    9   12   15   18
```

```
rbind(m1,m2)
```

```
##      [,1] [,2] [,3]
## [1,]    1    4    7
## [2,]    2    5    8
```

```
## [3,]    3    6    9
## [4,]   10   13   16
## [5,]   11   14   17
## [6,]   12   15   18
```

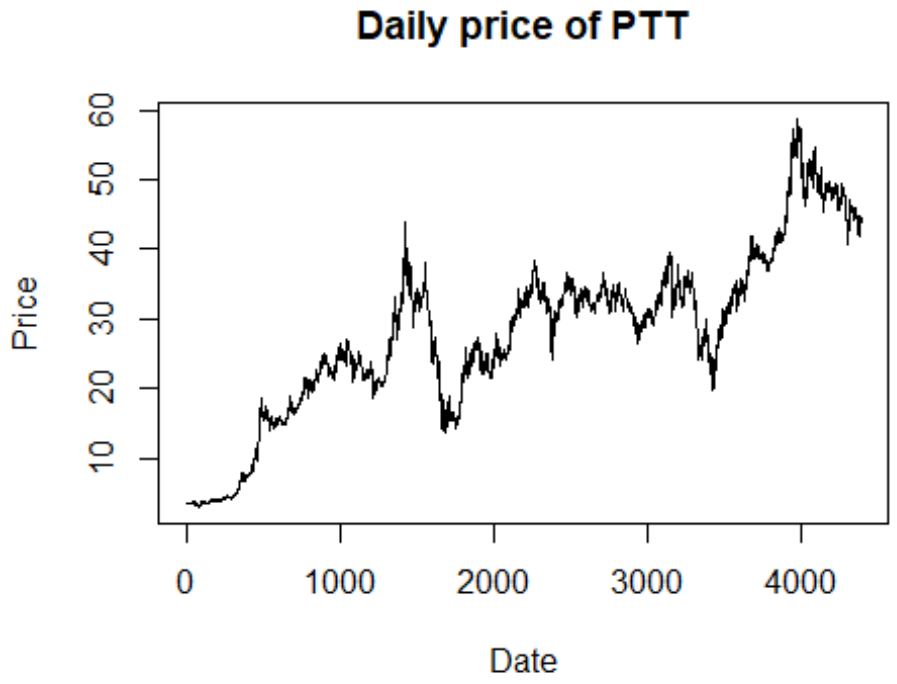
#Other manipulations - abs(x) - sqrt(x) - log(x) - log10(x) - exp(x)

#Data Summarization ##Basic statistical summarization - mean(x) - sd(x) - median(x) - quantile(x) - range(x)

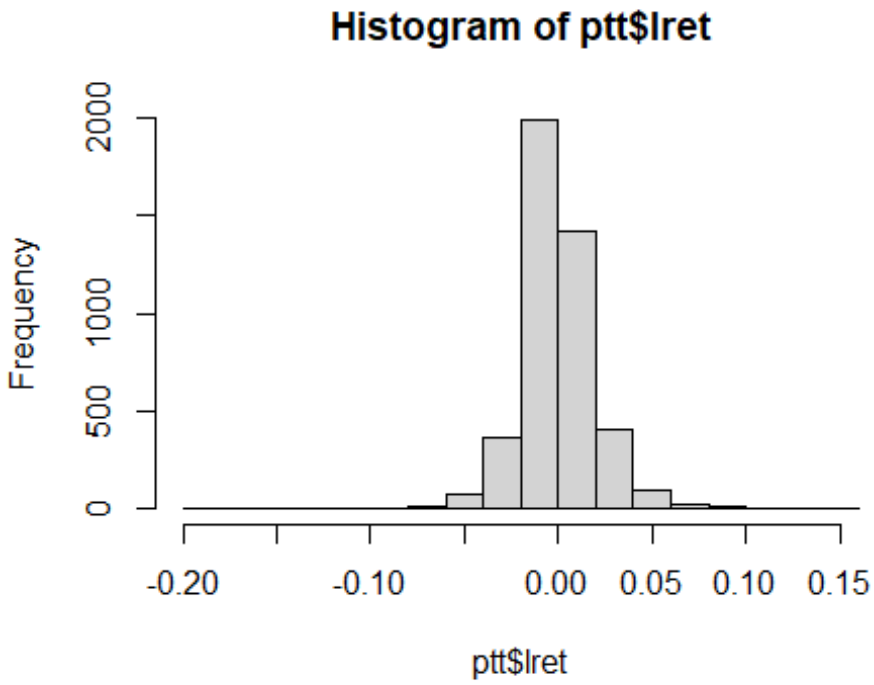
##Basic summarization plots - plot(x,y) - hist(x) - plot(density(x))

#Basic plot

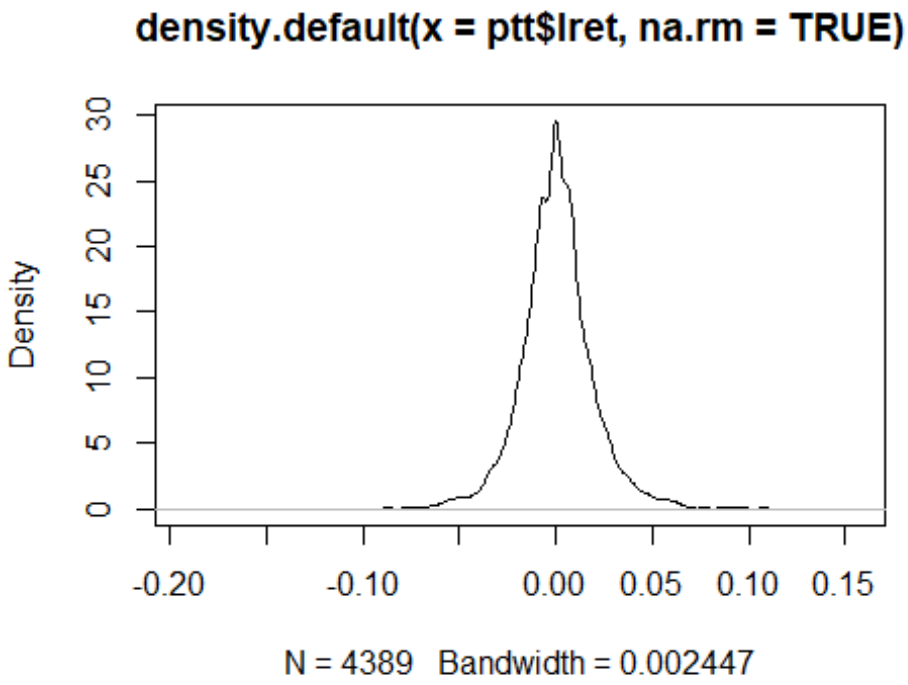
```
plot(ptt$Price, type="l", main="Daily price of PTT", xlab="Date",
ylab="Price")
```




