제2강 : R 스크립트 프로그래밍과 패키지 사용/제작법

금융 통계 및 시계열 분석

TRADE INFORMATIX

2014년 1월 10일

Outline

1. 스크립트 프로그래밍
   - 함수 제작
   - 흐름 제어
   - 디버깅
   - 예외 처리
2. 객체지향 프로그래밍
   - S3 클래스 구현
   - S4 클래스 구현
3. 패키지 프로그래밍
   - 데이터셋
   - 데이터 입출력
   - R 패키지 제작
4. 추가적인 데이터구조
   - 날짜/시간
   - 시계열 자료형
5. 금융 데이터 관련 패키지
   - TTR
   - quantmod
   - Quandl

함수 제작

- `function.name <- function(arg1, arg2, ...) { expression }`
- `return()`이 없으면 마지막으로 계산된 expression 값을 리턴
- `...`는 가변길이 인수 (variable length arguments)
- `function(arg1=arg1_default)` 과 같이 인수의 디폴트 값 이용 가능

```
> func.add <- function(x, y) {
+   ans.sum <- x + y
+   ans.sum
+ }
> func.add(1, 2)
[1] 3
> func.inc <- function(x, y=1) {
+   ans.sum <- x + y
+   return(ans.sum)
+ }
> func.inc(1)
[1] 2
```

함수의 인수 (parameter) 매칭

- 인수 매칭은 이름 (name) 과 위치 (position) 를 이용
- `function.name(x=1, y=2)` 처럼 이름을 지정하여 함수를 호출하면 이름 정보를 이용하여 인수 매칭. 이름을 지정할 경우 인수의 위치는 어디에 오든 상관없음
- 이름 지정으로 매칭된 인수를 제외한 나머지 인수는 위치에 따라 매칭
- `...` 로 받은 인수는 `list(...)` 로 해석 (parsing)

```
> func1 <- function(a, b, c, d) {
+   ans <- sprintf("a=%s, b=%s, c=%s, d=%s",
+                  as.character(a), as.character(b),
+                  as.character(c), as.character(d))
+  return(ans)
+ }
> func1(1,2,3,4)

[1] "a=1, b=2, c=3, d=4"

> func1(b=1,2,c=3,4)

[1] "a=2, b=1, c=3, d=4"

> func2 <- function(...) {
+   args <- list(...)
+   ans <- paste(names(args), args, sep="=", collapse=",")
+   return(ans)
+ }
> func2(a=1, b=2, c=3)

[1] "a=1,b=2,c=3"
```

흐름 제어 (Flow Control)

- `if (condition) { true-expression } else { false-expression}` : 조건 분기
- `ifelse(vector, true-value, false-value)` : 벡터 조건 처리
- `switch (expression, input1=output1, input2=output2)` : 복수 조건 분기
- `for (var in sequence) { expression }` : 반복
- `while (condition) { expression }` : 조건 반복
- `next` : 이하 생략하고 다음 반복문 실행
- `break` : 루프 탈출

if else

- `if (condition) { true-expression } else { false-expression }`
- 조건 분기
- `condition`이 `TRUE`이면 `true-expression`이 실행,
  `condition`이 `FALSE`이면 `false-expression`이 실행

```
> x <- c(2,7,1,8,9)
> if (length(x) %% 2 == 0) {
+   median_value <- (sort(x)[length(x)/2]+sort(x)[1+length(x)/2])/2
+ } else {
+   median_value <- sort(x)[ceiling(length(x)/2)]
+ }
> median_value
[1] 7
```

ifelse

- `ifelse(vector, true-value, false-value)`
- 내재 루프(implicit loop)에 의한 벡터 조건 처리
- `vector`와 같은 크기의 벡터 출력
- `vector`의 모든 원소에 대해
  원소값이 `TRUE` 이면 출력벡터의 해당 원소값은 `true-value`에 해당하는 값이,
  원소값이 `FALSE` 이면 출력벡터의 해당 원소값은 `false-value`에 해당하는 값이 출력

```
> x <- c(6:-4)
> sqrt(ifelse(x >= 0, x, NA))
 [1] 2.449490 2.236068 2.000000 1.732051 1.414214 1.000000
 [7] 0.000000       NA       NA       NA       NA
```

switch

- 복수 조건 분기
- `switch (expression, input1=output1, input2=output2, ...)`
- `vector`의 모든 원소에 대해
  `expression` 값이 `input1` 이면 `output1` 값이,
  `expression` 값이 `input2` 이면 `output2` 값이 출력

```
> f1 <- function(type) {
+  if (type == "mean") 1
+  else if (type == "median") 2
+  else if (type == "trimmed") 3
+ }
> f2 <- function(type) {
+  switch(type,
+         mean = 1,
+         median = 2,
+         trimmed = 3)
+ }
> c(f1('mean'), f1('median'), f1('trimmed'))

[1] 1 2 3

> c(f2('mean'), f2('median'), f2('trimmed'))

[1] 1 2 3
```

for

- `for (var in sequence) { expression }`
- 벡터/리스트 `sequence`의 모든 원소에 대해 순차적 루프 반복
- `sequence` 의 각 원소를 `var`에 할당한 후 `expression` 실행

```
> x <- seq(1, 10, by=2)
> x.squared = NULL
> for (i in 1:length(x) ) {
+   x.squared[i] <- x[i]^2
+ }
> x.squared
[1]  1  9 25 49 81
```

while

- `while (condition) { expression }`
- 조건 반복. `condition` 값이 `TRUE`이면 `expression` 실행
- `next` : 이하 생략하고 다음 반복문 실행
- `break` : 루프 탈출

```
> x <- 0
> while(x<5) {print(x<-x+1)}

[1] 1
[1] 2
[1] 3
[1] 4
[1] 5

> x <- 0
> while(x < 5) {x <- x+1; if (x == 3) next; print(x); }

[1] 1
[1] 2
[1] 4
[1] 5

> x <- 0
> while(x < 5) {x <- x+1; if (x == 3) break; print(x);}

[1] 1
[1] 2
```

디버깅 (debugging)

