A new method to teach Financial Modeling using Excel

Yuxing Yan[1]
1/11/2017

Abstract

When applying Excel to finance, there exist many issues, such as how to focus on hands-on experience, how to inspire students to learn Excel, how to teach macros and how to get data into our spreadsheets in the first place. This paper offers a new approach with the following features. First, during each lecture, students do at least two hands-on exercises. Second, students are given many explanations and formulae. Thus, they could learn Excel by themselves. Third, students access a huge amount of data such as US GDP, US national debt, unemployment rate, CPI, Fama-French factors, risk-free rate, historical stock prices, latest several years financial statements. Fourth, students could apply an efficient way to download data, about 3 second for each time series. Fifth, all supporting materials are written in a text format. Thus, any instructor could modify them easily. The setup is trivial for both students and instructors: one-page instruction on R installation plus one-line R codes.

Keywords: financial modeling, Excel, hands-on experience, data intensive learning, R

[1] Department of Economics and Finance, Canisius College, 2001 Main Street, Buffalo, NY, 14208. Tel: (716) 888-2604. Email: yany@canisius.edu.

1. Introduction

Nowadays, we are overwhelmed by a huge amount of information (data). One of the major feedbacks from employers regarding to business school graduates is that they should know how to handle various types of data. Fortunately, many rich sources of publicly-available economic, financial, and accounting data are available for classroom use. Here is a partial list: Yahoo!Finance[2], Google Finance[3], Wall Street Journal[4], Professor French's data library[5], SEC filings[6], Census Bureau[7], Bureau of Labor Statistics,[8] and Bureau of Economic Analysis[9]. Rich data are available, such as daily, weekly or monthly stock price data and corporate financial statements from Yahoo!Finance. More than 385,000 U.S. and international time series are available from the St. Louis Fed's Federal Reserve Economic Data Library (FRED)[10]. Daily, monthly or annual Fama-French factor data, risk-free interest rate series and equity market excess return data since 1926 up to now are available from Professor French's data library. The SEC web page provides access to the 10K and 10Q filings for 580,225 publicly traded companies from the first quarter in 1993 to the 2nd quarter in 2016.[11] How to use those free data is a really challenging task faced by professors at business schools around the world.

In terms of skills needed for business school's graduates, the second feedback is that students should have a good understanding of Excel. There are several aspects of this type of skills. First, students should master many basic Excel functions, such as pv(), fv(), rate() and nper() functions. Second, they should know how to use Excel to input and process data, run various statistics analysis and output them. Third, some companies might demand more advanced techniques, such as Macro and VBA (Visual Basics Applications). For the first one, many instructors teach students various types Excel functions. For the aspect related to data, one of the big issues is how to download data to our spreadsheet in the first place. This is always a big headache. In other words, it is a bottleneck to train our students applying Excel skills to

[2] http://finance.yahoo.com.
[3] http://www.google.com/finance
[4] http://www.wsj.com
[5] http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html
[6] http://www.sec.gov/edgar.shtml
[7] http://www.census.gov/compendia/statab/hist_stats.html
[8] http://download.bls.gov/
[9] http://www.bea.gov/
[10] https://research.stlouisfed.org/fred2/
[11] The value of 580,225 is based on the unique CIK from the index files from the first quarter in 1993 to the second quarter in 2016.

process data related to economics, finance and accounting. In short, we need an efficient way to access and download various types of data. In this paper, a good solution is offered. For the advanced knowledges of Macro and VBA, we argue that it is not a good idea to teach such a skill to undergraduates from business schools. The reason is that learning R or Python is more meaningful and less time consuming. R and Python are way too powerful than macro and VBA. For our teaching, an alternative is offered: just two lectures to give students a basic idea about macro or VBA. Later in the paper, we would explain how to do so.

To make teaching more efficient, many instructors have tried many innovated methods. Zhang (2014) shows how to incorporate Excel tools into finance teaching. Excel data table, charts, scenario manager, goal seek and solver are powerful tools in answering many finance questions. He suggests all business schools should integrate Excel into the core business course and major courses. One potential issue for Zhang's approach is how to download data from various data sources effectively. Carini, Kuh, and Klein (2006) demonstrate that student engagement would improve a student's academic performance. Cagle, Glasgo and Hyland (2010) conduct a test of using spreadsheet to do assignments. They find that students doing homework with spreadsheets perform better than those without. In this paper, we show by combining Excel tools with real-world data, we could improve teaching effectiveness greatly.

Over years, we have taught Financial Modeling courses at several schools over a dozen times. The textbook used is "Financial Modeling" by Benninga. In 2015, we used the same approach at Canisius. After just two weeks, we realized that there was a big mismatch between the textbook and the backgrounds of students. There was no way that majority students could satisfy the minimum level required by the book. As the consequence, a complete different approach was adopted. Based on our multiyear programming experience, a so-called R-assisted teaching and learning environment came to exist for the course of "Financial Modeling using Excel". In total, we have produced materials for 27 chapters. For each chapter, 20 explanations are available. Since hands-on experience is crucial for learning Excel, over 50 in-class exercises are developed. In addition, over 40 formulae are available at students' finger tips. It is quite difficult to image for a course over 300 useful web links could be used. Fortunately, this becomes feasible by using our approach. The most important contribution is that we have developed an efficient way to get data, from multiple public sources, to our spreadsheets.

The paper is organized as follows. The next section outlines the setup for both students and instructors. Section 3 illustrates the first and last weeks' lectures. Section 4 shows how to conduct in-class exercises. Sections 5 and 6 offer an efficient way to download data from several public sources and explain 3 functions used to download and save data. To help students understand Excel functions, the next section introduces mimic functions with and without Excel sign convention. Section 8 presents a new financial calculator. The beauty of this financial calculator is that for each function, students could see its objective, definitions of all input variables, the formula used plus a few examples. Section 9 shows how to spend just one week (two lectures) to teach/learn Excel macro and VBA. The last section concludes and points out several potential extensions.

2. Setup for students and instructors

The setup could be summarized with one sentence: R installation plus one line R codes. To download and install R, we have 5 steps:

Step 1: Go to http://www.r-project.org

Step 2: Click "CRAN" under "Download" (left-hand side)

Step 3: Choose a mirror address

Step 4: Choose appropriate software (PC or Mac)

Step 5: Click "base".

