An Internet Connected Financial Calculator

Yuxing Yan
Hofstra University

When discussing the meaning of β, I hope that I could punch a few keys on my calculator to show students IBM's β in 2010. When explaining the Sharpe ratio, I dream that my calculator could download the DELL's returns and a risk-free rate. In this paper, I demonstrate how to accomplish these tasks by calling a set of R programs. For instance, typing Beta("IBM", 2010) will offer IBM's β in 2010. Mastering R is not easy. Fortunately, teaching how to call prewritten R programs is much easier than teaching students how to use an ordinary financial calculator.

INTRODUCTION

After 8 years working as a consultant on financial databases and programming, recently I started teaching again. In the fall 2010, one of my assignments was an introductory finance course for undergraduates. In order to select an appropriate textbook, I searched many excellent ones. Finally, I settled with Corporate Finance by Berk and DeMarzo (2011). One of the notable features is its 13 data cases. For example, the data case at the end of Chapter 10 asks students to download historical price from Yahoo Finance for 12 stocks for various analyses. It is a great idea to utilize public information to help students comprehend concepts, formulae and theories. In terms of computation, the authors adopt the traditional approach by using ordinary calculator, financial calculator, and related Excel functions. In reality, this approach is employed universally by all finance textbooks. Here are two more examples: Ross, Westerfield and Gordon (2010) and Madura (2011). In this short paper, I argue that using an ordinary financial calculator is outdated. In our internet era, we need a *web connected* financial calculator.

For example, when discussing the meaning of β in CAPM (Capital Asset Pricing Model), I really hope that I could show students the β values for 50 stocks in 2010. When talking about the Sharpe ratio, I dream that my calculator could download return data for those 12 stocks mentioned above, and estimate their Sharpe ratios. When explaining the January effect, how about showing students the mean returns in different months for 10 stocks? How about just type Current_ratio("IBM") to get its current ratios? When explaining the size effect, if I could sort 500 stocks into 10 size portfolios and estimate their mean returns, it would have a huge impact on my students' learning. Unfortunately, an ordinary financial calculator could not achieve those, nor could Excel. In a sense, it is the time that an internet connected financial calculator should be introduced into our classrooms. For this very purpose, I recommend a substitute, a better and free substitute: R.

R AND FINANCE 101

R is open source statistical and computational software. It is publicly available at no cost. "No cost" only means that the software itself. We do have to spend time and efforts to learn it. Thus, it is not completely free since the opportunity cost is a fundamental concept in Finance 101. Luckily, the tiny cost of learning R for the purpose of teaching or leaning Finance 101 is negligible. Any interested instructor or student could download and install this wonderful software in 5 minutes. Learning R is not straightforward since a potential user has to spend tremendous time and efforts to master the language. However, when certain programs are supplied (prewritten), teaching undergraduate students how to call R functions for Finance 101 is much easier than teaching them how to use a financial calculator. Next, I will discuss why R fits perfectly into Finance 101 teaching.

The objective of teaching an introductory finance course is to help students to understand basic concepts, a set of formulae and the underlying logics, e.g., concepts of time value of money, present value, future value, IRR (Internal Rate of Return), YTM (Yield to Maturity) and NPV (Net Present Value). When asked to estimate the price of a bond with a given set of parameters (face value, coupon rate, maturity date and discount rate), students could use an ordinary calculator, a financial calculator, or Excel. In terms of understanding formulae and their underlying logics, the first approach of using an ordinary calculator is the best since students have to apply a formula or formulae explicitly to solve problems. On the other hand, a financial calculator and Excel are much easier to use. This is especially true during an exam since a financial calculator could save a couple of minutes which might be crucial to some students. However, the functions for both financial calculators and Excel are a black box because the underlying formulae are hidden from students.

