

Introduction to R

Time Series Analysis

AMS 316

What is R?

Programming language and software environment for data manipulation, calculation and graphical display.

Originally created by Ross Ihaka and Robert Gentleman at University of Auckland, and now developed by the R Development Core Team.

Why use R?

- ✓ *IT IS FREE*

- ✓ Pre-compiled binary versions are provided for Microsoft Windows, Mac OS X, and several other Linux/Unix-like operating systems

- ✓ Open source code available freely available on GNU General Public License

- ✓ For computationally-intensive tasks, C, C++ and Fortran code can be linked and called at run time

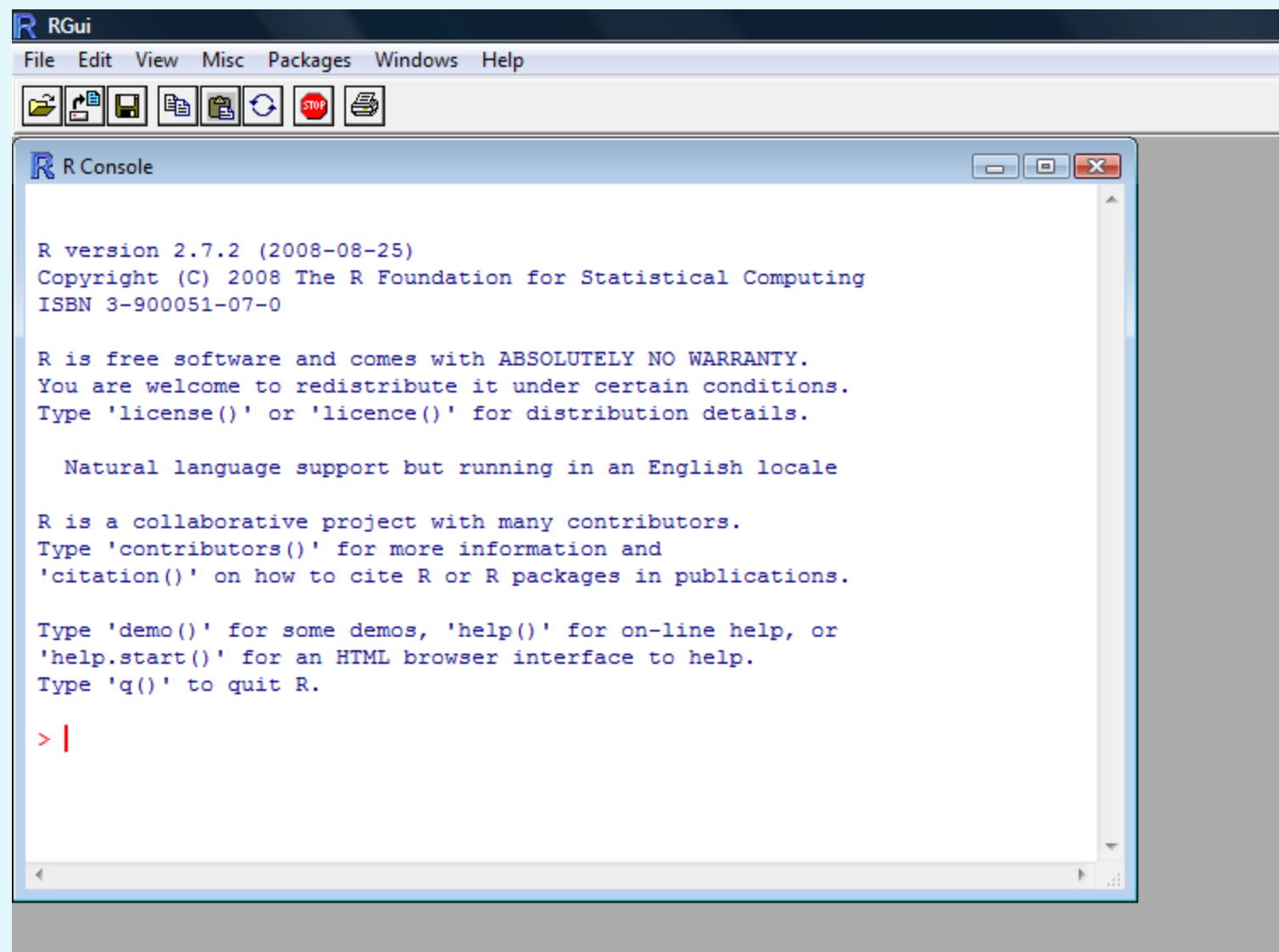
- ✓ An effective data handling and storage facility