- 함수 단위로 디버깅
- 디버그 설정이 된 함수는 호출되면 디버그 모드로 들어감
- `debug(function.name)`으로 함수 디버그 설정, `undebug(function.name)` 으로 디버그 해제.
- 함수 내부에 `browser()` 명령을 삽입하여 원하는 위치에서 디버그 모드에 들어갈 수 있음 (브레이크 포인트)
- `setBreakpoint(srcfile, line)`으로 외부에서 브레이크 포인트 설정 가능
- 디버그 모드 명령
  - `n` (next) : 다음 라인으로 진행
  - `c` (continue) : 다음 브레이크포인트로 진행
  - `where` : 현재 코드 표시
  - `Q` (Quite) : 중지

디버그 예제

❏ 함수 정의

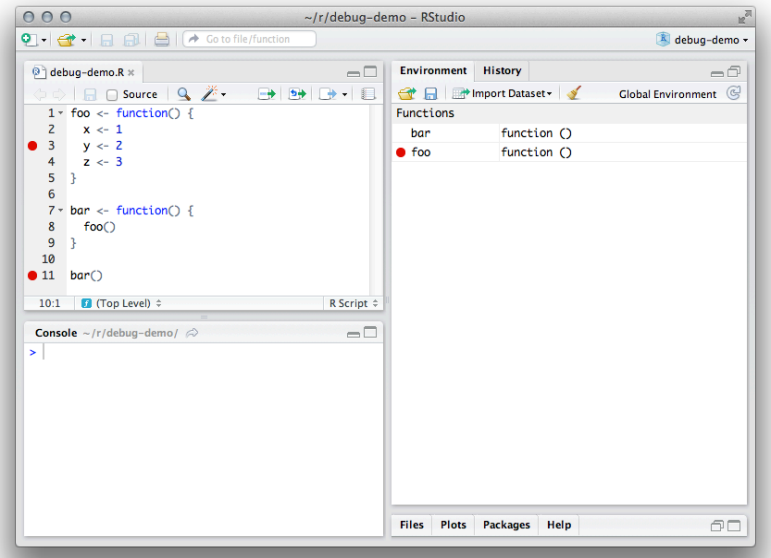
```
f1 <- function(x) {
  x1 <- 2 * x
  x2 <- 4 + x1
  x4 <- f2(x2)
  return(x4)
}

f2 <- function(x2) {
  x3 <- x2 / 2
  browser()
  x4 <- x3 - 2
  return(x4)
}
```

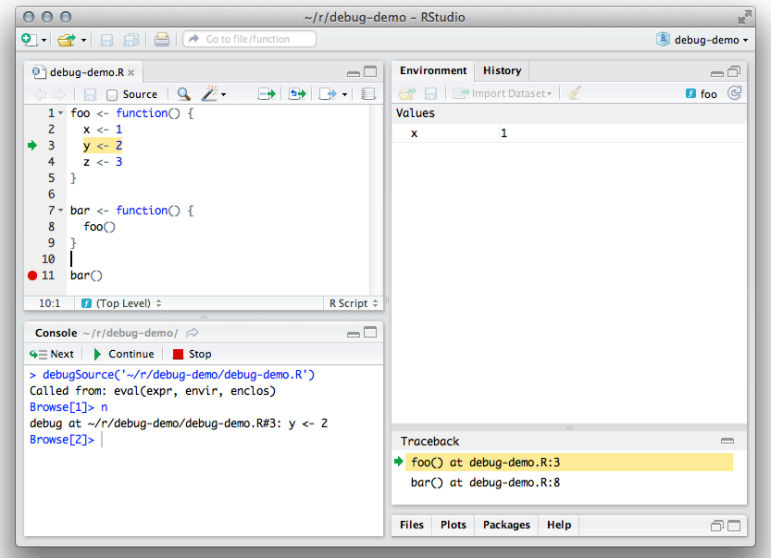
디버그 예제 (계속)

```
> debug(f1)
> f1(1)
debugging in: f1(1)
debug at ~/temp1.R#1: {
    x1 <- 2 * x
    x2 <- 4 + x1
    x4 <- f2(x2)
    return(x4)
}
Browse[2]> n
debug at ~/temp1.R#2: x1 <- 2 * x
Browse[2]> n
debug at ~/temp1.R#3: x2 <- 4 + x1
Browse[2]> ls()
[1] "x"  "x1"
Browse[2]> where
where 1: f1(1)

Browse[2]> c
Called from: f2(x2)
Browse[1]> where
where 1 at ~/temp1.R#4: f2(x2)
where 2: f1(1)

Browse[1]> n
debug at ~/temp1.R#11: x4 <- x3 - 2
Browse[2]> ls()
[1] "x2" "x3"
Browse[2]> Q
```

RStudio의 디버깅 기능

- RStudio는 GUI에 의한 디버그 모드 기능을 제공
  - 에디터 화면 클릭: 디버그 포인트 지정/해제
  - TraceBack 화면 제공: 현재 실행지점 표시
  - Next (F10)
  - Continue (Shift+F5)
  - Stop (Shift+F8)





예외처리 (Exception Handling)

- `warning()` : warning 강제 발생
- `stop()` : error 강제 발생

```
> mylog <- function(x) {
+   if (x == 0) {
+     warning("log of zero is minus infinity!")
+     return(-Inf)
+   } else {
+     if (x < 0) {
+       stop("log of negative value is not allowed!")
+     }
+   }
+ }
> mylog(0)
[1] -Inf
Warning message:
In mylog(0) : log of zero is minus infinity!
> mylog(-1)
Error in mylog(-1) : log of negative value is not allowed!
> log("a")
Error in log("a") : Non-numeric argument to mathematical function
```

예외처리 (Exception Handling)

❑ `try(expression, silent=TRUE)`: 단순 에러출력 방지

```
> inputs = list(1, -5, 'oops', 0, 10)
> for(input in inputs) {
+     print(paste("log of", input, "=", log(input)))
+ }
[1] "log of 1 = 0"
[1] "log of -5 = NaN"
Error in log(input) : Non-numeric argument to mathematical function
In addition: Warning message:
In log(input) : NaNs produced


> inputs = list(1, -5, 'oops', 0, 10)
> for(input in inputs) {
+   try(print(paste("log of", input, "=", log(input))))
+ }

[1] "log of 1 = 0"
[1] "log of -5 = NaN"
[1] "log of 0 = -Inf"
[1] "log of 10 = 2.30258509299405"
```

예외처리 (Exception Handling)