When applying R, we could combine the advantages of using an ordinary calculator with the advantages of using a financial calculator or Excel. First, students can *see* the formulae they are calling. More importantly, most formulae written in R take the same forms shown on the textbooks. Second, calling a prewritten R program is straightforward. Third, most Finance 101 related R functions are simple. For example, we need just one line of R codes for a Present-value function, see below.

```
pv_f<-function(fv,r,n) fv/(1+r)^n
```

The above line is simple and easy to understand since it takes the same form in most finance textbooks. pv_f is our function name followed by the keyword of "function". The input parameters are included in the parentheses and the last part will be the contents of the function. To call the function, we have two methods for a set of input variable: position and keyword.

Method I: "position variables" approach, i.e., enter values in the order how they are defined. For example, we estimate the present value of $100 occurs in one year with a discount rate of 10%. According to the structure of the pv_f function, the order of inputs is: future value, effective period rate, and the number of periods. In other words, the meaning of each input variable depends on its order in the input set.

```
> pv_f(100,0.1,1)
[1] 90.90909
```

Method II: use a "keyword" approach, i.e., specify the corresponding keyword before each input value. For example, to input a value of 100 for the future value, we use "fv=100". Since R is case sensitive, we cannot use FV=100 because future value is defined as fv instead of FV.

```
> pv_f(fv=100,r=0.1,n=1)
[1] 90.90909
```

The advantage of using the "keyword" approach is that the order of input variables no longer plays a role, see the following equivalent three ways.

```
> pv_f(fv=100,n=1,r=0.1)
[1] 90.90909
> pv_f(n=1,fv=100,r=0.1)
[1] 90.90909
> pv_f(r=0.1,fv=100,n=1)
[1] 90.90909
```

Here are more one-line formulae written in R.

```
> fv_f<-function(pv,r,n) pv*(1+r)^n
> pv_perpertuity(c,r) c/r
> pv_growing_perpetuity(c,r,g) c/(r-g)
> PMT_f<-function(n,r,pv,fv)(pv-fv/(1+r)^n)*r/(1-1/(1+r)^n)
> r_continuous<-function(r_annual,m)m*log(1+r_annual/m)
```

Although one line is the simplest form for many financial formulae in Finance 101, it is a good idea to add an explanation. Below, the definitions of input variables are included. In addition, two examples are given on how to call this function. This is an extra benefit of using R instead of a financial calculator.

```
pv_f<-function(fv,r,n) {
 "
  Objective: estimate present value
     fv : future value
      r : effective period rate
      n : number of periods
     e.g., pv_f(100,0.1,1)          # order of inputs is vital
       [1] 90.90909
           pv_f(r=0.1,fv=200,n=1.5)# order does not matter
       [1] 173.3568
  "
  return(fv/(1+r)^n)
}
```

To see the above explanation, type the function name, i.e., *pv_f*.

```
> pv_f    # show the underlying codes and examples
```

In R, there are two ways to add comment lines. A "#" at the beginning of a line (or a phrase) indicates that this is a comment line (phrase). The R complier will ignore the contents after a "#" when it compiles the program. For a multiple-line comment, it is cumbersome to write many "#"s. In such cases, we use a pair of double quotation (") to encircle all our comments, see the example above.

It works fine if we write a simple R program then call it by using interactive mode. However, it is much better to put all R programs into just one text file. Assume that we have 50 R functions we need for Finance 101. We could combine them together and save the final file as fin101.txt. The following command could be used to activate (load) those functions.

```
>source("c:/fin101.txt")
```

To view the all available functions, just type *ls()*.

```
> ls()     # list all variables and functions
```

One of the most important advantages of using R is that we could use publicly available information, such as historical stock price, and latest several years' balance sheets, income statements or cash flow statements. It is a shame if we could not utilize such huge quantity of public information in our classrooms. With R, we could easily download historical price for individual stocks, see the following one-line R codes to download the IBM's daily price.

```
> x<-read.csv("http://chart.yahoo.com/table.csv?s=IBM",header=T)
```

To view the first and last several records, we use the *head()* and *tail()* functions.

```
> head(x)
        Date   Open   High    Low  Close  Volume Adj.Close
1 2011-05-19 170.86 171.40 169.32 170.59 3588000    170.59
2 2011-05-18 170.10 171.19 169.46 170.44 4154300    170.44
3 2011-05-17 167.85 171.41 166.53 170.50 8773200    170.50
4 2011-05-16 169.81 170.64 168.31 168.86 4662200    168.86
5 2011-05-13 171.70 172.15 169.44 169.92 5167400    169.92
6 2011-05-12 169.65 172.77 168.65 172.24 5138500    172.24
```