- ✓ A suite of operators for calculations on arrays, in particular matrices

- ✓ A large, coherent, integrated collection of intermediate tools for data analysis

- ✓ Graphical facilities for data analysis and display either directly at the computer or on hardcopy

R Operating Environment



R Operating Environment

R RGui

File Edit Packages Windows Help

R Console

```
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> x <- rnorm(50)
> y <- rnorm(x)
> plot(x, y)
> ls()
[1] "x" "y"
> rm(x, y)
> x <- 1:20
> w <- 1 + sqrt(x)/2
> dummy <- data.frame(x=x, y= x + rnorm(x)*w)
> dummy
   x     y
1 1 0.1092139
2 2 4.9343512
3 3 2.1227078
4 4 5.8692791
5 5 8.2562772
6 6 5.3174608
7 7 9.6317077
8 8 7.4829529
9 9 7.1423278
10 10 12.5036819
```

Untitled - R Editor

```
x <- rnorm(50)
y <- rnorm(x)
plot(x, y)
ls()
rm(x, y)
x <- 1:20
w <- 1 + sqrt(x)/2
dummy <- data.frame(x=x, y= x + rnorm(x)*w)
dummy
```

Run line or selection Ctrl+R

Undo Ctrl+Z

Cut Ctrl+X

Copy Ctrl+C

Paste Ctrl+V

Delete

Select all Ctrl+A

R Graphics: Device 2 (ACTIVE)

A scatter plot titled 'R Graphics: Device 2 (ACTIVE)'. The x-axis is labeled 'x' and ranges from approximately -1.5 to 2.5. The y-axis is labeled 'y' and ranges from -2 to 2. The plot shows a collection of open circle data points forming a clear positive linear trend. The points are scattered around the line $y = x$, with some vertical spread due to the added noise.

Start with R

>help.start()

Search Engine



Search

You can search for keywords, function and data names and text in help page titles.

Usage: Enter a string in the text field below and hit RETURN.

Help page titles Keywords Object names

For search to work, you need Java installed and both Java and JavaScript enabled in your browser.

On the Mozilla/Netscape family of browsers you should see 'Applet SearchEngine started' at the left edge of the status bar. For help on the [R Installation and Administration](#) manual.

On some Mozilla-based browsers the links on the results page will become inactive if you return to it: to work around this you can open a link in a new tab or window.

Even if this search does not work on your system, you can always use `help.search` at the R prompt.

Keywords

Keywords by Topic

Basics

- [attribute](#): Data Attributes
- [chron](#): Dates and Times

Getting Help with Functions

> help(solve) // get more information on “solve”

> ?solve

> help("[[") // help for special characters and “if”, “for”...

> help.start() // launch a Web browser



R commands

1. R is case sensitive.
2. Commands can be executed by calling an external file.
 > source("commands.R")
3. The following functions can be used to display the names of (most of) the objects which are currently stored within R.
 > objects()
 > ls()
4. Objects can be removed by the following function.
 > rm(x, y, z)

Simple manipulations

1. Vectors and assignments

```
> x<- c(10.4, 5.6, 4.1)  
> assign("x", c(10.4, 5.6, 4.1))  
> y <- c(x, 0, 1/x)
```

2. Vector arithmetic

```
> v<-2*x + y +1  
> sum( (x-mean(x))^2/length(x)-1)  
> sort(x) // returns a vector of the same size as x  
           // with the elements arranged in increasing order.
```

```
> max(x)  
> min(x)
```

3. Sequence generation

```
> z <- seq(-5, 5, by=0.2)  
> z <- rep(x, times=5)
```

Arrays & Matrices

```
> dim(z) <- c(3,5,100)

> x <- array(1:20, dim=c(4,5)) # Generate a 4 by 5 array.
> x
 [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    9   13   17
[2,]    2    6   10   14   18
[3,]    3    7   11   15   19
[4,]    4    8   12   16   20
> i <- array(c(1:3,3:1), dim=c(3,2))
> i
 [,1] [,2]
[1,]    1    3
[2,]    2    2
[3,]    3    1
> x[i] # Extract those elements
[1] 9 6 3
> x[i] <- 0 # Replace those elements by zeros.
> x
 [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    0   13   17
[2,]    2    0   10   14   18
[3,]    0    7   11   15   19
[4,]    4    8   12   16   20
>
```

Reading data from files

File types that can be imported into R:

.data, .txt, .xls, .xlsx, .html, .xml, etc.