Usually, it takes about 5 minute to download and install R. For the whole course, an instructor needs to give his/her students just one line of R codes. Below is one example.

```
>source("http://canisius.edu/~yany/excel.R")
```

Where ">" is the R prompt. Obviously, for instructors from other schools, they should use their own websites. For an instructor at my school, i.e., Canisius College, the above line contains one line, see an example below.

```
source("http://canisius.edu/~yany/fmExcel/w01.R")
```

Obviously, w01.R is for the first week. When teaching this course, the only thing that an instructor has to do is to change this line every week. For the second week, w01 in the above line

should be replaced by w02. With such a setup, an instructor's extra efforts would be kept at a minimum. [12]

3. The first and the last weeks

Assume that the following the command line is issued.

```
>source("http://canisius.edu/~yany/excel.R")
```

Assume further that the excel.R file contains the first week's contents. After issuing the above command, the result is shown below.

```
 *-----------------------------------------------*
 * Financial Modeling using Excel                *
 *-----------------------------------------------*
 * Chapter 1: R installation & Excel basics (I)  *
 *-----------------------------------------------*
 * >c1        # go to chapter 1                   *
 * >fm        # back to this menu                 *
 *-----------------------------------------------*
>
```

The above instruction is very clear: c1 for going to chapter 1, and fm would come back to this main menu. Since R is case-sensitive, we have to type c1 instead C1. This is true for lower-case 'fm'. After typing c1, the following window would pop up.

```
> c1()
 *---------------------------------------*
 * Chapter 1: Introduction               *
 *---------------------------------------*
 * Functions    Utilities                *
 * ------------------------------------- *
 * pv_f         e1                       *
 *---------------------------------------*
 * >pv_f        # see its usage          *
 * >c1          # back to chapter 1      *
 * >fm          # back to the  main menu *
 *---------------------------------------*
```

From above window, we see two columns called Functions and Utilities. The instruction at the bottom tells us that typing 'pv_f' would lead to the help related to the usage of this specific function. Remember this functionality: typing a function name could get its related help, see below.

[12] Please contact us if any instructor is interested in adopting the same approach described in the paper.

```
> pv_f
function(r,n,fv) {
"Objective    : estimate present value
                     fv
  formula used: pv= --------
                   (1+r)^n
           pv : present value
           fv : future value
           r  : effective period rate
           n  : number of periods
 Example 1: > pv_f(0.1,1,100)  # meaning of inputs depend on order
             [1] 90.90909

 Example 2: > pv_f(0.045,5,225)
             [1] 180.5515

 Example 3: > pv_f(fv=100,r=0.1,n=1)  # meaning depend on keywords
            > pv_f(r=0.1,fv=100,n=1)
            > pv_f(n=1,r=0.1,fv=100)

            Note: The above three give the same result
";pv_f_(fv,r,n)
}
```

Compared with Excel functions, the function called pv_f() is quite straightforward and user-friendly since it explains the meanings of three input variables, the formula used (the exactly same format as shown on our textbooks), plus a few examples. After typing e1, we will see 20 explanations related to chapter 1, see below.

```
> e1
function(i){
" i  Chapter 1: R installation and Excel Basics
  -  ------------------------------------
  1  How to install R
  2  one line R codes for this course
  3  Excel is a two-dimensional spread sheet
  4  =A1 What does it mean?
  5  +, -, /, and * have their normal meanings
  6  format a cell
  7  round() function
  8  exponential function
  9  power function
 10  log() and ln() functions
 11  sqrt() function and power function
 12  count() and counta() functions
 13  average() function
 14  stdev() stdev.p() and stdev.s()
```

```
 15  max()   and min() functions
 16  floor() and ceil() functions
 17  string variable
 18  True and False
 19  isnumber() and istext() functions
 20  change the name of your spread sheet

 Example #1:>e1    # see the above list
 Example #2:>e1(1) # see the 1st explanation

";explain1_(i)
}
```

The instruction at the bottom tells us how to find each explanation. After typing e1(15), the 15th explanation would appear, see below.

```
> e1(15)
Max() function
////////////////////////////////////////

  The max() function would give us the maximum value for a
    given set of values (cells).

     = max(1,2)

     = max(1,-1,2,67)

     = max(A1:A10)

     = max(A1:C250)

     similarly, you could try the min() function.

////////////////////////////////////////
>
```

In order to show all 27 chapters plus a few utility functions (submenus), the last week is called.

```
>source("http://canisius.edu/~yany/fmExcel/w15.R")
```

Note that when teaching, an instructor doesn't give the above line to his/her students. Instructors simply modify the contents of excel.R by replacing w01.R with w15.R. Again, our objective is to minimize the efforts by any potential instructor. After teaching this course a few times, we argue that our approach might be better, at least in certain areas, than some famous platforms, such as D2L. After issuing the above command, the following window would come out.

```
*------------------------------------------------------------*
* Financial Modeling using Excel                             *
*------------------------------------------------------------*
*  1: R installation/Excel basics(I)                         *
*  2: Time value of money            17: Excel basics (II)   *
*  3: R basics                       18: Widely used functions *
*  4: Sources of open data           19: vlookup, solver     *
*  5: Financial statement analysis   20: Data input          *
*  6: Risk vs. return                21: Data manipulation   *
*  7: Interest rate                  22: Data output         *
*  8: Bond evaluation                23: Simple graph        *
*  9: Stock evaluation                                       *
* 10: CAPM                           24: Matrix manipulation *
* 11: Fama-French 3 factor, Sharpe                          *
* 12: Statistical tests              25: Macro(1) record micros *
* 13  Black-Scholes options model    26: Macro(2) copy-paste *
* 14: Monte Carlo Simulation                                *
* 15: Portfolio Theory                                      *
* 16: VaR (Value at Risk)            27: Pivotal table       *
*------------------------------------------------------------*
* Utilities                                                  *
*------------------------------------------------------------*
* ice            : in class exercises                       *
* fincal         : a free financial calculator              *
* mimicExcel     : mimic Excel                              *
* allDataFunction: many functions related to data           *
* dataCases      : 7 data cases                             *
* showFormula    : show various finance formulae            *
* usefulLinks    : over 300 useful web links                *
* funs           : several fun submenus                     *
*------------------------------------------------------------*
* >c1            # go to chapter 1                          *
* >fm            # back to this menu                        *
*------------------------------------------------------------*
```

In total, there exist 27 chapters. Chapters 1 to 16 are associated with finance while rest is centered on Excel skills. For each week, one finance chapter plus one excel chapter usually will be taught. For each chapter, 20 explanations are available with the format, such as e2 for a list of 20 explanations for chapter 2, while e14 offers a list of 20 explanations for chapter 14. The second part of the above menu shows several utilities, such as *ice* (in Class Exercises), *fincal* (a financial calculator), *allDataFunctions* (on how to download and save data) and *usefulLinks*. In the next few sections, some of them would be discussed in detail.