To download and save the IBM's data for a further analysis, we have two lines.

```
> x<-read.csv("http://chart.yahoo.com/table.csv?s=IBM",header=T)
> write.table(x,file="c:/test_R/ibm.csv",sep=",",quote=F,row.name=F)
```

For the Data Case at the end of Chapter 10 of Berk and DeMarzo (2010), students are reluctant to download data for 12 stocks, let alone 50 stocks. Fortunately, we could write an R program to download data for a large quantity of stocks such as 50. The R program used to download historical prices from Yahoo Finance is presented in Appendix E.

Table 1 summaries the advantages and barriers of applying R in our basic finance teaching. The flexibility means that users could adopt their own favorite function names. For instance, we could rename *pv_f* as *pv_function*. When undergraduate students pursue a master degree, the knowledge of R will definitively give them an edge. The knowledge and skills of R add certain weight when graduates try to land a Wall Street job since many financial institutions use S-Plus, a cousin of R. Unfortunately, there are some barriers for applying R in our teaching. The most critical barrier is the attitude of our instructors. Some might feel uncomfortable to learn a new tool, while others might be too occupied with their research.

TABLE 1
ADVANTAGES AND BARRIERS OF USING R IN FINANCE 101

Panel A: advantages of using R	
1	No cost (free to download)
2	Not a black box (transparent in terms of formulae and logics)
3	More flexible than a financial calculator or Excel. For example, a user could generate their own functions by renaming the existing functions
4	Users could find out examples for each function
5	Could estimate beta, alpha, illiquity measure, CAPM
6	Could download data from internet such as Yahoo Finance
7	Could handle large quantity of data, such as estimate 5,000 beta
8	Way more powerful than both financial calculators and Excel For example, R has powerful graph ability
9	Extensible for many extra functionality
10	Very useful for doing real research
11	Good for a student's CV
12	R is used intensively in the financial industry
13	Many researchers around the world continue to develop R
14	More than 3 dozen packages related to finance
Panel B: barriers of using R	
1	Most instructors don't know R
2	It is easy to design a close book exam with a financial calculator
3	No finance textbooks includes R
4	The current authors might be reluctant to change their textbooks to incorporate R
5	The current publishers might be reluctant to change
6	Resistance from the manufacturers of financial calculators
7	Mentality

I have written several dozen R programs related to Finance 101. It is quite easy to activate (load) them by issuing the following one-line command. Remember to use *ls()* to list all available functions.

```
> source("c:/test_R/fin101.txt")
```

Let us look at an example of estimating multiple IRRs. A publisher offers Mr. Clinton a down payment of $550,000 plus $1m when the proposed book published in year 4. Assume that he needs 3 years to finish the book and the annual opportunity cost is $500,000, what are the IRRs for this project? Obviously, a financial calculator would fail because of multiple IRRs. Usually, I use Excel to explain. However, using R is much easier, see codes below.

```
> x<-c(550000,-500000,-500000,-500000,1000000)
> IRR_f(x)
[1] 0.072 0.337
```

To get the beta for IBM in 2010, we issue the following command.

```
> Beta("IBM",2010)
[1] "Beta for IBM in 2010 = 0.7718"
```

Most finance textbooks spend one chapter to discuss financial statement analysis. For students, downloading financial statements is always a headache. With R, we could easily look or save the latest years' Balance Sheet, Income Statement or Cash Flow Statements. To get DELL's Annual Income Statements, we have just one-line.

```
> x<-get_statement("DELL","IS","A")
Annual Income Statement for DELL
```

To view the first several lines, again we use the *head()* function.

```
> head(x)
                                  2011-01-28 2010-01-29 2009-01-30 2008-02-01
Revenue                              61494  52902      61101      61133
Other Revenue, Total                  NA       NA         NA         NA
Total Revenue                        61494  52902      61101      61133
Cost of Revenue, Total               50098   43641      50144      49462
Gross Profit                         11396   9261       10957      11671
Selling/General/Admin. Expenses, Total 7234   6228        6966       7538
```