Example of importing text files into R:

```
data<-read.table("C:/...../data.txt", header=TRUE, sep="\t")
```

Other data import commands: scan()

For data import/export:

<http://cran.r-project.org/doc/manuals/R-data.html>

Reading data from files

Input file form with names and row labels:

	Price	Floor	Area	Rooms	Age	Cent.heat
01	52.00	111.0	830	5	6.2	no
02	54.75	128.0	710	5	7.5	no
03	57.50	101.0	1000	5	4.2	no
04	57.50	131.0	690	6	8.8	no
05	59.75	93.0	900	5	1.9	yes
...						

```
> HousePrice <- read.table("houses.data")
```

Input file form without row labels:

	Price	Floor	Area	Rooms	Age	Cent.heat
	52.00	111.0	830	5	6.2	no
	54.75	128.0	710	5	7.5	no
	57.50	101.0	1000	5	4.2	no
	57.50	131.0	690	6	8.8	no
	59.75	93.0	900	5	1.9	yes
...						

```
> HousePrice <- read.table("houses.data", header=TRUE)  
> inp <- scan("input.dat", list("",0,0))
```

Probability Distributions

Distribution	R name	additional arguments
beta	beta	shape1, shape2, ncp
binomial	binom	size, prob
Cauchy	cauchy	location, scale
chi-squared	chisq	df, ncp
exponential	exp	rate
F	f	df1, df2, ncp
gamma	gamma	shape, scale
geometric	geom	prob
hypergeometric	hyper	m, n, k
log-normal	lnorm	meanlog, sdlog
logistic	logis	location, scale
negative binomial	nbinom	size, prob
normal	norm	mean, sd
Poisson	pois	lambda
Student's t	t	df, ncp
uniform	unif	min, max
Weibull	weibull	shape, scale
Wilcoxon	wilcox	m, n

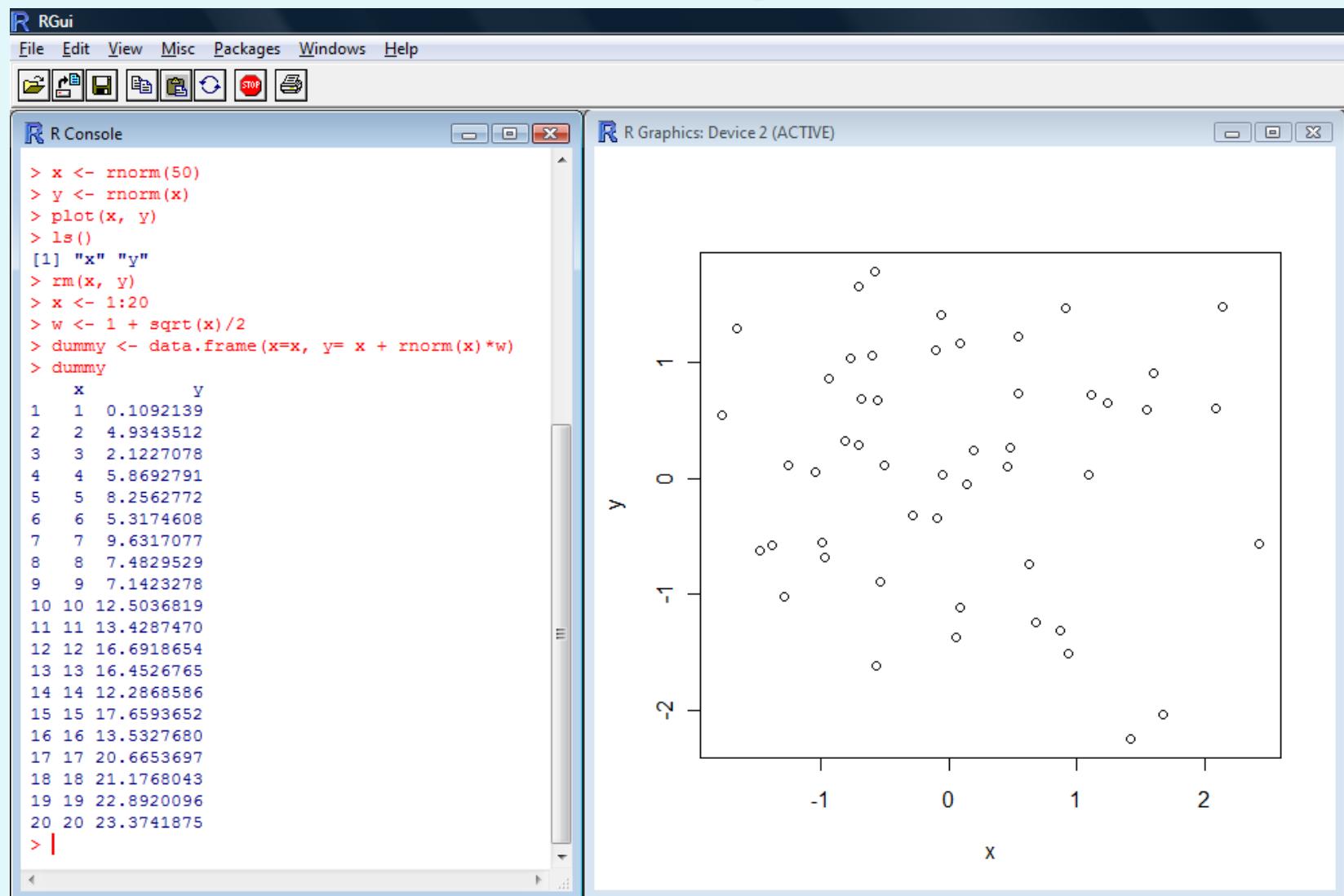
Control Statements

```
> if (expr_1) expr_2 else expr_3  
> for (name in expr_1) expr_2  
> while (condition) expr
```