- `tryCatch(expression, warning, error, finally)`
- `warning`, `error`, `finally` 에는 각각
  warning 발생시의 핸들러, error 발생시의 핸들러, final 핸들러

```
> robustLog = function(x) {
+   tryCatch(log(x),
+     warning=function(w) {
+       print(paste("negative argument", x));
+       log(-x)
+     },
+     error=function(e) {
+       print(paste("non-numeric argument", x));
+       NaN
+     })
+ }
> inputs = list(1, -5, 'oops', 0, 10)
> for(input in inputs) {
+   print(paste("robust log of", input, "=", robustLog(input)))
+ }

[1] "robust log of 1 = 0"
[1] "negative argument -5"
[1] "robust log of -5 = 1.6094379124341"
[1] "non-numeric argument oops"
[1] "robust log of oops = NaN"
[1] "robust log of 0 = -Inf"
[1] "robust log of 10 = 2.30258509299405"
```

객체지향(OOP: Object Oriented Programming) 프로그래밍

- ❏ R도 다형성(polymorphism) 구현을 위한 OOP 구조가 있음 (S3 구현과 S4 구현)
- ❏ S3 구현은 클래스 정의 없이 리스트에 class 속성을 지정하는 방법
- ❏ S4 구현은 클래스 정의를 위한 메소드 제공

	S3 구현방식	S4 구현방식
클래스 정의	별도의 클래스 정의 없음 (리스트 생성시 멤버 설정)	`setClass()`
인스턴스 생성	리스트 생성후 클래스 속성 설정	`new()`
멤버 referencing	`$`	`@`
generic 함수 나열	`methods()`	`showMethods()`
generic 함수 구현	함수이름 이용 `f.classname`	`setMethod()`
generic 선언	`useMethod()` 명령 이용	`setGeneric()`

S3 클래스 구현

- 리스트 지정후 class 속성에 원하는 클래스 이름 설정
- generic 함수를 이용하면 함수 인수의 클래스에 따라 `func.classname` 이 호출.
  - 예 : generic 함수인 `print`에 `myclass` 클래스 인스턴스를 넣으면 실제로는 `print.myclass`가 호출됨
- generic 함수는 내부적으로 `UseMethod()` 명령 호출
- `methods()` 명령으로 generic 함수에 연결된 클래스 메소드 확인 가능

```
> print
function (x, ...)
UseMethod("print")
<bytecode: 0x6d556b8>
<environment: namespace:base>
> methods(print)
  [1] print.abbrev*
  [2] print.acf*
  [3] print.anova
  [4] print.Anova*
  [5] print.anova.loglm*
  [6] print.aov*
  (...omitted...)
```

S3 클래스 구현 (계속)

```
> x <- list(name="john", age=40)
> x
$name
[1] "john"

$age
[1] 40

> attr(x, "class") <- "employee"
> x
$name
[1] "john"

$age
[1] 40

attr(,"class")
[1] "employee"
> print.employee <- function(x) {
+   cat("name:", x$name, "\n")
+   cat("age:", x$age, "\n")
+ }
> x # print(x)
name: john
age: 40
```

S4 클래스 구현

- S4 클래스 본격적인 클래스 구현 방법을 제공
- S4 클래스의 generic 함수의 예 : `show()`
- `setClass(class.name, representation())` : 클래스 정의
  - `representation()` 함수에 멤버의 이름과 자료형을 입력
- `new(class.name)` : 클래스 인스턴스 생성
- `setMethod(generic.func.name, class.name, func.obj)` : generic 함수 구현
- `showMethods(generic.func.name)` : generic 함수의 구현 목록 확인 가능

```
> show
standardGeneric for "show" defined from package "methods"

function (object)
standardGeneric("show")
<bytecode: 0x1c2a4b28>
<environment: 0x1c210ba8>
Methods may be defined for arguments: object
Use  showMethods("show")  for currently available ones.
(This generic function excludes non-simple inheritance; see ?setIs)

> showMethods(show)
Function: show (package methods)
object="ANY"
object="classGeneratorFunction"
object="classRepresentation"
object="envRefClass"

(...omitted...)
```

S4 클래스 구현 (계속)

```
> setClass("employee", representation(name="character", age="integer"))
> x <- new("employee", name="john")
> x@age <- as.integer(40)
> x
An object of class "employee"
Slot "name":
[1] "john"

Slot "age":
[1] 40
> show.employee <- function(object) {
+   cat("name:", object@name, "\n")
+   cat("age:", object@age, "\n")
+ }
> setMethod("show", "employee", show.employee)
[1] "show"
> show(x)
name: john
age: 40
```

패키지

- **R 패키지**는 관련된
  - 함수
  - 문서
  - 데이터셋

  등을 모아놓은 것을 말한다. 일반적인 프로그래밍 랭귀지의 라이브러리에 해당
- 데이터, 함수, 문서 3가지를 모두 제공함으로써 재현가능한 연구 결과 (reproducible research) 제공
- R을 시동하면 자동으로 로드되는 기본 패키지
  - base, methods, utils, datasets, graphics, grDevices, stats

R 패키지 설치/업데이트 명령어

- 설치된 패키지의 확인

```
library()
installed.packages()
```

- CRAN에서 지원하는 패키지의 설치

```
install.packages("packagename")
```

- CRAN에서 지원하지 않는 기타 패키지 설치

```
install.packages("packagename", repo=repoaddress)
```

- 패키지 업데이트

```
update.packages()
```

R 패키지 로드/언로드 명령어

- 패키지 로드

```
library(packagename)
```

- 패키지 언로드

```
detach("package:packagename", unload=TRUE)
```

- 현재 로드된 패키지 목록의 확인

```
search()
```

데이터셋

- 현재 로드된 모든 라이브러리가 제공하는 데이터셋 목록 확인

```
data()
```

- 특정 패키지 (로드되었든 로드되지 않았든 상관없음) 가 제공하는 데이터셋 목록 확인

```
data(package="packagename")
```

- 현재 로드된 라이브러리가 제공하는 데이터셋을 워크스페이스에 로드

```
data(dataname)
data("dataname", package="packagename")
```