4. Hands-on experience

As Zhang (2014) shows that for learning Excel, a hand-on experience is critical. With this in mind, for each lecture students would have at least 2 hands-on exercises. To satisfy this, a function called *inClassEx()* could be used. Its short form is *ice*.[13] After typing *ice*, a list of exercises would show. For example, for the 1st week, we could see the following result.

```
> ice
function(i){
" i   Description
  -   ------------------------------------
  1   =A1 What does it mean?
  2   showFormula function
  3   When developer is not available
  4   type =is    and if() function etc.
  5   conditional formatting, vlookup()

 Example 1:>ice        # show all exercises
 Example 2:>ice(1)     # see the first one

";inClassEx_(i)
}
>
```

Following the instruction, typing *ice(2)* would lead to the second exercise.

```
> ice(2)
   Step 1: Highlight and copy the following 6 lines

Function showFormula(x As Range)
    If x.HasFormula Then
        showFormula = x.Formula
    Else: showFormula = x
    End If
End Function

   Step 2: Click "Developer" on your menu bar
   Step 3: Click "VBA" -> Insert -> Module
   Step 4: Right click your mouse and paste
   Step 4: Click "File" -> Close and Return to Microsoft Excel

   Now, you can try this function =showFormula(A1)
        assume there is a formula in cell A1
>
```

Since for most schools, students meet professors twice each week, there are over 50 in-class exercises in total, see Appendix B for a complete list.

[13] Users could generate their short names. For example, aa=ice. Then aa(1) would be equivalent to ice(1).

5. An efficient way to download data

As mentioned before, one big challenging issue for learning Excel is how to download data into our spreadsheets in the first place. At the moment, many finance professor would use Yahoo!Finance or other similar sources to download historical stock data to finish their exercises, such as estimate the market risk for individual stocks. There is no doubt in our mind that this is a good achievement of using the real world data. Even so, there is mounting difficulty in terms of using data for our financial modeling courses. First, it is not feasible for an instructor to introduce more than one data source since they could not afford to spend so much time trying to understand the structures of different data platforms. Second, even for the same data source, it is really time consuming for students to download data for multiple stocks. We never heard that any professor would ask his/her students to download and process 50 stocks' historical information. Third, it is almost impossible for an instructor to show data related to economics, finance and accounting at the same time. For all those purpose, our method overcomes all those shortcomings. The submenu is called *allDataFunctions*, see the following window.

```
> allDataFunctions()
 *-----------------------------------------------------------*
 * All data related functions                                *
 *-----------------------------------------------------------*
 * Economics           Finance              Accounting       *
 *-----------------------------------------------------------*
 * show_usGDPannual    showffMonthly       getBSannual       *
 * show_usGDPquarterly showffDaily         getBSquarterly    *
 * show_usUnemployRate showAaaYieldMonthly getISannual       *
 * show_usDebt_annual  showAaaYieldDaily   getISquarterly    *
 * show_usCPI_annual   showBaaYieldMonthly getCFannual       *
 * show_usCPI_monthly  showBaaYieldDaily   getCFquarterly    *
 * show_euroDollar_1m  getDailyPrice                         *
 * show_dollarIndex    getMonthlyPrice                       *
 * show_goldPrice      getSP500daily       saveYan           *
 * show_fedFundRate    getSP500monthly     saveFinStatement  *
 *-----------------------------------------------------------*
 * >show_usGDPannual   # find the useage of this function    *
 * >fm                 # back to the main menu               *
 *-----------------------------------------------------------*
>
```

Three big categories are available: economics, finance and accounting. For most functions, two common starting phrases are *show_* and *get_*. The main difference is that functions start with get_ would retrieve the current data from certain public data sources, while functions start with

show_ would retrieve data from the author's web page. Again, typing a function name would show its usage.

```
> show_usGDPannual
function(n=2){
"Objective : show US annual GDP value
       n   : number of observations (default is 2)
             n > 0 for the first n obs
             n < 0 for the last  n obs
             n = 0 for all obs

      source:  http://www.usgovernmentspending.com/download_raw
      range  : 1929 -2015
      Unit   : billion

  Example 1: > show_usGDPannual()
             DATE GDP_CURRENT GDP2009DOLLAR
           1 1929       104.6        1056.6
           2 1930        92.2         966.7

  Example 2: > show_usGDPannual_(-3)
              DATE GDP_CURRENT GDP2009DOLLAR
               85 2013     16663.2       15583.3
               86 2014     17348.1       15961.7
               87 2015     17947.0       16348.9

  Example 3: > x=show_usGDPannual(0)
               In billion dollar
             > dim(x)
               [1] 87  3

";show_usGDPannual_(n)
}
>
```

For the "show" categories, we have three possible outputs: beginning n lines (a positive integer), the ending n lines (a negative integer) and whole data set (a zero). The default value of n is 2, i.e., when no input value. For example, *show_usGDPannual(10)* shows the first 10 lines, while *show_usGDPannual(-5)* shows the last 5 lines. If we don't include any input, *show_usGDPannual()* will be equivalent to *show_usGDPannual(2).*

```
> show_usGDPannual()
  DATE GDP_CURRENT GDP2009DOLLAR
```

```
1 1929       104.6        1056.6
2 1930        92.2         966.7
> show_usGDPannual(-4)
   DATE GDP_CURRENT GDP2009DOLLAR
84 2012     16155.3       15354.6
85 2013     16663.2       15583.3
86 2014     17348.1       15961.7
87 2015     17947.0       16348.9
> show_usGDPannual(4)
  DATE GDP_CURRENT GDP2009DOLLAR
1 1929       104.6        1056.6
2 1930        92.2         966.7
3 1931        77.4         904.8
4 1932        59.5         788.2
>
```

To get all observations, we *use us_GDPannual(0)*, see the codes below. In the next section, it will be shown how to save all of those data to an external file which could be retrieved into Excel for further analysis.

```
> x=show_usGDPannual(0)
> dim(x)
[1] 87  3
```