Writing your own functions

```
> name <- function(arg_1, arg_2, ...) expression
```

Example



Example

```
> fm <- lm(y ~ x, data=dummy)
> summary(fm)

Call:
lm(formula = y ~ x, data = dummy)

Residuals:
    Min      1Q  Median      3Q     Max 
-4.6955 -1.6053  0.7523  1.3564  2.9376 

Coefficients:
            Estimate Std. Error t value
(Intercept) 0.33230   1.03056   0.322
x           1.11850   0.08603  13.001
Pr(>|t|)    
(Intercept) 0.751
x           1.37e-10 ***

---
Signif. codes:  0 '****' 0.001 '***' 0.01 '**' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.218 on 18 degrees of freedom
Multiple R-squared: 0.9038,    Adjusted R-squared: 0.8984 
F-statistic: 169 on 1 and 18 DF,  p-value: 1.374e-10

> fm1 <- lm(y ~ x, data=dummy, weight=1/w^2)
> summary(fm1)

Call:
lm(formula = y ~ x, data = dummy, weights = 1/w^2)

Residuals:
    Min      1Q  Median      3Q     Max 
-1.5609 -0.7937  0.2666  0.5378  1.3798 

Coefficients:
            Estimate Std. Error t value
(Intercept) 0.34517   0.81159   0.425
x           1.11690   0.08276  13.496
Pr(>|t|)    
(Intercept) 0.676
x           7.44e-11 ***

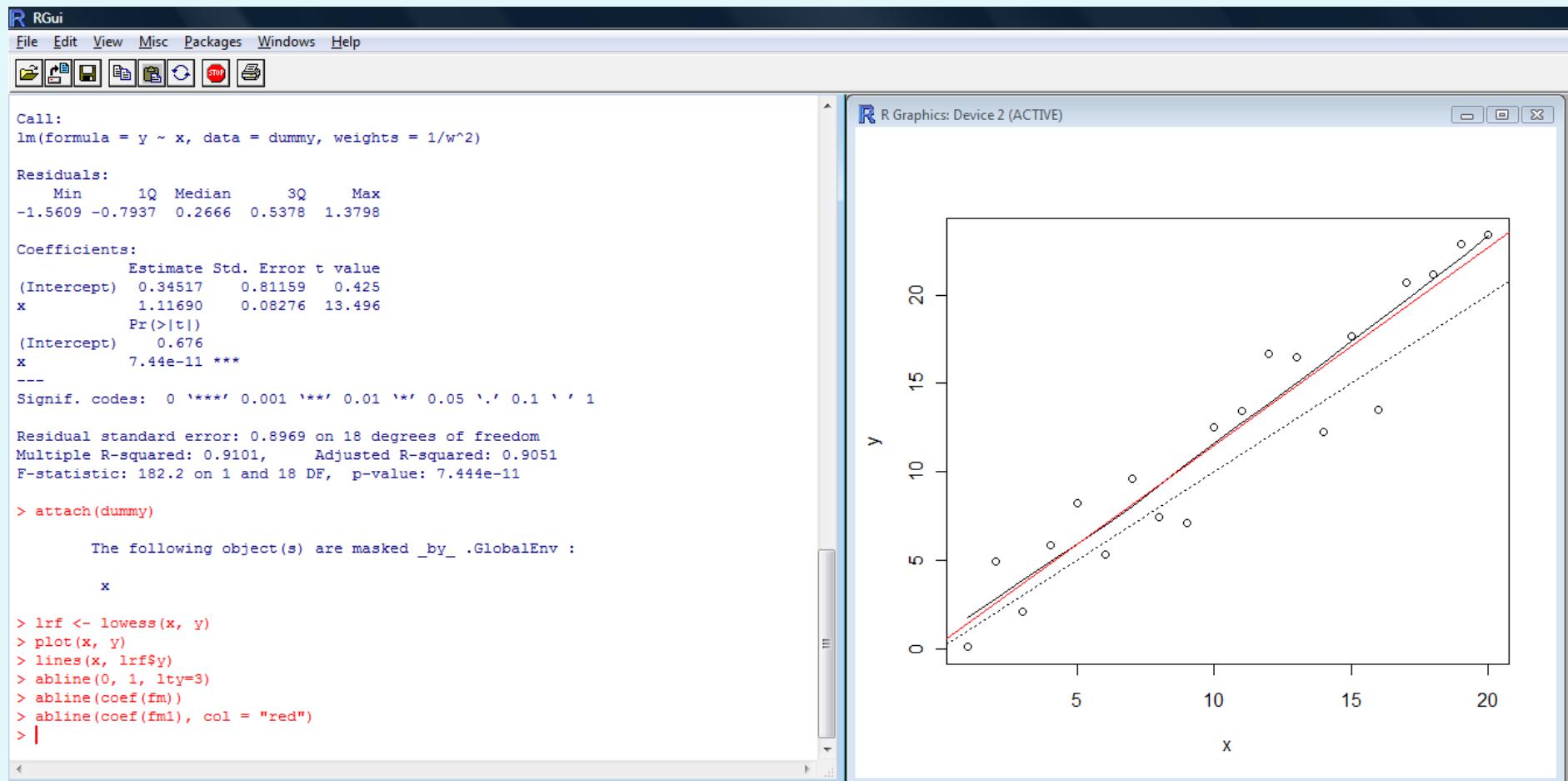
---
Signif. codes:  0 '****' 0.001 '***' 0.01 '**' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.8969 on 18 degrees of freedom
Multiple R-squared: 0.9101,    Adjusted R-squared: 0.9051 
F-statistic: 182.2 on 1 and 18 DF,  p-value: 7.444e-11
```