To save a downloaded Balance Sheets as a csv (Comma Separated Values) file for further analysis, we have the following one-line codes, where "BS" for Balance Sheet and "A" for Annual.

```
> save_statement("IBM","BS","A","c:/test_R/bs_ibm.csv")
Annual Balance Sheet for IBM
[1] "Your saved file is ==>c:/test_R/bs_ibm.csv"
```

The Ratio Analysis plays a central role in the Financial Statement Analysis. To estimate the Current Ratios (defined as the Current Assets divided by the Current Liabilities) for Microsoft (MSFT) by using the latest several years' Annual Balance Sheets, we have the following one-line codes.

```
> current_ratio("msft")
Annual Balance Sheet for msft
2010-06-30 2009-06-30 2008-06-30 2007-06-30
     2.129      1.823      1.447      1.691
```

I have little doubt that the codes underlying many R programs, such as downloading the historical price data, are very complex for most finance professors and business major students. Because of this, there is little chance that a finance/business major student could learn their meanings and logics quickly in terms of programming. However, I argue that the understanding the underlying codes is irrelevant since no finance student tries to understand the internal structure of a financial calculator!

CONCLUSION

In this paper, I demonstrate that R could be used as an internet-based next generation financial calculator. Applying R in our Finance 101 teaching will reap multi-benefits for our students. Calling simple functions, such as PV, FV, PV(annuity), PV(annuity due), is straightforward, e.g., pv_f(fv=100,r=0.1,n=1). In addition to those functions, R offers more features which are not available with financial calculators or Excel. To get β for IBM in 2004, we type Beta('IBM',2004). The prewritten program will automatically download historical price data from Yahoo Finance and estimate the β. For the Sharpe ratio for DELL in 2002, we issue Sharpe("DELL", 2000'). In addition, we could easily download historical price data or latest several years balance sheet, income statements or cash flows by just punching a few keys. In summary, although mastering R is not easy, teaching students how to call

prewritten R programs for Finance 101 should not be more difficult than teaching them how to use a financial calculator.

REFERENCES

Berk, Jonathan, Peter DeMarzo (2011). Corporate Fanance, 2nd edition, *Pearson Education, Ltd.* Madura, Jeff, 2011, Personal Finance, 4th edition, *Pearson Education, Ltd*.

Madura, Jeff (2011). Personal Finance, 4th, Prentice Hall.

Ross, Stephen A., Randolph W. Westerfield and Bradford D. Jordon (2012). Fundamentals of Corporate Finance, 9th edition, *McGraw-Hill Irwin*.

APPENDIX A: STEPS TO CALL PREWRITTEN R FUNCTIONS FOR FINANCE 101

Step 1: R's installation

To install R, we have 4 steps. For a new user, please apply the default settings. After you finish the installation, an R icon will appear on your desktop.

```
Step 1: go to web page http://www.r-project.org
Step 2: click "Download" on the left hand-side
Step 3: choose a mirror address
Step 4: Download appropriate software (PC, Mac)
```

Step 2: Launch R by clicking R's icon on your screen.

Step 3: Issue the following command to activate functions, see Appendix B for fin101.txt.

```
source("c:\fin101.txt")
```

Step 4: just type ls() to view all prewritten functions

```
>ls()
```

To know how to use each prewritten function, just type its name.

```
>pv_f()
```

APPENDIX B: A FEW R COMMANDS

The following R commands will help you to use R more efficiently. ">" is the R prompt.

```
> # this is a comment line
> ls()           # list all functions
> pv_f           # will show you the function of pv_f
> q()            # quit R program
> quit()         # 2nd way to quit
> pv<-100        # assign 100 to a variable called pv
> x<-1:10        # assign 1, 2, 3, …, 10 to x
> head(x)        # see the first couple of lines of x
> tail(x)        # see the last several lines of x
> x<-10; y<-20   # put two commands together using a semicolon
> rm(x)          # remove x
> rm(x,y)        # remove x and y
> rm(list=ls()) # remove all variables or functions
> r<-seq(0,0.6,by=0.01) # r is a vector from 0,0.01,0.02, .. 0.6
```