The *dim()* function show the dimensions of an input data set, the number of rows and the number of columns. Here is another example of downloading the historical daily price data for IBM. All data is saved to x. The *head()* and *tail()* functions are used to view the first several lines and last several lines.

```
> x=getYahooDaily("ibm")
> head(x)
        Date   Open   High    Low  Close  Volume Adj.Close
1 2016-12-29 166.02 166.99 166.00 166.60 1594000    166.60
2 2016-12-28 167.29 167.74 166.00 166.19 1721800    166.19
3 2016-12-27 166.98 167.98 166.85 167.14 1396000    167.14
4 2016-12-23 167.00 167.49 166.45 166.71 1699200    166.71
5 2016-12-22 167.36 168.23 166.58 167.06 2778900    167.06
6 2016-12-21 166.25 167.94 165.25 167.33 3568400    167.33
> tail(x,2)
            Date  Open  High Low Close Volume Adj.Close
13844 1962-01-03 572.0 577.0 572   577 288000  2.280246
13845 1962-01-02 578.5 578.5 572   572 387200  2.260487
>
```

The second input value of the *head()* and *tail()* function indicates how many lines. The default value is 6. Thus, *tail(x,2)* would show the last two observations. The third example is to download the latest several years balance sheet for IBM.

```
> x=getBSannual("ibm")
Annual Balance Sheet for ibm
> head(x)
                                2015-12-31 2014-12-31 2013-12-31 2012-12-31
Cash & Equivalents                    7686       8476      10716      10412
Short Term Investments                 711        656        573        726
Cash and Short Term Investments       8397       9132      11289      11138
Accounts Receivable - Trade, Net      8333       9090      10465      10667
Receivables - Other                     NA         NA         NA         NA
Total Receivables, Net                28554      31831      31836      30578
> tail(x)
                                 2015-12-31 2014-12-31 2013-12-31 2012-12-31
Treasury Stock - Common          -155518.00 -150715.00 -137242.00 -123131.00
Other Equity, Total               -29611.00  -27861.00  -21601.00  -25764.00
Total Equity                      14262.00   11868.00   22792.00   18860.00
Total Liabilities & Shareholders&#39;Equity    110495.00  117271.00  126223.00
119213.00
Shares Outs - Common Stock Primary Issue    NA         NA         NA         NA
Total Common Shares Outstanding    965.73     990.52    1054.39    1117.37
```

To make downloading data easier, another function called *loadYan()* is generated. After tying *loadYan*, we would see the following list of available data sets.

```
> loadYan
function(i){
"Objective: load an R data set from Yan's web page
  i    description
 ----  ------
  1    ffMonthly
  2    ffDaily
  3    ffMonthly5
  4    retDIBM
  5    usGDPannual
  6    is50
  7    retD50
  8    crspInfo
  9    stockMonthly
 10    indexMonthly
 11    indexDaily
 12    tradingDaysDaily
 13    tradingDaysMonthly

 Example #1>loadYan(1)
            The data set called  ffMonthly is successfully loaded.
            Use header() and tail() functions to explore.
```

```
";loadYan_(i)
}
>
```

Getting data is easy by using *loadYan* function. For example, typing *loadYan(1)* would load the monthly Fama-French factors with a data set called *ffMonthly*, while typing *loadYan(12)* would load the number of days for daily frequency with a data set called *tradingDaysDaily*. [14]

6. Save our data for Excel

Obviously, we need an efficient way to save our downloaded data shown in the above section to an external file. After that, students could use Excel to input and process those downloaded data. For those purposes, two functions called *saveYan()* and *saveFinStatement()* are available.[15]

```
> saveYan
function(x,outfile="test.csv"){
"Objective: save your data as a csv file
    data  : your data
  output  : name of your csv file (default is test.csv)

 Example 1: > x<-show_usGDPannual(0)
            > saveYan(x,'c:/temp/usGDP.csv')
              [1] 'Your saved file is ==>c:/temp/usGDP.csv'

 Example 2: > x<-getDailyPrice('ibm')
            > saveYan(x,'c:/temp/ibm.csv')
              [1] 'Your saved file is ==>c:/temp/ibm.csv'

 Example 3: > x<-getDailyPrice('ibm')
            > saveYan(x)
              [1] 'Your saved file  is
                     ==>C:/Users/yany/Documents/test.csv'
              Note: your data will be saved with a
                  file name of test.csv under your
                   current working directory. To find your
                   current working directory, issue
              >getwd()

";saveYan_(x,outfile)
}
```

[14] When teaching at University at Buffalo, we offer two functions called *loadCRSP* and *loadComp*. For example, we have generated over 100 dataset based on CRSP (Center for Research in Security Prices) databases.
[15] Note that those two functions are contained under submenu called *allDataFunction*.

Based on the above explanation, the usage of the function is very clear. To save space, we could not offer more examples here. When saving a dataset, we have to know the destination (directory) and the file name. The *getwd()* function would show our current working directory. Similarly, typing *saveFinStatement* would offer us the following explanation.

```
> saveFinStatement
function(x,outfile='test.csv'){
"Objective: save a financila statement to a csv file
        x : input data set
   outfile: outfile, default value is test.csv

 Example 1: > x=getBSannual('ibm')
            > saveFinStatement(x)
            [1]'Your saved file is ==>C:/Users/yany/Documents/test.csv'
            Note: location depends on you default working directory
            To know your working directory, issue the following command
            >getwd()

";saveFinStatement_(x,outfile)
}
>
```

One of the major properties of those two saving functions is that the exact location would be printed on screen after each execution to remind the user.

7. Mimic Excel

Since Excel is the computational tool for this course, we have to focus on Excel functions, especially those related to finance, such as pv(), fv(), pmt() and the like. However, there are some issues with the design of those functions in Excel. First, the sign convention is really confusing: enter a positive future value will lead to a negative present value, vice versa. If we use those Excel functions just once, it could be manageable. However, if we use those values resulted from applying those Excel functions for our next steps, students are prone to errors. The submenu called *mimicExcel* is designed to help students understand those functions.

```
> mimicExcel
function(){
" *-------------------------------------------------------------*
  *   Mimic Excel                                               *
  *-------------------------------------------------------------*
  *   Functions         Utilities                               *
  *-------------------------------------------------------------*
  *   pvExcel           pvExcelNoSignConvention                 *
```

```
  *   fvExcel           fvExcelNoSignConvention                  *
  *   pmtExcel          pmtExcelNoSignConvention                 *
  *   nperExcel         nperExcelNoSignConvention                *
  *   rateExcel         rateExcelNoSignConvention                *
  *   npvExcel          npv_f                                    *
  *   effectExcel       effectYan                                *
  *   days360Excel                                               *
  *   priceExcel        shortCut                                 *
  *   yieldExcel        severalRcommands                         *
  *-----------------------------------------------------------*
  *   >pvExcel          # find out the usage of pvExcel          *
  *   >mimicExcel       # back to this menu                      *
  *-----------------------------------------------------------*

";mimicExcel_()
}
>
```

