Learning R and Python for Business School Students

Learning R and Python for Business School Students

By

Yuxing Yan

Cambridge Scholars Publishing

Learning R and Python for Business School Students

By Yuxing Yan

This book first published 2023

Cambridge Scholars Publishing

Lady Stephenson Library, Newcastle upon Tyne, NE6 2PA, UK

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

ISBN (10): 1-5275-9018-6
ISBN (13): 978-1-5275-9018-2

To: Rene De La Pedraja and Lee-Hsien Pan

(Two of my wonderful colleagues)

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

Since this book is based on my lecture notes for Programming for Data Analytics, I would like to thank Christian Shin (for recommending me to teach this course after he received my three books: Python for Finance, Financial Modeling using R, and Hands-on Data Science with Anaconda (with James Yan)), Mary Ellen Zuckerman and Delbert Brown for assigning me to teach it.

I would like to thank Ben Amoako-Adu, Brian Smith (who taught me the first two finance courses and offered unwavering support for many years after my graduation), George Athanassakos (one of his assignments "forced" me to learn C), and Jin-Chun Duan. I would also like to thank Wei-Hung Mao, Jerome Detemple, Bill Sealey, Chris Jacobs, Mo Chaudhury, Summon Mazumdar (my former professors at McGill) and Lawrence Kryzanowski (his wonderful teaching inspired me to concentrate on empirical finance and he edited my doctoral thesis word by word, even though he was not my supervisor!); Kee Chung, Cristian Tiu, Sudhir Suchak, Yoon Shin, Lisa Fairchild, Karyl Leggio, Shuo Chen, Lei Gao, Ken Pan, Rafiqul Bhuyan, James Early, Qian Sun, Shaojun Zhang, Paul Sauer, Edward J. Garrity, James Goldstein, Mark P. Zaporowski, Jinsook Lee, K.G. Viswanathan, Liang Hsu, Vadim S. Balashov, Xiaodi Zhu, and Christopher Armstrong. In addition, I'd like to thank Anthony Gu, Shaobo Ji, Tong Yu, Shaoming Huang, and Xing Zhang. I thank Eric Ulm, Premal Vora, Ronald Crowe, Dehong Wang, Andreas Chouliaras, Andy Kim, Sheng Xiao, Cong Xie, Mesut Ozdag, Lulu Zeng, Daniel Folkins, and James Nordlund for adopting my R or Python books as their textbooks.

There is no doubt that my experience at Wharton has shaped my thinking and enhanced my skill sets. I thank Chris Schull and Michael Boldin for offering me the job; Mark Keintz, Dong Xu, Steven Crispi, and Dave Robinson, my former colleagues, who helped me greatly during my first two years at Wharton; and Eric Zhu, Paul Ratnaraj, Premal Vora, Shuguang Zhang, Michelle Duan, Nicholle Mcniece, Russ Ney, Robin Nussbaum-Gold, and Mireia Gine for all their help.

In particular, I would like to thank my wife for her strong support and James Yan for proofreading the whole manuscript and checking the code.

PREFACE

It is crazy to teach business students both R and Python at the same time! It was my initial thought when asked to teach such a course in 2020. However, after struggling heavily together with my students (over three semesters), I found that such a course is not only possible, but also desirable. In a recent paper, I summarized my related teaching experiences into several areas: 1) write my own lecture notes; 2) use concepts and formulae suitable for business students; 3) tons of in-class exercises; 4) use the same formulae, data sets or even the same exercises several times; 5) at least one video for each chapter; 6) over 1,000 small programs; 7) many utility functions, and 8) spend more time on R, then compare Python with R constantly.

This book is the result of my two years' effort. At Geneseo (SUNY), this book is used by close to 200 students over 3 semesters in 2021 and 2022. Since this course is open to all students at Geneseo, 80% of my students come from the School of Business, and rest from other schools. Many students are majoring Accounting, Marketing or Business Administration. During the first lecture, I always give students a survey asking for their major, which year, any knowledge related to coding, level of Excel and career plan (the last being optional). Since the majority of students know nothing related to programming, this book is designed to serve them.