  - 라이브러리가 로드되면 포함된 데이터셋은 바로 사용가능하지만 워크스페이스에는 로드되지 않음. `data` 명령을 사용하여 명시적으로 워크스페이스에 로드해야 함
  - 이미 로드된 라이브러리에 포함된 데이터셋을 워크스페이스에 로드하는 경우에는 라이브러리 이름을 명시할 필요가 없으며 데이터셋 이름에 따옴표 필요 없음

콘솔에서 데이터 입력

❑ `scan(what, sep)` 명령 : 오브젝트 값 입력. 입력을 마치려면 공백 + 엔터 (enter)

- ▶ `what=double()` : 오브젝트 타입 지정
- ▶ `sep=""` : 디폴트 원소 구분자는 공백 (space)

```
> x <- scan()
1: 1 2 3
4:
Read 3 items
> typeof(x)
[1] "double"
> y <- scan(what=integer())
1: 1 2 3
4:
Read 3 items
> typeof(y)
[1] "integer"
> z <- scan(what=character())
1: 1 2 3
4:
Read 3 items
> z
[1] "1" "2" "3"
> typeof(z)
[1] "character"
> a <- scan(what="", sep=",")
1: a b c, d e, f g h
4:
Read 3 items
> a
[1] "a b c"  " d e"   " f g h"
```

파일에서 데이터 읽기 (Import)

- `read.table(file, header, sep)` 명령 : 파일에서 데이타 읽기
  - `file` : 데이터 파일 패스
  - `header=FALSE` : `TRUE`이면 첫 행은 헤더라인
  - `sep=""` : 디폴트 원소 구분자는 공백 (space)
- `read.csv(file)` 명령 : csv 파일에서 데이타 읽기

```
bash-3.2$ cat test1.dat
index    x     y     z
  122    1   2.4   4.8
  123    2   4.2   8.4
  124    3   3.6   3.7
```

```
> x <- read.table("test1.dat", header=T)
> x
  index x   y   z
1   122 1 2.4 4.8
2   123 2 4.2 8.4
3   124 3 3.6 3.7
```

파일로 데이터 쓰기(Export)

- ❏ `write.table(x, file, sep)` 명령: 파일로 데이타 쓰기
- ❏ `write.csv(x, file)` 명령: CSV 파일로 데이타 쓰기

```
> x
  index x   y   z
1   122 1 2.4 4.8
2   123 2 4.2 8.4
3   124 3 3.6 3.7
> write.table(x, "test2.dat")
> write.csv(x, "test2.csv")
```

```
bash-3.2$ cat test2.dat
"index" "x" "y" "z"
"1" 122 1 2.4 4.8
"2" 123 2 4.2 8.4
"3" 124 3 3.6 3.7
bash-3.2$ cat test2.csv
"","index","x","y","z"
"1",122,1,2.4,4.8
"2",123,2,4.2,8.4
"3",124,3,3.6,3.7
```

R 패키지 작성 순서

1. package.skeleton 명령으로 패키지 디렉토리 구조 생성
2. DESCRIPTION 파일 작성
3. R/script-name.R 스크립트 파일들 작성
4. man/document-name.Rd 문서화 파일들 작성
5. R CMD check 명령으로 패키지 check
6. R CMD install 명령으로 패키지 build/install

R 패키지 디렉토리 구조 생성

- 패키지 구조

DESCRIPTION 파일	(필수) 패키지의 정보를 서술
R 서브디렉토리	(필수) 스크립트
man 서브디렉토리	(필수) 문서 및 헬프 정보
data 서브디렉토리	(옵션) 데이타

- `package.skeleton` 명령

```
> package.skeleton(name="mypackage")
Creating directories ...
Creating DESCRIPTION ...
Creating NAMESPACE ...
Creating Read-and-delete-me ...
Saving functions and data ...
Making help files ...
Done.
Further steps are described in './mypackage/Read-and-delete-me'
```

DESCRIPTION 파일 작성

❏ DESCRIPTION 파일 : 패키지의 핵심 정보를 서술하는 파일

Package	(필수) 패키지이름
Version	(필수) 버전
License	(필수) 라이센스
Description	(필수) 설명
Title	(필수) 문서 제목
Author	(필수) 저자
Maintainer	(필수) 관리자

❏ DESCRIPTION 파일의 예

```
Package: mypackage
Type: Package
Title: my test package
Version: 0.1-0
Date: 2013-09-04
Author: Kildong Hong
Maintainer: Kildong Hong <hong.kildong@gmail.com>
Description: long description about my first test package
License: GPL-2
```

스크립트 파일 작성

- R 서브디렉토리 아래에 임의의 이름의 .R 파일 작성
- 파일내에서 object (변수, 함수 등) 생성
- 스크립트의 예 : `/mypackage1/R/myadd.R`

```
myadd <- function(x) {
  x + 1
}
```

문서화 파일 작성

- man 서브디렉토리 아래에 임의의 이름의 .Rd 파일 작성
- 파일내에서 패키지/함수/데이터셋 등에 대한 문서화
- 문서화 파일의 예 : `/mypackage1/man/myadd.Rd`

```
\name{myadd}
\alias{myadd}
\title{myadd}
\description{myadd function returns x + 1}
\usage{myadd(x)}
\arguments{
  \item{x}{input argument. numeric value}
}
\details{myadd function returns x + 1}
\value{x + 1}
\examples{
myadd(1)
}
```

패키지 체크

- 콘솔에서 `R CMD check packagename` 명령
- 스크립트 코드 및 문서화 파일의 오류 체크

```
* using R version 2.15.2 (2012-10-26)
* using platform: x86_64-redhat-linux-gnu (64-bit)
* using session charset: UTF-8
* checking for file ‘mypackage/DESCRIPTION’ ... OK
(... omitted ...)
* checking Rd files ... OK
* checking Rd metadata ... OK
* checking Rd cross-references ... OK
* checking for missing documentation entries ... OK
* checking for code/documentation mismatches ... OK
* checking Rd \usage sections ... OK
* checking Rd contents ... OK
* checking for unstated dependencies in examples ... OK
* checking examples ... OK
* checking PDF version of manual ... OK

R CMD check succeeded
```

패키지 빌드 및 설치

❏ 체크가 완료되면 `R CMD INSTALL packagename` 명령으로 빌드 및 설치

```
* installing *source* package ‘mypackage’ ...
** R
** preparing package for lazy loading
** help
  converting help for package ‘mypackage’
*** installing help indices
    finding HTML links ...    myadd                                   html
 done
** building package indices
** testing if installed package can be loaded

* DONE (mypackage)
```