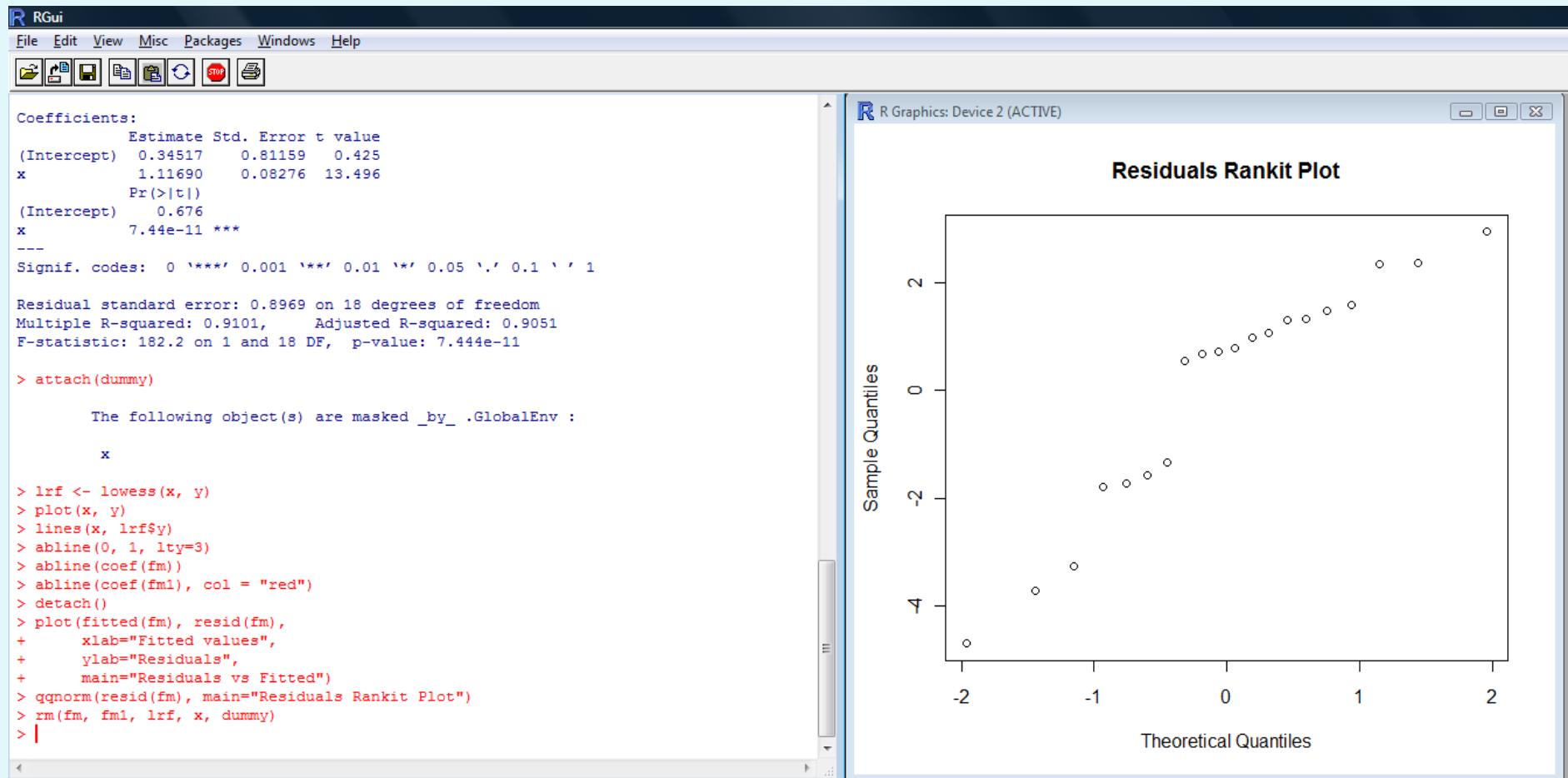
Example

```
attach(dummy)                                //Make the columns in the data frame visible as variables//
lrf <- lowess(x, y)                          //Make a nonparametric local regression function//
plot(x, y)                                    //Standard point plot//
lines(x, lrf$y)                             //Add in the local regression//
abline(0, 1, lty=3)                           //The true regression line: (intercept 0, slope 1)//
abline(coef(fm))                            //Unweighted regression line//
abline(coef(fm1), col = "red")               //Weighted regression line//
detach()                                     //Remove data frame from the search path//
plot(fitted(fm), resid(fm),
      xlab="Fitted values", ylab="Residuals",
      main="Residuals vs Fitted")              /*A standard regression diagnostic plot to check for
qqnorm(resid(fm), main="Residuals Rankit Plot")/*A normal scores plot to check for skewness, kurtosis and
rm(fm, fm1, lrf, x, dummy)                  //Clean up again//
```

