Lecture 1: Overview of R-Package

Brian Thelen
443 West Hall
bjthelen@umich.edu

Statistics 509 - Winter 2016

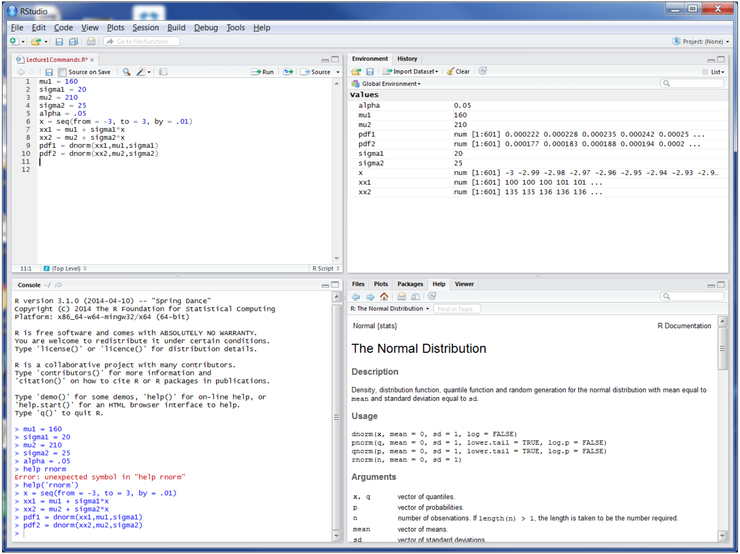
Overview.

- R is a free software environment for statistical computing and graphics
- Runs on UNIX platforms, Windows and Mac
- Website is http://www.r-project.org/
  - Downloadable precompiled binary distribution (for all of the platforms) available on Comprehensive R Archive Network (CRAN) sites
  - Example CRAN site is http://www.biometrics.mtu.edu/CRAN/
- History
  - Freeware version of S (S-Plus) developed at Bell Labs
- Many add-on packages avaiable via CRAN

R References and Resources on CTools

- **R-refcard** – quick reference card for many R-commands
  - Quick high-level list/description
- **R-intro** – brief introduction to structure of R and how to do most basic things
- **R-debuts** – another brief introduction to structure of R and how to do most basic things
- RStudio Integrated Development Environment (IDE) is a powerful and productive user interface for R.
  - Free and open source, and works great on all platforms

GUI for RStudio



Overview of R and Basic Commands

- R is an object-oriented language
  - objects are variables, data, functions, results, etc,.
  - stored in the active memory of the computer in the form of objects which have a name
- Simple data definition and arithmetic operations

```
> n <-10
> x <-c(1,3,5)
> n
[1] 10
> x
[1] 1 3 5
> xx <-c(1,3,5)+1
> xx
[1] 2 4 6
> ls()
[1] "n" "x" "xx"
> rm(n,x)
> ls()
[1] "xx"
> y <- 2*xx
> y
[1]  4  8 12
```

Getting help in R

Remark. To get help simply do help() with command in parentheses

```
> help(ls)
```

and this generates window with

- description of ls
- example commands using ls
- similar/related commands using ls

As an example, the command below generates all objects with an "x" in the object name.

```
> ls(pattern="x")
[1] "x"  "xx"
```

Screen Shot - help(ls)

ls(base) R Documentation

List Objects

Description

`ls` and `objects` return a vector of character strings giving the names of the objects in the specified environment. When invoked with no argument at the top level prompt, `ls` shows what data sets and functions a user has defined. When invoked with no argument inside a function, `ls` returns the names of the functions local variables. This is useful in conjunction with `browser`.

Usage

```
ls(name, pos = -1, envir = as.environment(pos),
   all.names = FALSE, pattern)
objects(name, pos= -1, envir = as.environment(pos),
        all.names = FALSE, pattern)
```

Arguments

`name`	which environment to use in listing the available objects. Defaults to the *current* environment. Although called `name` for back compatibility, in fact this argument can specify the environment in any form; see the details section.
`pos`	an alternative argument to `name` for specifying the environment as a position in the search list. Mostly there for back compatibility.
`envir`	an alternative argument to `name` for specifying the environment evaluation environment. Mostly there for back compatibility.
`all.names`	a logical value. If `TRUE`, all object names are returned. If `FALSE`, names which begin with a `.` are omitted.
`pattern`	an optional regular expression. Only names matching `pattern` are returned. `glob2rx` can be used to convert wildcard patterns to regular expressions.

Details

The `name` argument can specify the environment from which object names are taken in one of several forms: as an integer (the position in the `search` list); as the character string name of an element in the search list; or as an explicit `environment` (including using `sys.frame` to access the currently active function calls). By default, the environment of the call to `ls` or `objects` is used. The `pos` and `envir` arguments are an alternative way to specify an environment, but are primarily there for back compatibility.