❏ 설치후 `library(packagename)` 명령으로 로드하여 확인

```
> library(mypackage)
> myadd(1)
[1] 2
```

날짜/시간

- `Date` 기본적으로 제공하는 날짜 자료형
  - 날짜까지만 표현 가능
- `POSIXct, POSIXlt` 기본적으로 제공하는 날짜/시간 자료형
  - POSIX 표준에 따른 날짜/시간 자료형
  - 1970-01-01 자정 기준으로 초단위 표현 (소숫점, 음수 가능)
  - 시간대를 표시하는 `tzone` 속성을 가짐
  - `POSIXct`는 초단위 double 벡터
  - `POSIXlt`는 년/월/일/초/요일 등의 원소를 가지는 리스트

```
> d1 <- as.Date("2013/01/01")
> d1
[1] "2013-01-01"
> typeof(d1)
[1] "double"
> class(d1)
[1] "Date"
> format.Date(d1, "%Y %m %d")
[1] "2013 01 01"
> d2 <- as.Date("2013-01-02")
> d1 > d2
[1] FALSE
> d2 + 1
[1] "2013-01-03"
```

Date 자료형

- ❏ as.Date(date.string, format='%m/%d/%Y') 문자열에서 날짜 자료형 생성
- ❏ format.Date(date, format) 날짜 자료형을 문자열로 변환
- ❏ Sys.Date() 현재 날짜 출력
- ❏ +/-/비교 연산 가능. Date 자료형의 차이는 difftime 자료형
- ❏ difftime 자료형의 숫자값은 as.numeric(dt, units) 명령으로 계산

```
> d1 <- as.Date("2013/01/01")
> typeof(d1)
[1] "double"
> class(d1)
[1] "Date"
> format.Date(d1, "%Y %m %d")
[1] "2013 01 01"
> d2 <- as.Date("2013-01-02")
> d1 > d2
[1] FALSE
> d2 + 1
[1] "2013-01-03"
> d <- d2 - d1
> d
Time difference of 1 days
> str(td)
Class 'difftime'  atomic [1:1] 1
  ..- attr(*, "units")= chr "days"
> as.numeric(d)
[1] 1
> as.numeric(d, units="hours")
[1] 24
```

POSIXct/POSIXlt 자료형

❑ Date 기본적으로 제공하는 날짜/시간 자료형

- ▶ `as.POSIXct(datetime, tz)`, `as.POSIXlt(datetime)` : 문자열에서 POSIXct/POSIXlt 자료형 생성
- ▶ `format.POSIXct(t, format)`, `format.POSIXlt(t, format)` : 날짜 자료형을 문자열로 변환
- ▶ `Sys.time()` 현재 날짜/시간
- ▶ `year`, `mon`, `year`, `mday`, `hour` 등의 원소는 년/월/일/시/분 등을 나타내지만 기준 시점이 다르기때문에 사용이 어렵다.
- ▶ 이런 속성은 `as.numeric(format.POSIXct())`을 사용하여 구하는 것이 더 쉬움

```
> t1 <- as.POSIXct("2013-01-01")
> t1

[1] "2013-01-01 KST"

> unclass(t1)

[1] 1356966000
attr(,"tzone")
[1] ""

> t2 <- as.POSIXlt(t1)
> c(t2$year, t2$mon)

[1] 113   0

> as.numeric(c(format.POSIXct(t1, "%Y"), format.POSIXct(t1, "%m")))

[1] 2013    1
```

시계열 (time-series) 자료형

- R에서 시계열 자료의 처리를 위해 제공하는 패키지와 자료형들은 다음과 같다.
  - `base` 패키지의 `ts` 클래스
  - `zoo` (Zeilies's Ordered Observations) 패키지의 `zoo`, `zooreg` 클래스
  - `xts` 패키지의 `xts` 클래스
  - `tseries` 패키지의 `irts` 클래스
  - `Rmetrics` 패키지의 `timeSeries` 클래스
  - `its` 패키지의 `its` 클래스
- 이러한 자료들은 vector/matrix/data.frame과 달리 자료의 시간을 나타내는 `index` 속성, 주기 (frequency) 등의 추가 속성, 시계열 처리에 유용한 method를 가진다.

ts 클래스

- 년/분기/월 단위의 시계열 표시에 적합
- `ts(data, frequency, start, end)`
  - `data` 시계열 자료가 되는 벡터/매트릭스/데이터프레임
  - `frequency` 시계열 주기 (연간:1, 분기:4, 월간:12)
  - `start` 시작 날짜, 연(year)의 경우에는 길이 1인 정수벡터, 년/월의 경우에는 길이 2인 정수벡터

```
> ts(1:10, start=1997)

Time Series:
Start = 1997
End = 2006
Frequency = 1
 [1]  1  2  3  4  5  6  7  8  9 10

> ts(1:12, frequency=4, start=1997)

     Qtr1 Qtr2 Qtr3 Qtr4
1997    1    2    3    4
1998    5    6    7    8
1999    9   10   11   12

> ts(1:20, frequency=12, start=c(1997, 5))

     Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
1997                   1   2   3   4   5   6   7   8
1998   9  10  11  12  13  14  15  16  17  18  19  20
```

ts 클래스 메소드

- ❏ `start()`, `end()` : 시작일/종료일 속성
- ❏ `frequency()`, `deltat()` : 빈도 및 주기
- ❏ `window(ts, start, end)` : 전체 시계열 중 일부 구간
- ❏ `lag(ts, k)` : 시계열 k 단위만큼 지연
- ❏ `diff(ts, lag)` : lag 단위 시계열 값의 차이

```
> ts <- ts(1:8, frequency=12, start=c(2000, 3))
> ts

     Mar Apr May Jun Jul Aug Sep Oct
2000   1   2   3   4   5   6   7   8

> lag(ts, k=2)

     Jan Feb Mar Apr May Jun Jul Aug
2000   1   2   3   4   5   6   7   8

> diff(ts, lag=1)

     Apr May Jun Jul Aug Sep Oct
2000   1   1   1   1   1   1   1

> c(start(ts), end(ts), frequency(ts), deltat(ts))

[1] 2.000000e+03 3.000000e+00 2.000000e+03 1.000000e+01
[5] 1.200000e+01 8.333333e-02

> window(ts, start=c(2000, 5), end=c(2000, 8))

     May Jun Jul Aug
2000   3   4   5   6
```

ts 클래스 메소드 (계속)