Beyond just one textbook

One of the main features of this book is that it is not just a book or a textbook. There are many accompanying materials, such as over 1,000 programs written in R and Python, 200 data sets in CSV, RData, sas7bdat, and pickle (Python data sets) formats. On the other hand, R is used to help readers and students learn, along with many utility functions. Among a dozen utility functions, a function called `.searchWebs()` could be used to search various web pages and data sets. This function makes addressing some data sets efficient with, just a few lines of R or Python code. We will elaborate those properties in the next sections or pages.

Starting from scratch

From this book, readers learn basic concepts and programming skills related to R, and Python, then apply them to finance, accounting, marketing, and other business areas. This is a beginner's book. In other words, we do not assume that readers have any prior knowledge or skills for R or Python. On the other hand, if readers have very basic business knowledge, such as understanding the present value formula of a given future value, present value of an annuity and the like, it will be great. Since the formulae used in this book are not that complex, a first-year business major student could learn them easily.

R first, then Python

A reader or student could learn both R and Python within one semester if the book is used as a textbook with a good instructor. On the other hand, if a learner learns both languages by himself/herself, it is a good idea to take two semesters, i.e., at a slow pace. There are three parts of the book: R, Python, and applications of R and Python. If this book is adopted as a textbook, the first two parts are more than enough for one semester. The first 13 chapters are centered on R. Starting from Chapter 14: Python Basics, we will focus on Python. One important feature of the second half of the book is that we constantly compare Python with R. Such a comparison serves two purposes. First, learners/students could refresh their mind about R knowledge and related functions. Second, it is much easier to use the same concept with a different language. For example, after students learn how to apply a double loop to download all the SEC quarterly zip files from 1993Q1 to 2021 Q2 using R, it is easy to do the same by using Python.

Using familiar concepts and formulae

Since this book's target audience is business school students, we use many concepts and formulae they have learned, such as the time-value of money, present value formula for a given future value, present value formula for perpetuity, present value of annuity, future value for given present value, future value of annuity, Net Present Value formula and the like. In addition, many formulae are used repeatedly. The underlying reasons will be mentioned below. For example, the following formula is used to calculate the present value of one given future cash flow.

$$PV=FV/(1+R)^n ,$$

where PV is the present value, FV is the future value, R is the discount rate and n is the number of periods.

Repeating the same concepts, formulae, and exercises

One assumption is that readers/students have no prior knowledge related to R and Python. For a beginner, it is a good idea to repeat the same concepts, formulae and even the same exercises. There are several reasons behind such a strategy. First, repeating will reinforce students' understanding of the same concept and procedure. For example, let us use the present value of one given future value as one example. We use it several times: when writing a one-line R program, 2-line Python program, inputting values, and adding a multi-line comment. Another example is the SEC quarterly index files, both zip and text files. After students learnt how to use R to write a double loop to download all the zip files from 1993 Q1 to 2021 Q2, they feel more confident when learning how to use Python to achieve the same goal. For this strategy, we tag it as: "repeat, repeat, and repeat". For example, the formula of $PV=FV/(1+R)^n$ is used at least 6 times, shown below.

1) used for one-line R code.
2) used when talking about three ways to input variables using R
3) used when talking about how to add a multi-line comment in R
4) used for two-lines Python code.
5) used when explaining two ways to input data for Python
6) used when talking about how to add a multi-line comment for Python

One example related to data set is called ff3Monthly, which is the Fama-French monthly data set. Its usages are shown below.