The first 5 pairs of functions are function with and without the Excel sign convention. Using the first function of *pvExcel()* as an example. Typing *pvExcel* would lead to the following output.

```
> pvExcel
function(rate,nper,pmt,fv=0,type=0){
  "Objective: mimic Excel function of =pv(rate,nper,pmt,[fv],[type])
        rate : effective period rate
        nper : number of periods
        pmt  : payment per period
        fv   : future value
        type : 0 payments at the ends (default), 1 at the beginnings

    Example 1: > pvExcel(0.1,1,0,100)
                 since fv>0, then pv <0
                 [1] -90.90909
    Example 2: > pvExcel(0.1,1,0,-100)
                 Since fv<0, then pv>0
                 [1] 90.90909
    Example 3: annuity
               > pvExcel(0.1,3,10)
                 Since pmt>0, then pv <0
                 [1] -24.86852
    Example 4: > pvExcel(0.1,3,0,100,1)
                 Cash flows at the beginings of periods (due).
                 Since fv>0, then pv <0
                 [1] -75.13148
              >  pvExcel(0.1,3,0,100,0)
                 Since fv>0, then pv <0
```

```
                  [1] -75.13148
 ";pvExcel_(rate,nper,pmt,fv,type)
}
>
```

From the above instruction, we could see a short explanation why we end up with a positive or negative value.

8. A new financial calculator

In the above section, it is shown that we could mimic Excel. Thus, students would understand Excel functions better. On the other hand, students are offered new financial calculator. After typing *fincal*, we could see the following menu.

```
> fincal
function(){
"
 *-------------------------------------------*
 * Financial calculator                      *
 *-------------------------------------------*
 *  Functions            Functions            *
 *-------------------------------------------*
 *  pv_f                 bondPrice            *
 *  pvPeretuity          npv_f                *
 *  pvPerpetuityDue      npvExcel             *
 *  pvAnnuity            nPeriod              *
 *  pvAnnuityDue         EAR_f                *
 *  pvGrowAnnuity        Rc_f                 *
 *  fv_f                 Rcontinuous          *
 *  fvAnnuity            rateYan              *
 *  fpvAnnuityDue        duration_f           *
 *  pvGrowAnnuity        modifiedDuration     *
 *  bond_price                                *
 *-------------------------------------------*
 *  >pv_f                # see its usage      *
 *  >fincal              # back to this menu  *
 *-------------------------------------------*
"
}
>
```

For most functions, we could find their equivalents from Excel. There are several advantages of those newly generated functions compared those in Excel. First, the objective of each function is clear stated. Second, the meaning of each input variable is presented. Third, the formula used is

also shown. More importantly, the formula is the same as it shown on our textbooks! In other words, there is not Excel sign convention. Last but not least, we could see a few examples. This suggests that students could learn those functions by themselves. After typing *pvAnnuityDue*, the following introduction would be shown.

```
> pvAnnuityDue
function(c,r,n) {
"Objective: estimate present value of an annuity due
            Note: the 1st cash flow is at time zero
                                c               1
    formula: PV(annuity due)= ---- * (1  -   -------) *(1+r)
                                r             (1+r)^n
       c   : cash flow
       r   : effective period rate
       n   : number of periods

 Example 1:> pvAnnuityDue(10,0.08,20)
               [1] 106.036

"; pvAnnuityDue_(c,r,n)
}
>
```

Actually, this financial calculator has more functions than Excel[16]. For example, there is no Excel function for calculating the present value of growing annuity even its formula appears on most finance textbooks. Thus, it should be viewed as one of the most frequently used financial formula. After typing *pvGrowingAnnuity*, we would see the following instruction.

```
> pvGrowingAnnuity
function(c,r,n,g=0) {
"Objective: estimate the present value of a growing annuity
                                       c           (1+g)^n
   formula: pv(growing annuity)= --------* [ 1 - ----------]
                                    R-g            (1+R)^n
   Inputs
      c   : cash flow happens at the end of 1st period
      r   : effective period rate
      g   : growth rate of the cash flow
      n   : number of periods

  Example 1:> pvGrowingAnnuity(100,0.1,2)
```

[16] We could expand the number of functions quite easily.

```
              [1] 231
   Example 2:> pvGrowingAnnuity(100,0.1,2,0)
              [1] 231
  Example 3:> pvGrowingAnnuity(100,0.1,2,0.03)
               [1] 234.3
"; pvGrowingAnnuity_(c,r,n,g)
}>
```

Conversion between different effective interest rates is a difficult and confusing topic for most finance students to comprehend. Even after one or two finance courses, many students still have a difficult time to understand its logic. To make things worse, there is no good Excel function to convert them except for one function called *effect()* to calculate EAR (Effective annual rate). To help students, this financial calculator offers a function called *rateYan*, see below.

```
> rateYan
function(APR,method){
  "Objective   : from one APR to another APR and effective rate
   Two inputs  :
           APR : value of the given Annual Percentage Rate
        method : Converting method, e.g., 's2a', 's2q', 's2c'
                 's2a' means from semi-annual to annual
                    a for annual
                    s for semi-annual
                    q for quarterly
                    m for monthly
                    d for daily
                    c for continuously

     Example 1: Convert APR compounded semiannually to APR compounded annually
                > rateYan(0.1,'s2a')
                                      Two rates
                effective annual rate    0.1025
                APR                      0.1025

     Example 2: Convert APR compounded semi-annually to APR quarterly
               > rateYan(0.1,'s2q')
                                       Two rates
               effective quarterly rate  0.02469508
                APR                      0.09878031
     ";rateYan_(APR,method)
}
>
```

From the above explanation, we know that the function has two inputs: *APR* and a *method* such as ‘s2a’. The conversion method of ‘s2a’ indicates that we intend to convert an APR

compounded semiannually to an rate compounded annually, while ‘c2q’ means a continuously compounded rate to a quarterly compounded rate.

9. How to teach macro in one week

We have taught Financial Modeling at several universities more than a dozen times. There is no doubt in our mind that Excel macros and VBA is a good tool. Even so, teaching macro to undergraduate or graduate students is not a good choice. We have some bad experience of doing so. There are several reasons. First, it is quite time consuming. Second, most finance-major students are “geared” to programming. Third, Excel macro or VBA could not be compared with R or other opens-source languages. In other words, if students learn R instead of VBA, they would get more by spending less time and efforts. In addition, both R and Python, both are free software, are more powerful compared with Excel macros or VBA.

