Solving Problems in R

Felix Chan

December 2012

1 Background

The aims of this document is to provide a set of tasks that explores the different aspects of programming in R. It covers basic programming concepts such as types, conditional statement, loop and user-defined functions.

2 Types and Basic Operations

Task 1. Create a list $x = \{-2, -1, 0, 1, 2\}$ using the `c()` function.

Task 2. Create the same list using the `seq()` function and assign it to `y`.

Task 3. Create a list with the same element using the `rep()` function.

Task 4. Run the command `z<-as.matrix(y)`. What is the difference between `z` and `y`?

Task 5. Run the command `w<-z+y` and `w<-t(z)+y`. What is the difference between the two commands and what is the potential danger? For the last question, use the function `as.matrix()` to check if both `z` and `y` are matrices.

Task 6. What is the difference between `x*y`, `t(x)%*%y` and `x%*%t(y)`? **Hint:** What does `t()` do?

Task 7. Create a list $s = \{x : -10 \leq x \leq 10, x \in \mathbb{N}\}$, where $\mathbb{N}$ denotes the set of integers. Run the command `s%%2` and `s%/%2`. What do the operators, `%%` and `%/%` do?

3 User-defined Functions

Task 1. What does the command `source()` do?

Task 2. Write a user-defined function that takes one input and return 0 if the input is an odd number; return 1 if the input is an even number and return an error message `This is not an integer` if the input is not an integer. **Hint:** Look up conditional statement in help, specifically, the usage of `if-then-else`.

Task 3. Write a user-defined function that returns the sample mean, variance, skewness and kurtosis, without using the in-built function in R. **Hint:** Simple matrix algebra will be useful. Also look up the usage of `list()`.

Task 4. Write a user-defined function that will compute $y_t = \phi y_{t-1}(1 - y_{t-1})$ for any given initial value, y_0, coefficient, ϕ, and total number of iterations. Plot the outputs of the function over different vales of $\phi = \{0, 1, 2, 3.99\}$ and initial value $y_0 = \{0.1, 0.5\}$.

Task 5. Create a user-defined function that compute the log return of a time series.

4 Linear Regression

Task 1. Import the dataset `exchange_20120920.csv` using the `read.table()` command and store the output dataframe into `m`.

Task 2. Run the command `n<-names(m)`. What does the command do?

Task 3. Calculate the log-returns for the data using the function you have written above.

Task 4. Calculate the descriptive statistics for the data and its log-returns using the function you have written above.

Task 5. Test the covered interest rate parity (CIP) by using the `lm()` function. Recall the covered interest rate parity states that:

$$(1 + i_t^d) = \frac{F_t}{S_t}(1 + i_t^f) \tag{1}$$

where i_t^d and i_t^f are the domestic and foreign interest rate at time t, respectively; F_t and S_t are the forward and spot exchange rate at time t, respectively. **Hint:** Note that $\log(1 + x) \approx x$ if x is small.

Task 6. Write a procedure to recursively test the CIP using recursive and rolling windows. **Hint:** For recursive windows, the idea here is to select an initial sample size, say 500, then test CIP based on the first 500 observations. Record all the results from the estimation (parameter estimates and their t-statistics), then add the next observation into the sample, so that the sample size is now 501, retest CIP and record the results. Repeat the process by adding one observation to the sample at a time until it reaches the full sample. For rolling windows, the idea is similar, but the sample size remains the same. Select an initial sample size, test CIP and record the results, then remove the first observation from the sample and add the next observation at the end of the sample and repeat the process until the rolling windows reaches the end of the full sample. So if the initial window size is 500 then the first rolling window spans observations 1 to 500, the second rolling window spans observations 2 to 501 and so on...

5 Packages

Task 1. Search for a package called `rugarch` and install it using the `install.packages()` command.

Task 2. Search of a package called `urca` and install it using the `install.packages()` command.

Task 3. Load up these packages onto your workspace using the `library()` command.

Task 4. Test for a unit root for each variable in `exchange_20120920.csv` using Dicky-Fuller test. **Hint:** Look up `ur.df()` under the `urca` package.

Task 5. Estimate a GARCH(1,1) model using the `rugarch` package. **Hint:** Look up `ugarchspec()` and `ugarchfit()` functions.