1) The original F-F_Research_Data_Factors_CSV.zip for R
2) ff3monthly.csv used for the read.csv() R function
3) ff3Monthly.RData is used for the R `load()`, `get()`, and `url()` functions
4) ff3Monthly.csv is used for `pandas.read_csv()` in Python
5) ff3Monthly.Pickle is used for `pandas.read_pickle()` in Python
6) The original F-F_Research_Data_Factors_CSV.zip for Python

The `.searchWebs()` function can be used, shown below.

```
> .searchWebs('ff3m')
  ID                                NAME
 412 ff3 monthly (no annual) txt
 413 ff3 monthly raw txt
 416 ff3 monthly raw csv
 417 ff3 monthly (no annual) csv
 638 4/18 ff3 monthly TXT
 639 4/18 ff3 monthly CSV
 501 Fama-French monthly 3 factors
 507 Fama-French monthly 3 factors
 508 Fama-French monthly 3 factors pickle
                                                                                  WEBSITE
 http://datayyy.com/data_txt/F-F_Research_Data_Factors2.txt
 http://datayyy.com/data_txt/F-F_Research_Data_Factors.txt
 http://datayyy.com/data_csv/F-F_Research_Data_Factors.CSV
 http://datayyy.com/data_csv/F-F_Research_Data_Factors2.CSV
 http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/ftp/F-F_Research_Data_Factors_TXT.zip
 http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/ftp/F-F_Research_Data_Factors_CSV.zip
 http://datayyy.com/data_csv/ff3Monthly.csv
 http://datayyy.com/data_R/ff3Monthly.RData
 http://datayyy.com/data_pickle/ff3Monthly.pickle
>
```

Over 1,000 small programs written in R and Python

To help readers and students using this book or taking a related course, we have generated over 1,000 small programs written in R and Python. To help users search, show and download those programs, we have designed three related utility functions: `.searchCode()`, `.showCode()`, and `.downloadCode()`. If we have 1,123 programs, then their `ID` will be from 1 to 1,123. Readers can use the `.searchCode()` function to find a program ID by entering a chapter number or a keyword.

```
> .searchcode(1)
   CODE_ID                                          NAME
1        1 c01_1_assignment.R.txt
2        2 c01_10_scan_function.R.txt
3        3 c01_11_read_table_from_clipboard.R.txt
4        4 c01_12_from_clipboard_with_header.R.txt
5        5 c01_13_some_existing_functions.R.txt
6        6 c01_14_finding_help.R.txt
7        7 c01_15_apropos_function.R.txt
8        8 c01_16_nLetterFunctions.R.txt
9        9 c01_17_precision.R.txt
10      10 c01_18_matrix.R.txt
11      11 c01_19_rename_all_files.R.txt
12      12 c01_2_comments.R.txt
13      13 c01_20_comparison_R_Python.txt
14      14 c01_3_caseSensitive.R.txt
15      15 c01_4_semi_colon.R.txt
16      16 c01_5_using_c.R.txt
17      17 c01_6_from_1_to_50.R.txt
18      18 c01_7_ls_rm_functions.R.txt
19      19 c01_8_print_cat_functions.R.txt
20      20 c01_9_seq_function.R.txt
>
```

In the above image, `.R.txt` is the R programs' extension. Since R is taught first and the above image is for Chapter 1, it is not a surprise to find no Python programs. The abbreviation for the `.searchCode()` function is `.sc()`. In addition to offering a list of programs for each chapter, users could search programs by using a key phrase, shown below.

```
> .searchCode("ff3m")
   CODE_ID                                                    NAME
1       39 c02_26_read.delim_ff3Montly.R.txt
2       90 c04_12_ff3Monthly.R.txt
3      156 c05_71_showff3Monthly.R.txt
4      166 c06_05_read_ff3monthly_RData.R.txt
5      439 c12_14_ff3monthly.R.txt
6      599 c16_24_read_pickle_ff3Monthly_pickle.py.txt
7      618 c16_75_generate_ff3Monthly_pickle.py.txt
8      649 c17_27_generate_ff3Monthly_pickle_from_txt.py.txt
9      651 c17_29_generate_ff3Monthly_pickle_from_csv.py.txt
10     652 c17_30_generate_ff3Monthly_pickle_from_csv.py.txt
11     653 c17_31_generate_ff3Monthly2.py.txt
12     656 c17_34_ff3Monthly_pickle.py.txt
13     659 c17_37_ff3Monthly_pickle.py.txt
14     665 c17_44_show_ff3Monthly_ff3Monthly2.py.txt
15     758 c19_24_ff3Monthly_pickle.py.txt
16     781 c19_49__ff3Monthly_pickle.py.txt
17     897 c21_21_generate_ff3Monthly_from_text_file.py.txt
18     903 c21_27_ff3Monthly_text.py.txt
19     918 c21_42_get_ff3Monhtly_from_raw_data_set.py.txt
20    1006 c23_15_ff3Monthly_text.py.txt
21    1008 c23_16_ff3Monthly_text.py.txt
22    1010 c23_21_gemerate_ff3Monthly_from_text_file.py.txt
23    1016 c23_26_ff3monthly sas7bdat to Python pickle.py.txt
```