For our Financial Modeling courses, a new approach was adopted: just one week (two lectures) on macro/VBA. For the first lecture, students learn how to use “recording” function to record their operations. This is related to chapter 26: Simple macro. After typing e26, we could have the following list.

```
explain26<-function(i){
" i  Description (chapter 26: Simple Macro)
  -  ------------------------------------
  1  If Developer is not available from your menu bar
  2  Simple Macro
  3  Simple marco 2
  4  Using Ctrl-a to run a macro
  5  save as an Excel Macro-Enable Workbook (*.xlsm)
  6  Macro
  7  show No in a cell
  8  Design a slot machine
  9  Macro: Hello
 10  Record Macro
 11  use the recorded the macro
 12  Ctrl-A, Ctrl-B, Ctrl-D

 Example #1:>e26    # see the above list
 Example #2:>e26(1) # see the 1st explanation

";explain26_(i)
}
```

Here is one example: typing *e26(12)* would lead to the following output.

```
> e26(12)

Ctrl-a, Ctrl-b, Ctrl-c, and Ctrl-d
  Step 1: click "Developer" on your menu bar
  Step 2: click "Record Macro"
  Step 3: for Short-cut enter
          a
          then click "OK"
  Step 4: move your curser to a cell and type anything
  Step 5: Click "Stop Recording"

     Now, you can try it by hitting Ctrl-a
     Exercise: when hitting Ctrl-d, clean your spreadsheet
>
```

For the second lecture, students are taught a "copy-and-paste strategy, see Appendix E for the contents of the second lecture (Chapter 26). The basic idea is to run others' VBA. In class, instructors show two or three examples. After doing 20 or 30 VBA written by others, students would learn a lot within just one or two hours.

10. Other utilities

There are other utilizes or submenu or functions an instructor could use, such as *showFormula*, *datacases*, *userfullLinks* and *funs*. For teach finance course, about several dozen formulae would be taught. For our teaching practice, students would get a formula sheet for their mid-terms. However, they would depend on their own formula sheet for final. Via *showForula* function, 40 formulae are at students' finger tips.

```
> showFormula
function(i){
"Objective: show formula
   i      : an integer

   i Description          i  Description
  -- ------------        --  -----------
  1  fv                  21  portfolioVar
  2  pv                  22  2stockPortfolio
  3  pvPerpetuity        23  ff3factorModel
  4  pvGrowingPerpetuity 24  geometricMean3numbers
  5  pvAnnuity           25  distributionStandardNormal
  6  pvAannuityDue       26  distributionNormal
  7  pvGrowingAnnuity    27  cumulativeStandardNormal
  8  fvAannuity          28  bondPrice
  9  fvAnnuityDue)       29  APR2EAR
```

```
 10  fvGrowingAnnuity     30  APR2Rc
 11  adjustedBeta         31  freeCashFlow
 12  CAPM                 32  APR1toAPR2
 13  annalizedVarStd      33  totalReturn2compoents
 14  sharpeRatio          34  nominalRateVSrealRate
 15  treynorRatio         35  liquidityRatios
 16  callAndPut           36  profitability
 17  variance             37  financialLeverage
 18  covariance           38  ROAandROE
 19  portfolioReturn      39  duPont
 20  portfolioBeta        40  percentageReturnLogReturn

 Example #1>showFormula

 Example #2>showFormula(1)

";showFormula_(i)
}
>
```

In total, we could show 40 different formulae, see a few examples below.

`> showFormula(1)`
`Formula is  fv.png`
`>`
	$FV = PV * (1 + R)^n$
	FV: future value
	PV: present value
	R : period rate
	n : number of periods

`> showFormula(18)`
`Formula is  covariance.png`
`>`
	$\sigma_{A,B} = \frac{\sum_{i=1}^{n}(R_{A,i} - \bar{R}_A)(R_{B,i} - \bar{R}_B)}{n-1}$
	$\sigma_{A,B}$ is the covariance between A and B.
	n is the number of observations
	$R_{A,i}(R_{B,i})$ is the i^{th} return for stock A (B)
	$\bar{R}_A(\bar{R}_B)$ is the mean return for stock A (B)

`> showFormula(25)` `Formula is distributionStandardNormal.png`	Density function for a standard normal distribution $f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$

Generating such a formula image is trivial: typing a formula by using Microsoft Word. Then use snipping tool to generate a picture with an extension of .png. We plan to double the number of formulae. Nowadays, our students have a quite different habits or attitudes toward learning. If instructors could combine funs with learning, it could be more effective. The utility called *funs* is designed with this in mind. Currently, the *funs* submenu includes: *randomCall*, *zipcode*, *journalRankings*, *ggf,* and *adv*. The *randomCall* would randomly call a student's name and show his/her picture on the screen, see Appendix G. Another one is called zipcode, see Appendix H. Based on a zip code, we could find the average house price, population, race composition. Here is an exercise really for fun: finding one C among 500 O's or finding one 6 among many 500 9's, see Appendix I for an illustration.

11. Conclusions and potential extensions

This paper introduces a new method to teach "Financial Modeling using Excel". This new method has a few features. First, during each lecture students would do at least two hands-on exercises. Second, students are offered many explanations and formulae. Thus, they could learn Excel and apply it to finance by themselves. Third, students are offered an efficient way to download data such as US GDP, US national debt, unemployment rate, CPI, Fama-French factors, risk-free rate, historical stock prices, latest three years financial statements. For example, if a student wants to download 50 stocks' monthly prices plus their 3 years financial statements, it should take less than 10 minutes. Fourth, all supporting materials are written with a text format. Thus, any instructor could modify those text files according to his/her needs. The approach is labeled as an "R-assisted learning environment. Even so, students and instructors are not required to learn R. Students are offered just a half-page instruction on how to download and install R. Literately students are responsible for one-line R codes. Even for this one line R codes, they could just copy and paste!

There are several potential extensions. First, we could have an interactive mode to help our students learn finance and Excel. This is called "interactiveLearning". The second direction is visual finance. There is no doubt in our mind that visual presentations of various finance concepts and trading strategies would help our students understand them better. Another obvious extension is to develop the same teaching materials for Portfolio Theory, Options and Futures and Corporate Finance. Hopefully, this paper has started a new era for teaching and learning various finance courses.

References

Benninga, Simon, Financial Modeling, 4th edition, MIT Press, 2014.

Cagle, J. A. B., P. W. Glasgo and D. C. Hyland, 2010, Spreadsheet: Do They Improve Student Learning in the Introductory Finance Course? Journal of Financial Education, 35-52.