- `cbind(ts1, ts2)` : 두 시계열 합치기
- `ts.union(ts1, ts2)` : `cbind(ts1, ts2)`와 동일
- `ts.intersect(ts1, ts2)` : 데이터가 없는 부분 생략

```
> ts1 <- ts(1:8, frequency=12, start=c(2000, 3))
> ts2 <- ts(80:91, frequency=12, start=c(2000, 1))
> cbind(ts1, ts2)

         ts1 ts2
Jan 2000  NA  80
Feb 2000  NA  81
Mar 2000   1  82
Apr 2000   2  83
May 2000   3  84
Jun 2000   4  85
Jul 2000   5  86
Aug 2000   6  87
Sep 2000   7  88
Oct 2000   8  89
Nov 2000  NA  90
Dec 2000  NA  91

> ts.intersect(ts1, ts2)

         ts1 ts2
Mar 2000   1  82
Apr 2000   2  83
May 2000   3  84
Jun 2000   4  85
Jul 2000   5  86
Aug 2000   6  87
Sep 2000   7  88
Oct 2000   8  89
```

zoo 클래스

- 주간/일간 시계열 자료 표현 가능
- `zoo(x, order.by)`, `as.zoo(x)` 명령어로 생성/변환
  - `x` : 시계열 자료 벡터/매트릭스/데이터프레임
  - `order.by` : 인덱스 벡터. `Date`/`POSIXct` 자료형
  - 인덱스 벡터의 길이는 자료 벡터/매트릭스의 행(row) 수와 같아야 함
- `ts`와 같이 `start`, `end`, `window`, `lag`, `diff` 메소드 가능
- `index()` : 인덱스 추출
- `coredata()` : 자료 추출
- `[]`을 이용한 인덱싱 가능

zoo 사용 예제

```
> library(zoo)
> x <- seq(5.0, length=12, by=0.3)
> t <- as.Date("2013-02-22") + 1:12
> z1 <- zoo(x, order.by=t)
> z1

2013-02-23 2013-02-24 2013-02-25 2013-02-26 2013-02-27 2013-02-28 2013-03-01 2013-03-02 2013-03-03
       5.0        5.3        5.6        5.9        6.2        6.5        6.8        7.1        7.4
2013-03-04 2013-03-05 2013-03-06
       7.7        8.0        8.3

> z2 <- as.zoo(x)
> index(z2) <- t
> z2

2013-02-23 2013-02-24 2013-02-25 2013-02-26 2013-02-27 2013-02-28 2013-03-01 2013-03-02 2013-03-03
       5.0        5.3        5.6        5.9        6.2        6.5        6.8        7.1        7.4
2013-03-04 2013-03-05 2013-03-06
       7.7        8.0        8.3

> index(z1)

 [1] "2013-02-23" "2013-02-24" "2013-02-25" "2013-02-26" "2013-02-27" "2013-02-28" "2013-03-01"
 [8] "2013-03-02" "2013-03-03" "2013-03-04" "2013-03-05" "2013-03-06"

> coredata(z1)

 [1] 5.0 5.3 5.6 5.9 6.2 6.5 6.8 7.1 7.4 7.7 8.0 8.3

> z1[as.Date("2013-02-25") + 0:2]

2013-02-25 2013-02-26 2013-02-27
       5.6        5.9        6.2
```

xts 클래스

- `xts(x, order.by)`, `as.xts(x)` 명령어로 생성/변환
- 날짜/시간 문자열을 이용한 인덱싱
- binary search 를 사용한 빠른 인덱싱
- `first`, `last`, `periodicity`, `endpoints`, `to.period` 등의 다양한 추가 메소드

```
> library(xts)
> data(sample_matrix)
> x1 <- xts(sample_matrix, order.by=as.Date(rownames(sample_matrix)))
> head(x1)

               Open     High      Low    Close
2007-01-02 50.03978 50.11778 49.95041 50.11778
2007-01-03 50.23050 50.42188 50.23050 50.39767
2007-01-04 50.42096 50.42096 50.26414 50.33236
2007-01-05 50.37347 50.37347 50.22103 50.33459
2007-01-06 50.24433 50.24433 50.11121 50.18112
2007-01-07 50.13211 50.21561 49.99185 49.99185

> x2 <- as.xts(sample_matrix, dateFormat='Date')
> head(x2)

               Open     High      Low    Close
2007-01-02 50.03978 50.11778 49.95041 50.11778
2007-01-03 50.23050 50.42188 50.23050 50.39767
2007-01-04 50.42096 50.42096 50.26414 50.33236
2007-01-05 50.37347 50.37347 50.22103 50.33459
2007-01-06 50.24433 50.24433 50.11121 50.18112
2007-01-07 50.13211 50.21561 49.99185 49.99185

> x2['/2007-01-07']

               Open     High      Low    Close
2007-01-02 50.03978 50.11778 49.95041 50.11778
2007-01-03 50.23050 50.42188 50.23050 50.39767
2007-01-04 50.42096 50.42096 50.26414 50.33236
2007-01-05 50.37347 50.37347 50.22103 50.33459
2007-01-06 50.24433 50.24433 50.11121 50.18112
2007-01-07 50.13211 50.21561 49.99185 49.99185
```

금융 데이터 관련 패키지

- TTR
  - 기술적 분석 (Technical indicator)
- quantmod
  - 금융 시계열 자료 임포트 및 차팅
- Quandl
  - 금융 시계열 자료 임포트