The `.showCode()` function will show a program on our screen for a given code ID. This function makes doing in-class exercises more efficiently.

```
> .showCode(652)
# -*- coding: utf-8 -*-
"""
Created on Sat Apr 10 20:30:52 2021

@author: pyan
"""
import pandas as pd
import numpy as np
path="http://datayyy.com/data_csv/"
myfile="ff3Monthly.csv"
infile=path+myfile
df=pd.read_csv(infile,skiprows=3)
df.columns=["DATE","MKT_RF","SMB","HML","RF"]
#
n=np.shape(df)[0]
dd=[]
for i in range(0,n):
    a=df['DATE'][i]*100+1
    dd.append(pd.to_datetime(a, format='%Y%m%d').date())
#
df['DATE']=dd
df.iloc[:,1:]=df.iloc[:,1:]/100
#df.to_pickle("c://temp/ff3Monthly.pkl")

>
```

Obviously, `.downloadCode()` is used to download a program for a given code ID. For example, for one program with an ID of 653, we have the following result.

```
> setwd('c://temp')
> .downloadCode(653)
   # The program's  name is  c17_31_generate_ff3Monthly2.py.txt
   #      located under the directory: c:/temp
>
```

Many in-class exercises

The best way to learn a programming language is via hands-on. For this reason, we have developed close to 100 in-class-exercises. When teaching the course "Programming of Data Analytics", students will do at least 2 of them for each lecture. To access a list of all in-class exercises, students type `.inClassEx` or `.ice`. Note that there are 15 lists for 15 weeks, see the first one shown below.

```
> .ice
function(i){
" i  chap Description
  -  ---- -------------------------------
  1   1   videos to install R
  2   1   Estimate present values
  3   2   Write an R function for the growing annuity
  4   2   Add comments (help)
  5   2   PVIF (Present value interest factor) table

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

Below is one in-class exercise of generating an R data set based on the ABDC (Australian Business Deans Council) journal list.

```
> .ice(72)
Download SEC zip fils from datayyy.com
////////////////////////////////////////////////////////

 Step 1: manuall download one zip file
         http://datayyy.com/sec/

        infile='http://datayyy.com/sec/y1993/QTR1/company.zip'
       > .is404(infile)
         [1] 200

 Step 2: write a Python program to download
        from download import download
        infile='http://datayyy.com/sec/y1993/QTR1/company.zip'
        outfile='index1993q1.zip'
        download(infile,outfile)


 step 3: generate two variable called y and q
         y=1993
         q=2
        from download import download
        infile='http://datayyy.com/sec/y'+y+'/QTR'+q+'/company.zip'
        outfile='index'+y+'q'+q+'.zip'
        download(infile,outfile)

        # note str() function

 Step 4: one loop

 Step 5: double loops
```

Prof. French maintains a data platform where users/researchers can download many useful data sets. The format of those data sets is a zip file. One related in-class exercise is to generate R data sets for all those zip files, shown below.