Carini, R. M., G. D. Kuh, and S. P. Klein, 2006, Student Engagement and Student Learning: Testing the Linkages, Research in Higher Education 47, 1-32.

Kane, David, 2006, Open Source Finance, working paper, Harvard University, SSRN link is at http://papers.ssrn.com/sol3/papers.cfm?abstract_id=966354

Yan, Yuxing, 2016, Mimic Excel, Journal of Economics and Finance Education 15, 1, 96-100.

Yan, Yuxing, An internet connected financial calculator, 2012, Journal of Accounting and Finance 12(5), 59-70.

Yan, Yuxing and Mark Mark Zaporowski, Easy Access to Data for Classroom Use, working paper, Canisius College.

Zhang, Chengping, 2014, Incorporating Powerful Excel Tools Into Finance Teaching, Journal of Financial Education, 87-113.

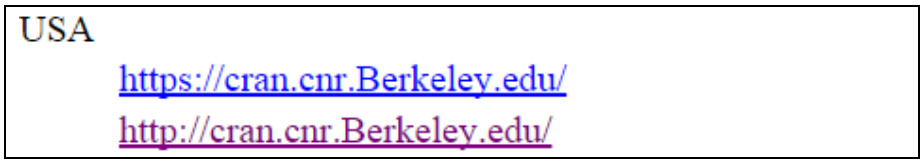
Appendix A: one page instruction distributed to students

To download and install R (free computational software), we have the following 5 steps.

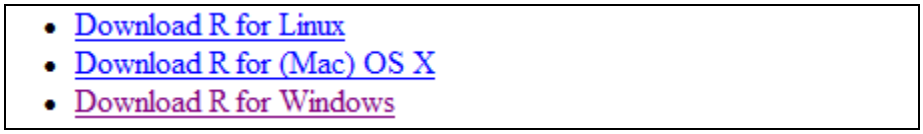
Step 1: Go to http://www.r-project.org

Step 2: Click "CRAN" under "Download" (left-hand side)

Step 3: Choose a mirror address



Step 4: Choose appropriate software (PC, Mac)



Step 5: Click "base". For example, for Windows, we have the following result.

After launch R, just issue the following one line R codes.

```
>source("http://canisius.edu/~yany/excel.R")
```

Note that R is case-sensitive.

Appendix B: A list of in class exercises

```
inClassEx<-function(i){
" i   Description
  -   ------------------------------------
  1   =A1 What does it mean?
  2   showFormula function
  3   When developer is not available
  4   type =is    and if() function etc.
  5   conditional formatting, vlookup()
  6   percentage vs. log returns, solver()
  7   usage of logRet, efficient frontier
  8   VBA for variance-covariance matrix
  9   date related functions
 10   All functions related to Date & time
 11   Ctrl-a,Ctrl-b, Ctrl-d (record your macros)
 12   A true NPV function: NPVyan
 13   Auto fill
 14   Excel matrix manipulations
 15   find out all functions
 16   generate random numbers from a uniform distribution
 17   generate random numbers from a normal distribution
 18   generate ransom numbers with seed
 19   Excel rate() vs. yield() function
 20   Ctrl-End, ctrl-home, ctrl-A
 21   All combinations with Ctrl
 22   Generate a Drop-down list
 23   Return the 'left column' using Vlookup() function
 24   String manipulation
 25   named range and named cell
 26   logic: IF, AND, OR, NOT
 27   simple graph (x vs. y)
 28   pv of growing annuity without formula
 29   VBA for present value of a growing annuity
 30   enter T1, T2, to T500000 (fill -> Series ->...)
 31   VBA   =todayPrice('ibm')
 32   index() function
 33   index() and match() functions
 34   Excel brute-force for PV(perpetuity) and PV(growing perpetuity)
 35   Randomly choose a cell from a range
 36   which party indicator
 37   Vlookup() range lookup function
 38   tests of equal means and equal variances
 39   explanation of Black-Scholes-Merton call option model
 40   pricing a call option
 41   Generate random numbers
 42   Using rand() function to estimate pi
 43   Benford law
 44   Simple macro
 45   Simple macro (2)
 46   hit Ctrl-a to call a macro
 47   Build a slot machine
 48   normal distribution
 49   Simulate stock price from t to t+1
 50   Simulate terminal stock price sT
 51   pricing a call (put) by using simulation
 52   Asian options
 53   Powerball Simulation
 54   VaR (assume a normal distribution)

 Example 1:>ice        # show all exercises
 Example 2:>ice(1)     # see the first one

";inClassEx_(i)
}
```

Appendix C: Explanation for Chapter 2: e2

```
> e2
function(i){
" i  Chapter 2:Time value of money/investment decision rules
  -  -----------------------------------
  1  a time line
  2  present value formula for one future cash flow
  3  present value formula for perpetuity
  4  present value formula for annuity
  5  Excel pv() function
  6  future value formulae
  7  Excel fv() function
  8  Excel sign convention
  9  perpetuity due and annuity due
 10  pv, fv, pmt, rate, nper
 11  Excel pmt() function
 12  Excel rate() function
 13  NPV (Net Present Value) definition
 14  NPV rule
 15  Excel npv() is actually a pv function
 16  a true NPV function
 17  IRR (Internal Rate of Return) definition and IRR rule
 18  IRR() functions
 19  payback period and related rule
 20  total risk (stdev.p vs. stdev.s)

 Example #1:> e2    # get the above list
 Example #2:> e2(1) # see the first explanation

";explain2_(i)
}
>
```

Appendix D: explanation 3 (e3) for Chapter 3

```
> e3
function(i){
" i  Chapter 3: R basics
  -  -----------------------------------
  1  From where to download R
  2  How to launch and quit R
  3  R is case sensitive
  4  How to use a financial calculator written in R
  5  3 ways to asign a value to a variable
  6  use getYahooDaily() to get historical price data from Yahoo!Finance
  7  use getBSannual() to get several years' annual Balance Sheets
  8  use getISannual() to get several years' annual Income Statements
  9  use getCFannual() to get several years' annual Cash flow Statements
 10  use getBSquarterly() to get several quarters' Balance Sheets
 11  use getISquarterly() to get several quarters' Income Statements
 12  use getCFquarterly() to get several quarters' Cash Flow statements
 13  use loadYan() to upload remote data sets
 14  current working directory
 15  use saveFin() to save financial statement
 16  use saveYan() to save data, such as to a csv file
 17  Some usfule R functions
 18  use nLetterFunction() to show all n-letter functions
 19  how to find more information about one function
 20  learn more about applying R to finance

 Example #1:> e3    # see the above list
 Example #2:> e3(1) # see the first explanation
";explain3_(i)
}
```