Note that the *order* of the resulting strings is locale dependent, see `Sys.getlocale`.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

See Also

`glob2rx` for converting wildcard patterns to regular expressions.

`ls.str` for a long listing based on `str`. `apropos` (or `find`) for finding objects in the whole search path; `grep` for more details on 'regular expressions'; `class`, `methods`, etc., for object-oriented programming.

Examples

```
.Ob <- 1
ls(pattern = "O")
ls(pattern= " O", all.names = TRUE)    # also shows ".[foo]"
```

Work Directory and Loading in Data

- **Workspace** (all objects) can be saved for future sessions
  - Default name is .RData – load (under file button) at future sessions
- **Command history** can be saved for future sessions
  - Default name is .Rhistory - load (under file button) at future sessions
  - Use up-arrow to go to previous command
- **Work directory** established via
  - Establish your work directory via the Change dir command under "File"
- **Options for inputting commands**
  - Via command line
  - Input set of commands in a file via "source"
    - Typically file is of the form "input_file_name.R"
- On start-up, the file .RProfile is sourced
  - Put here anything want done every time on startup – examples are changing directory, loading workspace, installing packages

Input data file for analysis within R

- **SP500 data for 1960-2014**
- From http://finance.yahoo.com/
  - Go to Investing (on bottom lefthand side) and then to Today Markets indices and then pick chart
  - Pick button corresponding to Historical Prices and it allows you to get what you want
    - typical output format is comma-separated-values – .csv

SP500_wkly_Jan1_60_Dec28_14

	A	B	C	D	E	F	G
1	Date	Open	High	Low	Close	Volume	Adj Close
2	12/22/2014	2069.28	2092.7	2069.28	2088.77	2.39E+09	2088.77
3	12/15/2014	2005.03	2077.85	1972.56	2070.65	5.08E+09	2070.65
4	12/8/2014	2074.84	2075.78	2002.33	2002.33	3.99E+09	2002.33
5	12/1/2014	2065.78	2079.47	2049.57	2075.37	3.66E+09	2075.37
6	11/24/2014	2065.07	2075.76	2064.75	2067.56	2.94E+09	2067.56
7	11/17/2014	2038.29	2071.46	2034.46	2063.5	3.4E+09	2063.5
8	11/10/2014	2032.01	2046.18	2030.17	2039.82	3.23E+09	2039.82
9	11/3/2014	2018.21	2034.26	2001.01	2031.92	3.73E+09	2031.92
10	10/27/2014	1962.97	2018.19	1951.37	2018.05	3.76E+09	2018.05
11	10/20/2014	1885.62	1965.27	1882.3	1964.58	3.59E+09	1964.58
12	10/13/2014	1905.65	1912.09	1820.66	1886.76	4.96E+09	1886.76
13	10/6/2014	1970.01	1977.84	1906.05	1906.13	4.07E+09	1906.13
14	9/29/2014	1978.96	1985.17	1926.03	1967.9	3.76E+09	1967.9
15	9/22/2014	2009.08	2009.08	1965.99	1982.85	3.23E+09	1982.85
16	9/15/2014	1986.04	2019.26	1978.48	2010.4	3.45E+09	2010.4
17	9/8/2014	2007.17	2007.17	1980.26	1985.54	2.95E+09	1985.54
18	9/2/2014	2004.07	2011.17	1990.1	2007.71	2.88E+09	2007.71
19	8/25/2014	1991.74	2005.04	1990.52	2003.37	2.31E+09	2003.37
20	8/18/2014	1958.36	1994.76	1958.36	1988.4	2.56E+09	1988.4
21	8/11/2014	1933.43	1964.04	1928.29	1955.06	2.75E+09	1955.06
22	8/4/2014	1926.62	1942.92	1904.78	1931.59	3.24E+09	1931.59

SP500_wkly_Jan1_60_Dec28_14

Data reading-Analysis-Plotting

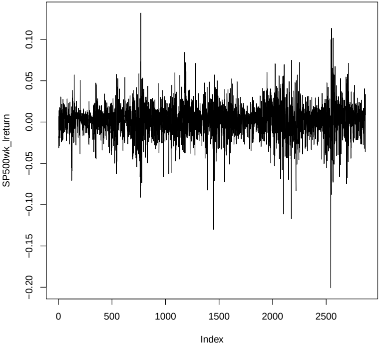
- Using source file of commands – SP500_analysis.R

```
> source("SP500_analysis.R")
```