```
>
> .ice(52)
Generate an R data set for all the zip files for French's Data Library
////////////////////////////
Manually check the website
--------------------------
   1) Go to Fench's Data Library at
        http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html
   2) Right-click  your mouth
         View Page Source

    infile<-'http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/data_library.html'
    x<-readLines(infile)

   hint:  path<-'http://mba.tuck.dartmouth.edu/pages/faculty/ken.french/ftp/'

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

Code Images

When doing an in-class exercise, instructors decide whether to offer students some sample code. By offering a few good sample programs, most students could learn faster. On the other hand, some students adopt a copy-and-paste strategy with those sample code. In the long run, this is not a good strategy to learn programming since hands-on is critical. To solve this dilemma, students could apply a function called `.codeImage`. This function has two features: it searches the ID for a specific program and shows its image on our screen. When searching for a specific program, a

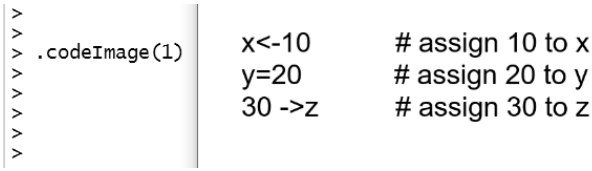
keyword is used as an input. Alternatively, we can show all the code related to a specific chapter. To find all the programs related to the Fama-French 3-factor models, "ff3" can be used as our input string, shown below.

```
>
> .codeImage('ff3')
  ID                                                NAME
 126 c12_14_ff3monthly.R
 286 c16_24_read_pickle_ff3Monthly_pickle.py
 305 c16_75_generate_ff3Monthly_pickle.py
 336 c17_27_generate_ff3Monthly_pickle_from.py
 338 c17_29_generate_ff3Monthly_pickle_from_csv.py
 339 c17_30_generate_ff3Monthly_pickle_from_csv.py
 340 c17_31_generate_ff3Monthly2.py
 343 c17_34_ff3Monthly_pickle.py
 346 c17_37_ff3Monthly_pickle.py
 352 c17_44_show_ff3Monthly_ff3Monthly2.py
 445 c19_24_ff3Monthly_pickle.py
 468 c19_49__ff3Monthly_pickle.py
 494 c2_26_read.delim_ff3Montly.R
 562 c20_56_download_ff3Daily.py
 615 c21_21_generate_ff3Monthly_from_text_file.py
 621 c21_27_ff3Monthly_text.py
 636 c21_42_get_ff3Monhtly_from_raw_data_set.py
 724 c23_15_ff3Monthly_text.py
 726 c23_16_ff3Monthly_text.py
 728 c23_21_gemerate_ff3Monthly_from_text_file.py
 733 c23_26_ff3monthly_sas7bdat_to_Python_pickle.py
 842 c4_12_ff3Monthly.R
 908 c5_71_showff3Monthly.R
 918 c6_05_read_ff3monthly_RData.R
>
```

The output shows two columns: the code ID and the names of those programs. To find all the code related to Chapter 1, we use "c1_" as our input, shown below.

```
>
> .codeImage('c1_')
 ID                                              NAME
  1 c1_1_assignment.R
  2 c1_10_scan_function.R
  3 c1_11_read_table_from_clipboard.R
  4 c1_12_from_clipboard_with_header.R
  5 c1_13_some_existing_functions.R
  6 c1_14_finding_help.R
  7 c1_15_apropos_function.R
  8 c1_16_nLetterFunctions.R
  9 c1_17_precision.R
 10 c1_18_matrix.R
 11 c1_19_rename_all_files.R
 12 c1_2_comments.R
 13 c1_20_comparison_R_Python
 14 c1_3_caseSensitive.R
 15 c1_4_semi_colon.R
 16 c1_5_using_c.R
 17 c1_6_from_1_to_50.R
 18 c1_7_ls_rm_functions.R
 19 c1_8_print_cat_functions.R
 20 c1_9_seq_function.R
>
```

With the above two methods, we can find a related code ID or IDs. For example, the ID for the first entry, `c1_1_assignment.R`, is 1. To show its image, we use 1 as our input value, shown below.



In the above code, `.ci()` is the abbreviation for the `.codeImage()` function. Now, students could type their own code on the R console according to the image shown on the right.