Appendix E: e27 for Chapter 27

```
> e27
function(i){
" i  Chapter 27: Copy-and-paste VBA
  -  ------------------------------------
  1  If developer is not available
  2  Copy-and-paste of others' VBA
  3  Save as an Excel Macro-Enable Workbook (*.xlsm)
  4  VBA for doubling any input value
  5  VBA for area estimation
  6  VBA for showFormula() function
  7  VBA for converting miles to km
  8  VBA for pvGrowingAnnuity
  9  VBA for fvGrowingAnnuity
 10  VBA for geometric mean (for returns) geomeanYan
 11  VBA for effective rate conversion: EFFECTyan()
 12  VBA for a true NPV function: NPYyan()
 13  VBA for generating a randomly choose a cell
 14  VBA for Variance-covariance matrix
 15  VBA for present value of a growing perpetuity
 16  VBA for Black-Scholes call option model  (no dividend)
 17  VBA for Black-Scholes put  option model  (no dividend)
 18  VBA for Black model for pricing European call options on futures
 19  VBA for converting 23.23 to Twenty Three Dollars and Twenty Three
Cents
 20  VBA for todayPrice() from Yahoo!Finance

 Example #1:>e27     # find out the above list
 Example #2:>e27(1)  # see the first explanation

";explain27_(i)
}
```

Appendix F: usefulLinks

```
>usefullLinks
usefulLinks<-function(i){
" i  Useful links
  -  -----------------------------------
  1: R installation/Excel basics(I)
  2: Time value of money            17: Excel basics (II)
  3: R basics                       18: Widely used functions
  4: Source of open data            19: vlookup, solver
  5: Financial statement analysis   20: Data input
  6: Risk vs. return                21: Data manipulation
  7: Interest rate                  22: Data output
  8: Bond evaluation                23: Simple graph
  9: Stock evaluation               24: Matrix manipulation
 10: CAPM                           25: Macro(1) record micros
 11: Fama-French 3 factor, Sharpe   26: Macro(2) copy-paste
 12: Statistical tests              27: Pivotal table
 13  Black-Scholes options model
 14: Monte Carlo Simulation
 15: Portfolio Theory
 16: VaR (Value at Risk)

 Example #1:>ul             # see the above list
 Example #2:>ul(1)          # see the links for Chapter 1

";usefulLinks_(i)
}
```

For example, tying *ul(7)* would lead to the following output.

```
> ul(7)
7: Interest rate
//////////////////////////
 Definition
     http://www.investopedia.com/terms/i/interestrate.asp
     http://www.investorwords.com/2539/interest_rate.html
     http://www.businessdictionary.com/definition/interest-rate.html
     http://dictionary.cambridge.org/us/dictionary/english/interest-rate

 Risk-free rate
     http://www.investopedia.com/terms/r/risk-freerate.asp
     http://www.investopedia.com/ask/answers/09/risk-free-rate.asp
     http://www.investinganswers.com/financial-dictionary/investing/risk-free-
rate-return-5133
     http://www.businessdictionary.com/definition/risk-free-rate-of-return.html

 Term structure of interest rate
     http://www.investopedia.com/terms/t/termstructure.asp

http://spears.okstate.edu/home/munasib/MoneyBanking_3313/Handout/Handout03.pdf
     http://www.investopedia.com/university/advancedbond/advancedbond4.asp

 Sources for data
     https://fred.stlouisfed.org/categories/22
     https://www.treasury.gov/resource-center/data-chart-center/interest-
rates/Pages/default.aspx
     https://www.treasury.gov/resource-center/data-chart-center/interest-
rates/Pages/Historic-LongTerm-Rate-Data-Visualization.aspx
     http://finance.yahoo.com/bonds
     http://www.tradingeconomics.com/china/interest-rate
>
```

Appendix G: *randomCall* menu

```
 *-------------------------------------------------*
 *-------------------------------------------------*
 * Randomly call a student 9/15                    *
 *-------------------------------------------------*
 * Function                                        *
 *-------------------------------------------------*
 * randomCall       # randomly call a student      *
 * randomNum        # randomly generate an integer *
 * showRemainder    # devided by n (default is 30) *
 * showStudentPhoto #                              *
 * showWebPNG       # show a given png file online *
 *-------------------------------------------------*
 * >randomCall     # show its usage                *
 * >rc             # come back to this menu        *
 *-------------------------------------------------*
 *-------------------------------------------------*
>
```

Appendix H: fun based on zip code

```
 *-------------------------------------------------- *
 *  zip code fun          8/24/2014                  *
 *-------------------------------------------------- *
 * Functions             Utilities                   *
 *-------------------------------------------------- *
 * zipPopulation         severalZipCodes             *
 * zipIncome             uniqueZipValues             *
 * zipHousePrice         validZipCode                *
 * zipCity                                           *
 * zipState                                          *
 * zipTimezone           severalRcommands            *
 * zipAreaCode                                       *
 * zipRace                                           *
 * zipCoolingCostIndex                               *
 * zipHeatingCostIndex                               *
 * zipLatitude                                       *
 *-------------------------------------------------- *
 * >zipIncome            # find its usage            *
 * >zipcode              # back to this menu         *
 *-------------------------------------------------- *
```

Appendix I: finding one C in 500 O's

```
*-----------------------------------------------------------------*
*  fun using R                                                    *
*-----------------------------------------------------------------*
*  Functions          Utilities                                   *
*-----------------------------------------------------------------*
*  where_is_C         done                                        *
*  where_is_6         solution                                    *
*  where_is_N         # note R is case sensitive                  *
*-----------------------------------------------------------------*
*  >where_is_C        # see the usage of this function            *
*  >cVSO              # back to the  main menu                    *
*-----------------------------------------------------------------*
```

One of the trials is shown below.

```
  [1] "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O"
 [19] "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O"
 [37] "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O"
 [55] "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O"
 [73] "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O"
 [91] "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O"
[109] "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O"
[127] "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O"
[145] "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O"
[163] "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O"
[181] "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O"
[199] "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O"
[217] "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O"
[235] "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O"
[253] "O" "O" "O" "O" "O" "O" "O" "O" "O" "C" "O" "O" "O" "O" "O" "O" "O" "O" "O"
[271] "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O"
[289] "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O"
[307] "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O"
[325] "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O"
[343] "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O"
[361] "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O"
[379] "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O"
[397] "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O"
[415] "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O"
[433] "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O"
[451] "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O"
[469] "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O"
[487] "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O" "O"
> 
```