```
hist(ptt$lret)
```




```
plot(density(ptt$lret, na.rm=TRUE))
```



#Probability Distributions - r for random number generation [e.g. rnorm( )] - d for density [e.g. dnorm( )] - p for probability [e.g. pnorm( )] - q for quantile [e.g. qnorm( )]

#Basic Statistics and tests

```r
mean(ptt$lret)
```

```
## [1] NA
```

```r
mean(ptt$lret, na.rm = TRUE)
```

```
## [1] 0.0005817016
```

```r
sd(ptt$lret, na.rm= TRUE)
```

```
## [1] 0.01967835
```

```r
library(fBasics)
```

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

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

```r
s3<-skewness(ptt$lret, na.rm= TRUE)
s3
```

```
## [1] -0.03538271
## attr(,"method")
## [1] "moment"
```

```r
T<-nrow(ptt)
t3<-s3/sqrt(6/T)
t3
```

```
## [1] -0.957079
## attr(,"method")
## [1] "moment"
```

```r
k4<-kurtosis(ptt$lret, na.rm= TRUE)
k4
```

```
## [1] 5.951371
## attr(,"method")
## [1] "excess"
```

```r
t4<-k4/sqrt(24/T)
t4
```

```
## [1] 80.49032
## attr(,"method")
## [1] "excess"
```

```r
normalTest(ptt$lret,method=c("jb"))
```

```
##
## Title:
##  Jarque - Bera Normalality Test
##
## Test Results:
##   STATISTIC:
##   P VALUE:
##
```

```
## Description:
##  Tue Sep 01 23:42:20 2020 by user: User
```

```
basicStats(ptt$lret)
```

```
##             X..ptt.lret
## nobs        4390.000000
## NAs            1.000000
## Minimum       -0.185899
## Maximum        0.149532
## 1. Quartile   -0.009479
## 3. Quartile    0.010017
## Mean           0.000582
## Median         0.000000
## Sum            2.553088
## SE Mean        0.000297
## LCL Mean      -0.000001
## UCL Mean       0.001164
## Variance       0.000387
## Stdev          0.019678
## Skewness      -0.035383
## Kurtosis       5.951371
```