Many good utility functions

Our utility functions make learning R and Python more efficient. In total, we have 15 lists, see the last list (week 15) shown below.

```
> .uu
function(){
"
     *--------------------------------------*
     *  Utility            Short-cut        *
     *------------------------ -------------*
     * .allChapters        # .all           *
     * .calendar           # .cal           *
new->* .code               # .cc            *
     * .codeImage          # .ci            *
     * .dictionary         # .dict          *
new->* .extraCredit        # .ec            *
new->* .fun                                 *
     * .is404                               *
     * .getdata            # .gd            *
     * .homework           # .hw            *
     * .inClassEx          # .ice           *
     * .macUsers           # .mac           *
     * .officeHours        # .oh            *
new->* .preFinal           # .pf            *
     * .preMidTerm         # .pmt           *
     * .searchWebs         # .sw            *
     * .showVideo          # .v2            *
     * .videos             # .v             *
     * .virtualLab         # .vl            *
     * .zoom               # .z             *
     *--------------------------------------*
     * >.ice   # see a list of ice          *
     * >.uu    # back to this utility menu *
     * >.rpy   # back to the main menu      *
     *--------------------------------------*
```

`.dictionary()` for searching commands

To help readers/students search a corresponding Python function for a given R function or vice versa, we have designed a `.dictionary()` function with an abbreviation `.dict()`. This function is used intensively

by both students and instructors. For example, since R is taught before Python, we could enter '`ls`' to find the corresponding Python function, shown below.

```
>
> .dict("ls")
  Name                                R
1 get data from Excel                 >library(xlsx);read.excel()
2 remove all the variables/functions  >rm(list=ls())
3 Show var/our functions              >ls()
4 List all the functions              >ls()
  Python
1 >>>pandas.read_excel()
2
3 >>>dir()
4 >>>print(dir())
>
```

Below is another example related to the `download.file()` function for R and `download()` function for Python.

```
>
> .dict("download")
  Name     R                                Python
1 download >download.file(infile,outfile) >>>download.download(infile,outfile)
>
>
```

In the above code, it is understandable that the first `download` in the Python code is a module while the second one is a function.

The `.searchWebs()` function for convenience

For this book/course, we have generated several hundred small data sets in CSV, RData, sas7bdat, and Pickle formats. To make accessing those data sets more efficient, we have designed a function called `.searchWebs()`. With this function, students and instructors could access a given data set in a few seconds by writing an R or Python program. For example, we are interested in data sets with `co2shares` in their names, shown below.

```
>
> .searchWebs("co2sh")
  ID              NAME                                                WEBSITE
 218 co2 shares csv    http://datayyy.com/data_csv/co2shares.csv
 219 co2 shares RData  http://datayyy.com/data_R/co2shares.RData
 220 co2 shares pickle http://datayyy.com/data_pickle/co2shares.pickle
>
>
```

This is the `co2` emissions based on each country from 1960 to 2020. For the Fama-French 3 or 4 factors, we have the following results.

```
>
> .searchWebs("fama-french mon")
  ID                                        NAME
 501 Fama-French monthly 3 factors
 502 Fama-French monthly 5 factors
 503 Fama-French-carhart monthly 4 factors
 507 Fama-French monthly 3 factors
 508 Fama-French monthly 3 factors pickle
 509 Fama-French monthly 5 factors
 510 Fama-French-carhart monthly 4 factors
                                                WEBSITE
 http://datayyy.com/data_csv/ff3Monthly.csv
 http://datayyy.com/data_csv/ff5Monthly.csv
 http://datayyy.com/data_csv/ffc4Monthly.csv
 http://datayyy.com/data_R/ff3Monthly.RData
 http://datayyy.com/data_pickle/ff3Monthly.pickle
 http://datayyy.com/data_R/ff5Monthly.RData
 http://datayyy.com/data_R/ffc4Monthly.RData
>
```

At least one video for each chapter

For many topics related to R and Python, there exist wonderful YouTube videos. For certain students, they learn better by watching these types of videos rather than sitting in a classroom listening to dry lectures. For programming, hands-on exercises is quite important. On the other hand, for many students, after doing a few in-class exercises, they prefer to view the related videos. For those students, we have generated videos (see a list below).