TTR 패키지

- Technical Trading Rules
- 50여개 이상의 기술적 분석 기능 및 보조 기능 제공
  - adjRatios, ADX, ATR, BBands, CCI, chaikinAD, chaikinVolatility, adjRatios, CLV, CMF, CMO, DonchianChannel, DPO, DVI, EMV, GMMA, KST, lags, MACD, MFI, OBV, PBands, ROC, rollSFM, RSI, runPercentRank, runSum, SAR, SMA, stoch, TDI, TRIX, VHF, volatility, williamsAD, WPR, ZigZag 등
- 다양한 자료형 사용 가능
  - vector, matrix, dta.frame, ts, zoo, xts, timeSeries, its 등

quantmod 패키지

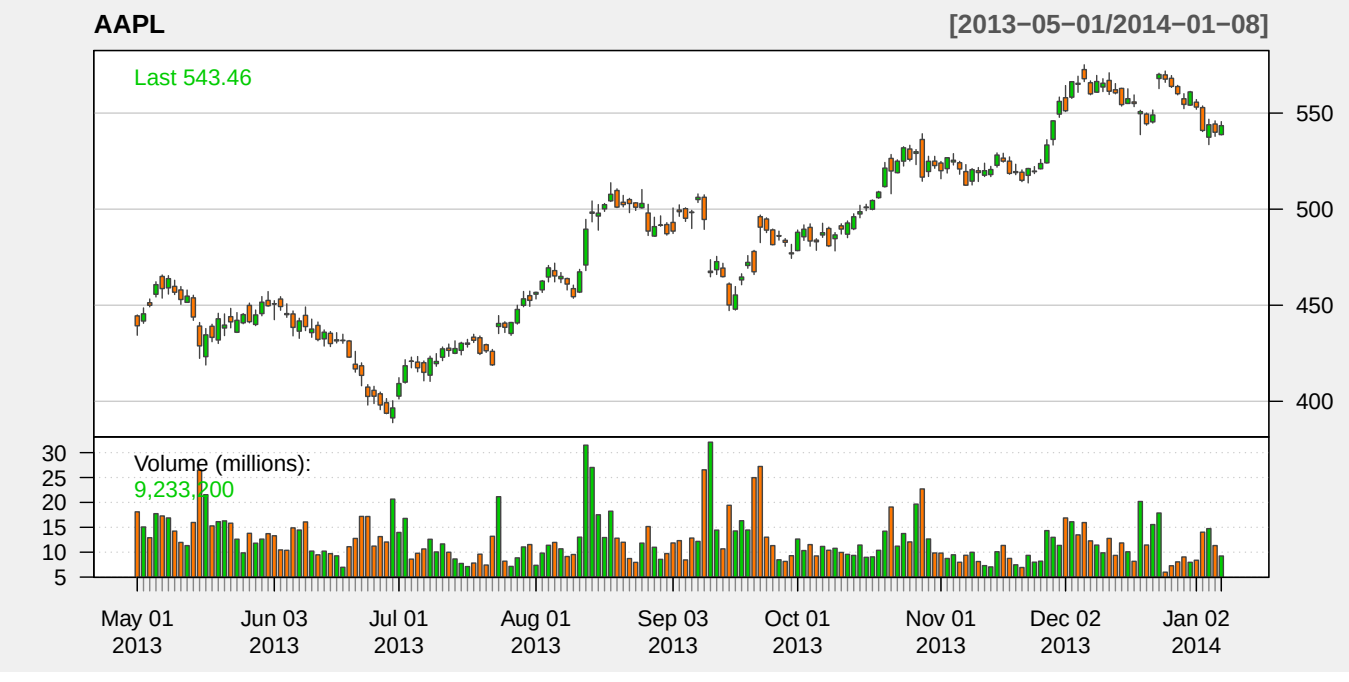
- Quantitative Financial Modelling Framework
- 기능
  - 인터넷/데이터베이스를 이용한 금융 자료 임포트
  - 차트 및 시각화 (visualization)
  - TTR 패키지의 기술적 분석 인디케이터 기능 이용 가능

quantmod 패키지의 데이터 임포트 기능

- `getSymbols(symbol, src, auto.assign=TRUE)`
- `auto.assign=TRUE` 일때는 symbol이름을 가진 변수를 워크스페이스에 생성
- `auto.assign=FALSE` 일때는 자료 리턴
- src에 따라 자료를 임포트하는 source 지정
- 인터넷 사이트에서 임포트하는 경우
  - src=yahoo : 야후 금융 사이트
  - src=google : 구글 금융 사이트
  - src=oanda : oanda 외환 사이트
  - src=FRED : FRED 데이터베이스
- 파일/데이터베이스에서 임포트하는 경우
  - src=RData : RData 파일
  - src=csv : csv 파일
  - src=SQLite : SQLite 데이터베이스 파일
  - src=MySQL : MySQL 데이터베이스

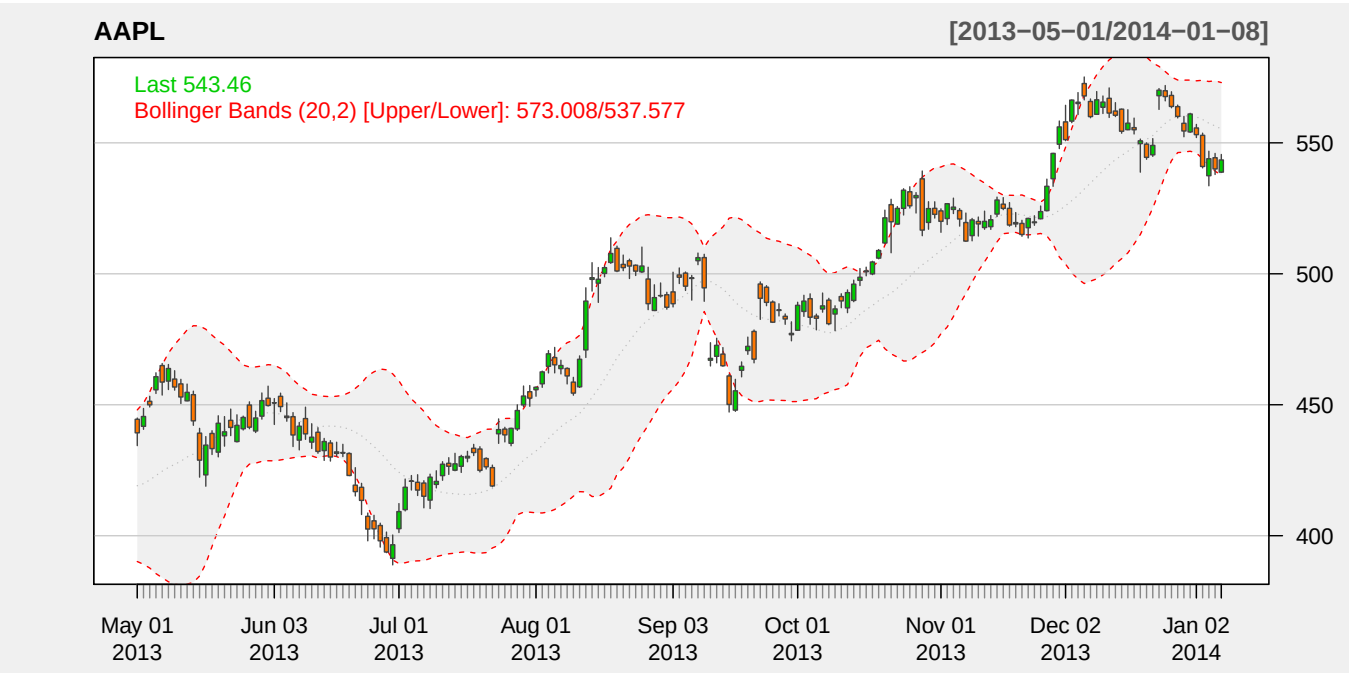
quantmod 패키지의 차트 기능 예 1: 캔들차트

```
> library(quantmod)
> getSymbols("AAPL")
> chartSeries(AAPL, subset='201305/', theme='white')
```