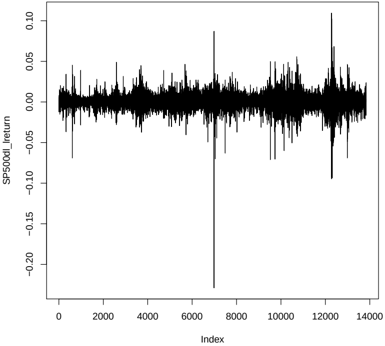
- Below is the file SP500_analysis.R

```
X = read.csv("Data\\SP500_wkly_Jan1_60_Dec28_14.csv",header=TRUE)
SP500wk <- rev(X$Close)
plot(SP500wk)

SP500wk_lreturn <- diff(log(SP500wk)) # generating log returns (weekly)
windows() # opening a new plotting window
plot(SP500wk_lreturn,type='l')

XX = read.csv("Data\\SP500_daily_Jan1_60_Dec28_14.csv",header=TRUE)
SP500dl <- rev(XX$Close)
windows() # opening a new plotting window
plot(SP500dl,type='l')
SP500dl_lreturn <- diff(log(SP500dl)) # generating difference in log(daily clos
windows() # opening a new plotting window
plot(SP500dl_lreturn)
```

Generated Plots



Plot of Weekly Log-Returns



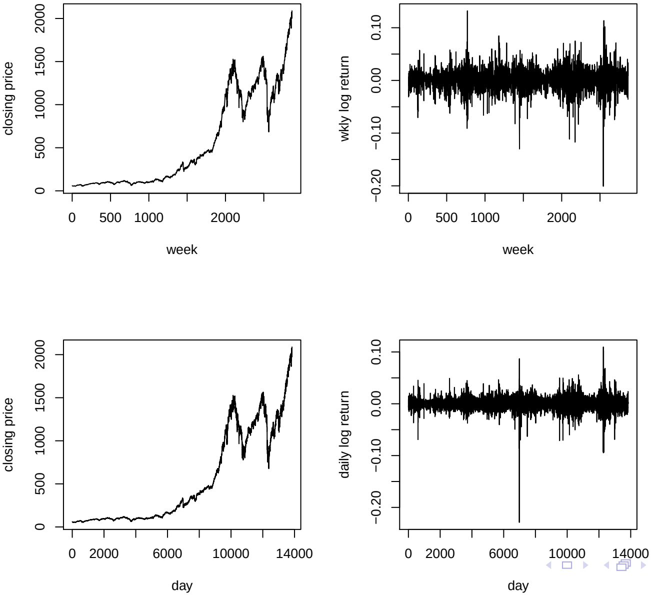
Plot of Daily Log-Returns

- Note the 20% loss on October 19 1987 - Black Monday

```
windows() # opening a new plotting window
par(mfrow=c(2,2)) # setting up for a 2 x 2 arrangement of subplots
plot(SP500wk,xlab='week',ylab='closing price',type='l')
plot(SP500wk_lreturn,xlab='week',ylab='wkly log return',type='l')
plot(SP500dl,xlab='day',ylab='closing price',type='l')
plot(SP500dl_lreturn,xlab='day',ylab='daily log return',type='l')
```

- First generate 4 separate plots in 4 separate windows
- Lastly generates the 4 plots in one window

Generated Plots



More SP500 Descriptive Statistics

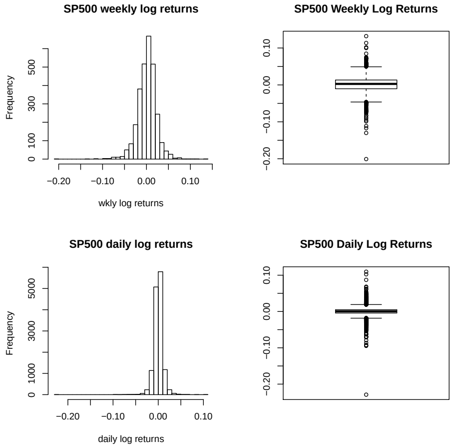
- Below are examples of quantitative descriptive statistics

```
> dim(XX)
[1] 13841     7
> summary(XX)
       Date              Open              High              Low              Close
 1/10/1961:     1   Min.    :  52.20   Min.    :  52.20   Min.    :  51.35   Min.    :  52.20
 1/10/1962:     1   1st Qu.:  98.16   1st Qu.:  99.01   1st Qu.:  97.48   1st Qu.:  98.16
 1/10/1963:     1   Median : 261.05   Median : 262.61   Median : 259.33   Median : 261.13
 1/10/1964:     1   Mean    : 536.50   Mean    : 539.98   Mean    : 532.87   Mean    : 536.65
 1/10/1966:     1   3rd Qu.:1073.43   3rd Qu.:1082.62   3rd Qu.:1065.22   3rd Qu.:1073.48
 1/10/1967:     1   Max.    :2084.30   Max.    :2092.70   Max.    :2084.30   Max.    :2088.77
 (Other)   :13835
     Volume             Adj.Close
 Min.    :1.890e+06   Min.    :  52.20
 1st Qu.:1.612e+07   1st Qu.:  98.16
 Median :1.427e+08   Median : 261.13
 Mean    :8.890e+08   Mean    : 536.65
 3rd Qu.:1.111e+09   3rd Qu.:1073.48
 Max.    :1.146e+10   Max.    :2088.77
```

```
> summary(SP500wk)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  52.68   98.34  259.70  536.60 1073.00 2089.00
> summary(SP500dl)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  52.20   98.16  261.10  536.70 1073.00 2089.00
> summary(SP500wk_lreturn)
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-0.200800 -0.010490  0.002732  0.001241  0.013350  0.132000
> summary(SP500dl_lreturn)
      Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
-0.2290000 -0.0043280  0.0004288  0.0002566  0.0050620  0.1096000
> sd(SP500wk_lreturn) # computing standard deviation
[1] 0.02158714
> sd(SP500dl_lreturn) # computing standard deviation
[1] 0.01010955
```