Title	YouTube links
R basics	*https://www.youtube.com/watch?v=xP6PTpcuciU*
Simple functions using R	*https://youtu.be/uHNgza5QULc*
Open data	*https://youtu.be/Z_90FL91WrA*
Data input using R	*https://youtu.be/q5Xohtni6l4*
Data manipulation using R	*https://youtu.be/2n8_szxbgHE*
Data input using R	*https://youtu.be/MVwum0j-QoM*
R Loops and conditions	*https://youtu.be/7YPTuKlN9b8*
R Loops and conditions (2)	*https://youtu.be/CRpyC7aNDG8*
Plot, graph and moving GIF	*https://youtu.be/xhcUzYMrLZU*
Date, `data.frame()`, list	*https://youtu.be/hDH9oqUqDdw*
Simple matrix manipulations	*https://youtu.be/hm8G5xAJPCw*
String manipulations vai R	*https://youtu.be/A2z4f9hUTLc*
Introduction to R packages	*https://youtu.be/OPgCJ_dBcro*
Zip,SAS, Pickle Google Drive	*https://youtu.be/hISKCEl2nCE*
Python Basics	*https://youtu.be/hvU2HjlOlZ0*
Intro. to Python Modules	*https://youtu.be/Ye5sc3CIuvQ*

Data input using Python	*https://youtu.be/GdfQ1ipyJKM*
True date variable, Data frame	*https://youtu.be/B42AWwQPveM*
Python loops, conditions	*https://youtu.be/z4WkZvvMCkQ*
Data manipulation via Python	*https://youtu.be/ILEIGApAt1Q*
Graphs, plots, visualizations	*https://youtu.be/3sxpUngDCZQ*
Python string manipulations	*https://youtu.be/O-8wYqfJlwo*
Projects: Types I and II	*https://youtu.be/dlxUk02d7yk* *https://youtu.be/0Zq-OA-XT8s*
Comparisons R/ Python	*https://youtu.be/VV0gsypdBwA*

Jointly, those videos were watched over 2,000 times, about 10 times of the number of our students. We generated a function called `.videos` with an abbreviation of `.v`, shown below.

```
>
> .videos
function(i){
" i    Description                              # of videos
 ---   -----------------------------------   -  ----------
  1    Chapter  1: R basics                        4
  2    Chapter  2: Write simple R functions        4  <-
  3    Chapter  3: Open data                       1
  4    Chapter  4: Data input                      3
  5    Chapter  5: Simple data manipulation        1
  6    Chapter  6: Data output                     3
  7    Chapter  7: Loops and conditions            4
  8    Chapter  8: plot, graph and moving GIF      1
  9    Chapter  9: Date variable,data.frame(),list 1
 10    Chapter 10: Matrix manipulations            1
 11    Chapter 11: Simple string manipulations     1
 12    Chapter 12: Introduction to R packages      3
 13    Chapter 13: Zip,SAS,Pickle,Google Drive     1
 14    Chapter 14: Python basics                   4
 15    Chapter 15: Introduction to Python modules  3
 16    Chapter 16: Data input using Python         1
 17    Chapter 17: Data output using Python        1
 18    Chapter 18: Python loops, conditions        1
 19    Chapter 19: Data manipulation using Python  1
 20    Chapter 20: Graphs and plots using Python   0
 21    Chapter 21: String manipulations using Python 1
 22    Chapter 22: Simple projects: Types I and II 1
       ------------------------------------------------
 23    Comparisions between R and Python           1

 Example #1:> .v    # get the above list
 Example #2:> .v(1) # see the first explanation
```

Using R itself to help students learn

The associated course called “Programming for Data Analytics” is designed completely based on R. This is called an R-assisted learning environment. The following table shows 15 weeks of materials.