APPENDIX C: A SIMPLE EXAMPLE OF FIN101.TXT

```
#-----------------------------------------------#
#-- Present value function ---------------------#
#-----------------------------------------------#
pv_f<-function(fv,r,n) {
 " Objective: estimate present value
     fv : future value
      r : effective period rate
      n : number of periods
     e.g., pv_f(100,0.1,1)      # order of inputs vital
       [1] 90.90909
           pv_f(r=0.1,fv=200,n=1.5)# order does not matter
       [1] 173.3568  "
  return(fv/(1+r)^n)
}
#-----------------------------------------------#
#-- Future value function ----------------------#
#-----------------------------------------------#
fv_f<-function(pv,r,n) {
    " Objective: estimate future value
         pv   : present value
         r    : effective period rate
         n    : number of periods
     e.g., fv_f(100,0.1,1)
         [1] 110
    "
    return(fv*(1+r)^n)
}
#-----------------------------------------------#
#-- Bond price function ------------------------#
#-----------------------------------------------#
bond_price<-
function(face_value,n_year,coupon_rate,r_annual,freq){
    "
     Objective    : estimate present value of a bond
       face_value : face value of a bond
       n_year     : number of years to maturity
       coupon_rate: coupon payment = coupon_rate*face_value/freq
       freq       : number of coupon payments per year
       r_annual   : annual discount rate
       e.g., bond_price(100,30,0.10,0.05,1)
       [1] 176.8623
       zero-coupon bond is a special case
       bond_price(100,30,0,0.06,1)
       [1] 17.41101
    "
    pmt<-coupon_rate*face_value/freq
    r<-r_annual/freq
    n<-n_year*freq
    return(pmt/r*(1-(1+r)^(-n)) +face_value/(1+r)^n)
}
```

APPENDIX D: EXAMPLE OF CALLING VARIOUS PREWRITTEN R PROGRAMS (FINANCE 101)

Function name	Examples
pv_f(fv,r,n)	pv_f(100,0.1,1) pv_f(fv=100,n=1,r=0.08) pv_f(n=2,r=0.05,fv=400)
fv_f(pv,r,n)	fv_f(100,0.1,2)
pv_annuity(pmt,r,n)	pv_annuity(100,0.06,10)
pv_annuaity_due(pmt,r,n)	pv_annuity_due(120,0.10,11)
fv_annuity(pmt,r,n)	fv_annuity(100,0.07,23)
fv_annuaity_due(pmt,r,n)	fv_annuity_due(120,0.068,10)
pmt_f(pv,r,n,fv)	pmt_f(980,0.1,20,1000)
n_period(pv,ptm,r,fv)	n_period(1234,0.1,10,4,1000)
YTM_f(face_value,coupon_rate,n,freq)	YTM_f(100,0.08,20,2)
Bond_price(face_value,r,coupon_rate,freq)	Bond_price(100,0.1,0.8,2)
NPV_f(cashflows,r)	cash<-c(-100,50,60,120) NPV_f(cash,0.12)
IRR_f(cashflows)	cash<-c(-100,50,60,120) IRR_f(cash)
Beta(ticker,year)	Beta("Dell",2010)
Sharpe(ticker,beg_year,end_year,freq)	Sharpe("Dell",2001,2002, "m")
Yahoo_price(ticker,begdate,enddate,freq)	Yahoo_price("IBM",1926,2010, "m")