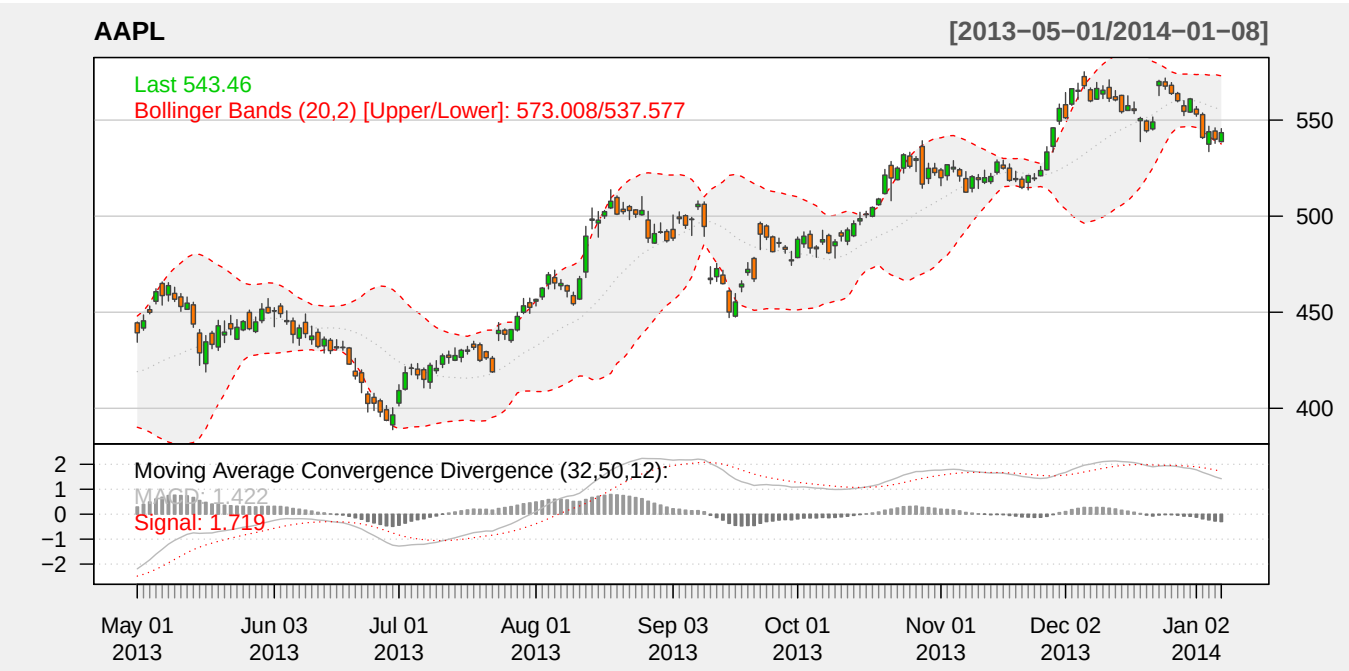
quantmod 패키지의 차트 기능 예 2: 볼랜드 밴드

```
> library(quantmod)
> getSymbols("AAPL")
> chartSeries(AAPL, subset='201305/', theme='white', TA=c(addBBands()))
```



quantmod 패키지의 차트 기능 예 3: MACD

```
> library(quantmod)
> getSymbols("AAPL")
> chartSeries(AAPL, subset='201305/', theme='white', TA=c(addBBands(),addMACD(32,50,12)))
```



Quandl 패키지

- Quandl : 7백만개 이상의 금융/경제/사회 지표 및 수치 자료를 검색할 수 있는 웹사이트
- Quandl 웹사이트 주소 : `http://www.quandl.com/`
- Quandl 웹사이트의 자료를 다운로드 가능
  - `Quandl.search()` 자료 검색
  - `Quandl()` 자료 다운로드

```
> library(Quandl)
> Quandl.search("Oil", source = "NSE")

Oil India Limited
Code: NSE/OIL
Desc: Historical prices for Oil India Limited (OIL), (ISIN: INE274J01014),  National Stock Exchange of India.
Freq: daily
Cols: Date|Open|High|Low|Last|Close|Total Trade Quantity|Turnover (Lacs)

K S Oils Limited
Code: NSE/KSOILS
Desc: Historical prices for K S Oils Limited (KSOILS), (ISIN: INE727D01022),  National Stock Exchange of India.
Freq: daily
Cols: Date|Open|High|Low|Last|Close|Total Trade Quantity|Turnover (Lacs)

Savita Oil Technologies Limited
Code: NSE/SOTL
Desc: Historical prices for Savita Oil Technologies Limited (SOTL), (ISIN: INE035D01012),  National Stock Exchange of India.
Freq: daily
Cols: Date|Open|High|Low|Last|Close|Total Trade Quantity|Turnover (Lacs)

> ts = Quandl("NSE/OIL",type = "ts")
> head(ts)

[1] 1096.0 1102.0 1152.0 1149.8 1153.8 1145.0
```