Example



Example



Graphics Plot in R

Plot Types: Line Charts, Bar Charts, Histograms, Pie Charts, Dot Charts, etc.

Format:

>PLOT-TYPE(PLOT-DATA, DETAILS)

PLOT-TYPE: plot, plot.xy, barplot, pie, dotchart, etc.

PLOT-DATA: Data, Data\$XXX, as.matrix(Data), etc.

Details: axes, col, pch, lty, ylim, type, xlab, ylab, etc.

For graphics plot:

<http://www.harding.edu/fmccown/R/>

Example

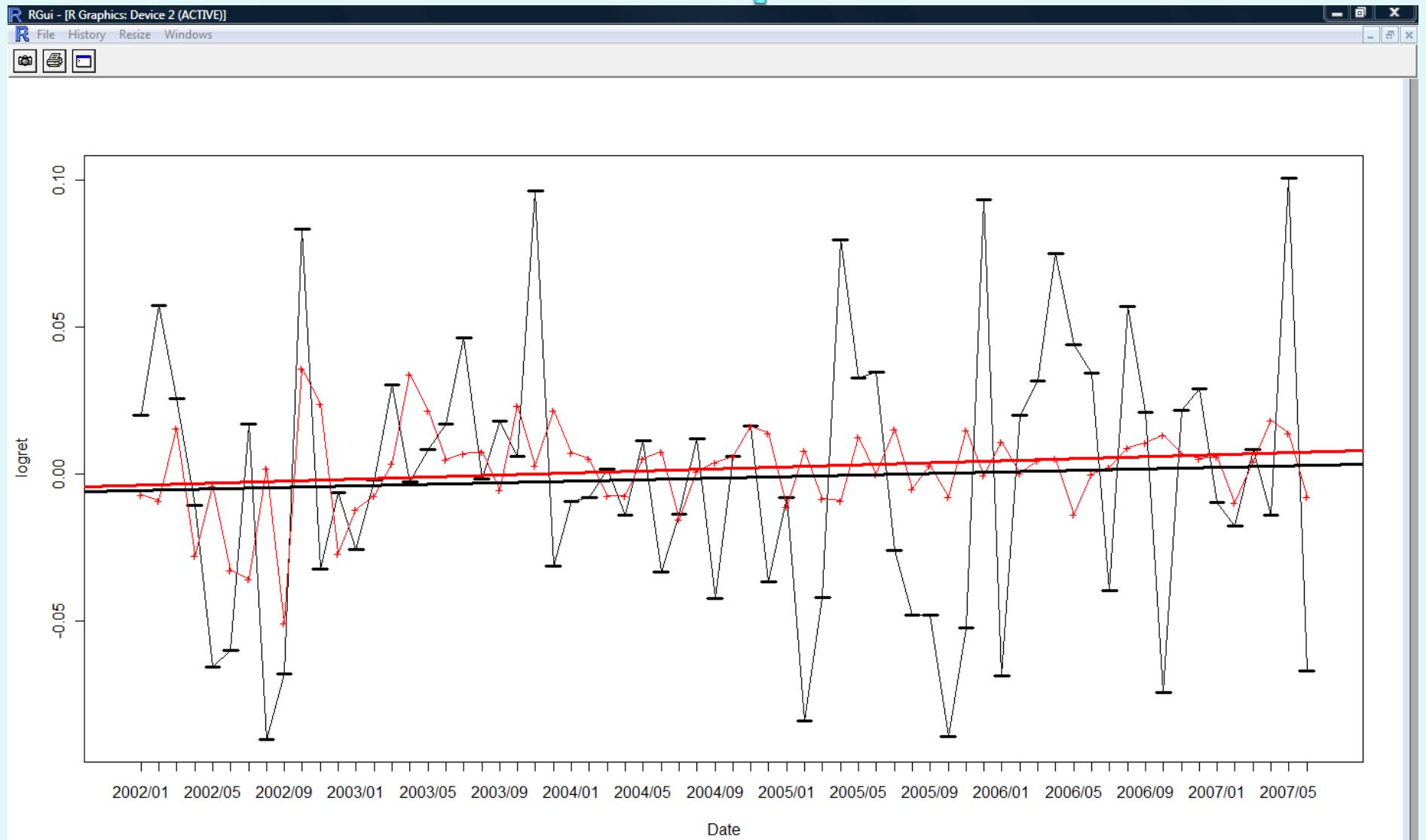
A comparison of GM monthly returns & SP500 monthly returns. GM and SP500 monthly return data during the period of Jan. 2002 to Jun. 2007 are taken. Plotted in R, they will be analyzed and compared.

Data from: [http://www.stanford.edu/~xing/
statfinbook/data.html](http://www.stanford.edu/~xing/statfinbook/data.html)

Example

```
GM<-read.table("C:/R Data/GM.txt", header=TRUE, sep="")
SP<-read.table("C:/R Data/SP.txt", header=TRUE, sep="")
plot(GM)
lines(GM$logret, lty=1)
lines(SP$logret, type="o", lty=1, pch="+", col="red")
x<-1:66
GML<-lm(GM$logret~x)
SPL<-lm(SP$logret~x)
abline(coef(GML), type="h", lwd=3)
abline(coef(SPL), col="red", type="h", lwd=3)
```

Example



Packages in R

Introduction to packages

All R functions and datasets are stored in packages. Only when a package is loaded are its contents available. This is down both for efficiency, and to aid package developers.

To see which packages are installed at your site, issue the command

```
>library(boot)
```

Users connected to the Internet can use `install.packages()` and `update.packages()` to install and update packages.

To see packages currently loaded, use `search()`.

Example

An easier way: use TS functions to plot time series.