APPENDIX E: R PROGRAM TO DOWNLOAD PRICE THEN CALCULATE RETURN

```
#---------------------------------------------#
#-- Download price data from Yahoo Finance -----#
#---------------------------------------------#
yahoo_price<-function(ticker,beg_year,end_year,d_or_m){
   "Objective: download historical price data from
               Yahoo finance, estimate returns
      ticker : ticker of a stock
     beg_year: beginning year
     end_year: ending year
     m_or_d  : 'm' for using monthly data
               'd' for using daily data
         -99 : missing code
       e.g., : data<-yahoo_price('IBM',2000,2000,'m')
    ticker     date   open   high    low  close volume adj_close return
    IBM 20000103 112.44 124.75 109.62 112.25  9073100   99.07 -99.000000
    IBM 20000201 112.37 119.75 100.94 102.75  7200400   90.78  -0.083678
    IBM 20000301 102.00 128.25  99.50 118.37  8797500  104.58   0.152016
   "
   k1<-'http://ichart.finance.yahoo.com/table.csv?'
   k2<-'s=$V&amp;amp;a=00&amp;amp;b=1&amp;amp;c=$Y1&amp;amp;d=11&amp;amp;'
   k3<-'e=30&amp;amp;f=$Y2&amp;amp;g=$D_M&amp;amp;ignore=.csv'
   k<-paste(k1,k2,k3,sep='')
   k<-sub('$Y1',toString(beg_year),k,fixed=T)
   k<-sub('$Y2',toString(end_year),k,fixed=T)
   k<-sub('$D_M',d_or_m,k,fixed=T)
   k4<-sub('$V',ticker,k,fixed=T)
   x<-read.csv(k4)
   d<-data.frame(format(as.Date(x[,1]),'%Y%m%d'),as.real(x[,2]),
     as.real(x[,3]),as.real(x[,4]),as.real(x[,5]),as.real(x[,6]),
     as.real(x[,7]))
   d<-d[order(d[,1]),]
   t<-dim(d)
   n<-t[1]
   stock<-data.frame(ticker, d[1:n,1:7],-99) # -99 missing code
   stock[2:n,9]<-(stock[2:n,8]-stock[1:n-1,8])/stock[1:n-1,8]
   colnames(stock)<-c('ticker','date','open','high','low',
   'close','volume','adj_close','return')
   stock[,9]=round(stock[,9],digits=6)  # six decimal places
   return(stock)
}
```

APPENDIX F: R PROGRAMS FOR BETA, SHARPE AND ILLIQUIDITY FUNCTIONS

```
#-------------------------------------------------------#
#---Beta: just stock ret and market --------------------#
#-------------------------------------------------------#
Beta<-function(ticker,year){
   "  > Beta('IBM',2010)
      [1] 'Beta for  IBM in 2010 = 0.7718'
   "
   k<-'http://chart.yahoo.com/table.csv?s=$S'
   k2<-sub('$S',ticker,k,fixed=T)
   t<-read.csv(k2,header=T)
   n<-nrow(t)
   ret<-data.frame(as.Date(t[1:(n-1),1]),(t[1:(n-1),7]-
t[2:n,7])/t[2:n,7])
   colnames(ret)<-c('date','ret'); rm(k,k2,t,n)
   t<-read.csv("http://chart.yahoo.com/table.csv?s=^GSPC",header=T)
   n<-nrow(t)
   mkt<-data.frame(as.Date(t[1:(n-1),1]),(t[1:(n-1),7]-
t[2:n,7])/t[2:n,7])
   colnames(mkt)<-c('date','mkt')
   data<-merge(ret,mkt)
   d2<-subset(data,format(data[,1],"%Y")==year)
   beta<-round(coef(lm(d2$ret~d2$mkt))[2],digits=4)
   show(paste('Beta for ',ticker,'in',year,'=',beta))
}
# ------------------------------------------------- #
# --- Sharpe ratio -(Note: ignore risk-free rate) -- #
# ------------------------------------------------- #
Sharpe<-function(ticker,beg_year,end_yer,d_or_m){
   " Objective : estimate beta of a stock
       ticker  : stock symbol
       beg_year: begin year
       end_year: ending year
       d_or_m  : using monthly or daily data
       >Sharpe('IBM',1990,1994,'m')
   "
   d<-yahoo_price(ticker,beg_year,end_yer,d_or_m)
   mean_ret<-mean(d[,9])
   std<-sd(d[,9])
   return(mean_ret/std)
}
```