Histograms and Boxplots

```
> par(mfrow=c(2,2)) # setting up for a 2 x 2 arrangement of subplots
> hist(SP500wk_lreturn,xlab='wkly log returns',breaks=25,main='SP500 weekly log returns')
# histogram with 25 bins
> boxplot(SP500wk_lreturn)
> title('SP500 Weekly Log Returns')
> hist(SP500dl_lreturn,xlab='daily log returns',breaks=25,main='SP500 daily log returns')
# histogram with 25 bins
> boxplot(SP500dl_lreturn)
> title('SP500 Daily Log Returns')
```



Objects/Mode in R

- In previous example, X and XX are example of data frames
  - To refer to variables, utilized X$Close X$Date X$Volume
  - Can subset/select
    - Similar, but a little more difficult than Matlab

object	modes	several modes possible in the same object?
vector	numeric, character, complex *or* logical	No
factor	numeric *or* character	No
array	numeric, character, complex *or* logical	No
matrix	numeric, character, complex *or* logical	No
data.frame	numeric, character, complex *or* logical	Yes
ts	numeric, character, complex *or* logical	Yes
list	numeric, character, complex, logical, function, expression, ...	Yes

```
> x = c(0,1)
> xf = as.factor(x)
> xf
[1] 0 1  Levels: 0 1
> is.factor(xf)
[1] TRUE
> xn = as.numeric(xf)
> xn
[1] 1 2
```

More Commands - Subsetting rows/columns

```
> X_nodate <- X[,c(2:7)]  # deletes date column
> X_nodate[c(1:5),]
     Open    High     Low   Close     Volume Adj.Close
1 1534.06 1547.23 1462.04 1473.17 3563670000   1473.17
2 1552.50 1555.90 1529.20 1534.10 3263540000   1534.10
3 1530.43 1555.10 1506.10 1552.50 3066650000   1552.50
4 1504.66 1532.40 1504.66 1530.44 2317562500   1530.44
5 1502.56 1517.53 1484.18 1503.35 3251210000   1503.35
>
>
> SP500wk_lreturn_pos <- SP500wk_lreturn[SP500wk_lreturn>0]
> SP500wk_lreturn_pos[c(1:4)]
[1] 0.006631443 0.013966207 0.014096343 0.017479497
```

Vectors and Matrices

- Vectors – example commmand below for creating a vector

```
x <- c(1:4) # makes a vector with elements of 1 2 3 4
```

- Matrices – example command below for creating matrix
  - First example names the rows/columns

```
mdat <- matrix(c(1,2,3, 11,12,13), nrow = 2, ncol=3, byrow=TRUE,
               dimnames = list(c("row1", "row2"), c("C.1", "C.2", "C.3")))
> mdat
     C.1 C.2 C.3
row1   1   2   3
row2  11  12  13
> > x <- cbind(c(1:4),c(5:8))
> x
     [,1] [,2]
[1,]    1    5
[2,]    2    6
[3,]    3    7
[4,]    4    8
> x <- rbind(c(1:4),c(5:8))
> x
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
```

Packages in R

- In R, can easily install and load packages
  - New set of commands/capabilities
  - To install package, easy by Tools pulldown menu in RStudio
    - Need to do this only once
  - To include a package, utilize "library" or "require" command
    - Need to do this every time
    - Example commands

```
> library(forecast)
> library(fExtremes)
> library(fGarch)
```

- We will use these packages in this course (and more)
  - All of these packages have documentation on CRAN and in the on-line after installing – varying quality
  - The packages fExtremes and fGarch are part of the Rmetrics project
    - http://www.rmetrics.org/