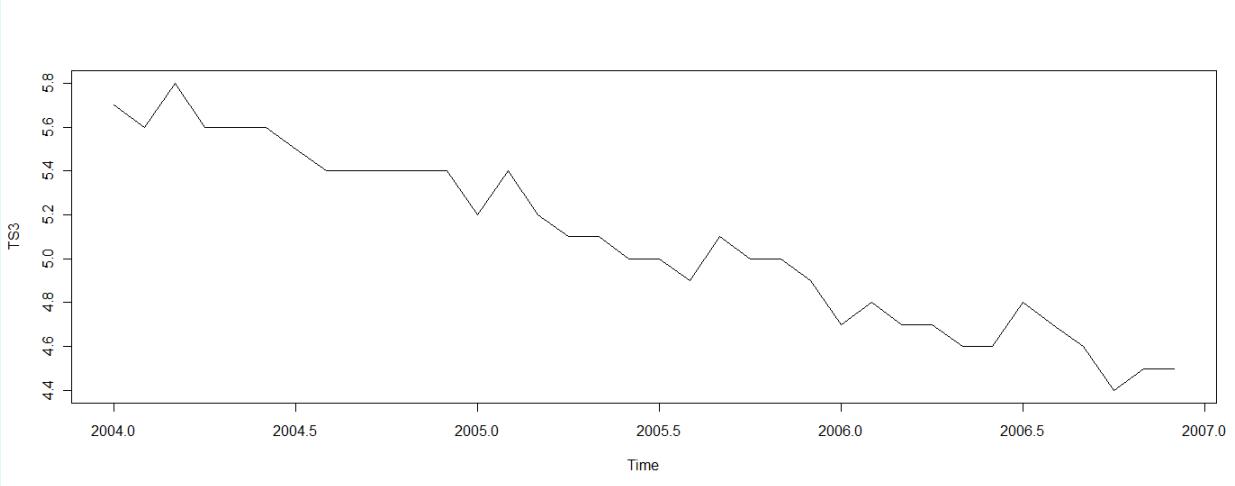
US unemployment rate from 1987 to 2007 are taken as a time series for analysis.

Data from

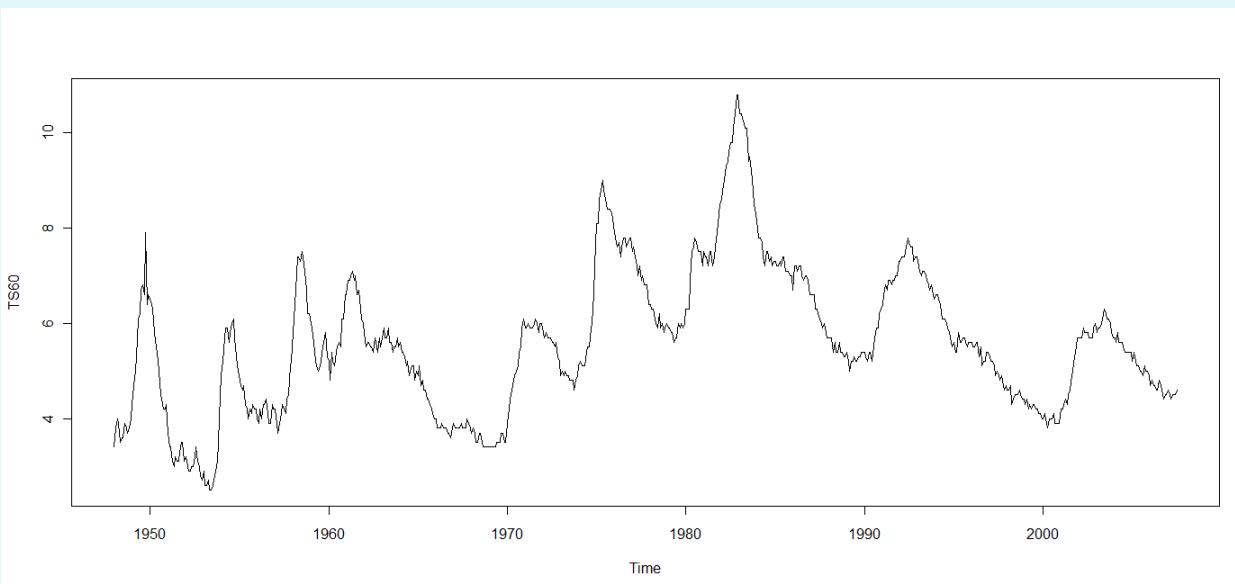
<http://www.stanford.edu/~xing/statfinbook/data.html>

Example

2004~2006



1948~2007



Example

An easier way: use TS functions to plot time series.

US unemployment rate from 1987 to 2007 are taken as a time series for analysis.

Data from

<http://www.stanford.edu/~xing/statfinbook/data.html>