Week	One-line R code
1	> source("http://datayyy.com/rpy/week1.txt")
2	> source("http://datayyy.com/rpy/week2.txt")
	> # more here
14	> source("http://datayyy.com/rpy/week14.txt")
15	> source("http://datayyy.com/rpy/week15.txt")

The first week's menu is shown below.

```
*-------------------------------------------------------*
* Programming for Data Analytics        2022 by Yan   *
*-------------------------------------------------------*
* .c1  R installation,basics, value assignment          *
* .c2  Writing simple R functions                       *
*-------------------------------------------------------*
* >.c1       # go to chapter 1 (a dot in front of c1) *
* >.uu       # go to the utility menu                  *
* >.rpy      # back to this main menu                  *
*-------------------------------------------------------*
```

And below is the last week's menu:

```
> .rpy()

*-------------------------------------------------------------------------*
* Programming for Data Analytics           2022 by Yan                    *
*-------------------------------------------------------------------------*
* .c1  R basics                        .c14 Python basics                 *
* .c2  Simple R functions              .c15 Python modules                *
* .c3  Open data                       .c16 Python data input             *
* .c4  Data input                      .c17 Python data output            *
* .c5  Simple data manipulation        .c18 Python loops, cond            *
* .c6  Data output                     .c19 Data manipulation             *
* .c7  Loops,if else, logic or, and    .c20 Graphs and plots              *
* .c8  Plot, graph and moving GIF      .c21 Python string manipulations   *
* .c9  Date var, data.frame and list    --- below is optional -----       *
* .c10 Matrix manipulation             .c22 Project I                     *
* .c11 String manipulation                                                *
* .c12 Introduction to R packages                                         *
* .c13 Zip,SAS,Pickle,Google Drive                                        *
*-------------------------------------------------------------------------*
* >.c20      # go to chapter 20                                           *
* >.uu       # go to the utility menu                                     *
* >.rpy      # back to this menu                                          *
*-------------------------------------------------------------------------*
```

Students' related data sets and functions

When teaching this course, we have generated some data sets directly related to students. For example, the following image shows the schedule for the final exams in Fall 2021.

Fall 2021
Class Schedule for Final Week
ALL CLASSES ARE TO MEET (FOR INSTRUCTION OR EXAM) DURING THIS WEEK AT ASSIGNED DATES/TIMES

	Wednesday *December 15*	Thursday *December 16*	Friday *December 17*	Monday *December 20*	Tuesday *December 21*
8:00-10:30am (3) 8:00-11:20am (4)	**MWF** 11:30-12:20 **MW/MF/WF** 11:30-12:45	**TR** 10:11-11:15 10:30-12:10 (4)	**MWF** 1:30-2:20 **MW/MF/WF** 1:00-2:15	**MWF** 8:30-9:20 **MW/MF/WF** 8:30-9:45 8:30-10:10 (4)	**TR** 1:00-2:15
12:00-2:30pm (3) 12:00-3:20pm (4)	**MWF** 9:30-10:20	**TR** 2:30-3:45 2:30-4:10 (4)	**TR** 11:30-12:45 12:30-2:10 (4)	**MF** 2:30-3:45 2:30-4:10 (4)	**TR** 8:30-9:45 8:30-10:10 (4)
3:30-6:00pm (3) 3:30-6:50pm (4)	**MWF** 4:00-4:50 **MW/MF/WF** 4:00-5:15 **MW** 4:30-6:10 (4) 5:00-6:15	**TR** 4:00-5:15 4:30-6:10 (4)	**TR** 5:00-6:15	**MWF** 12:30-1:20 **MW/MF/WF** 12:30-2:10 (4)	**MWF** 10:30-11:20 **MW/MF/WF** 10:00-11:15 10:30-12:10 (4)
7:00-9:30pm (3) 7:00-10:20pm (4)	**MW** 5:30-6:45 6:00-7:15 6:30-8:10 (4) 7:00-8:15	**M** 4:00-6:30 5:30-8:00 6:00-8:30 **& SOE grad courses** (M & MW evening)	**T** 4:00-6:30 5:30-8:00 6:00-8:30 **TR** 5:30-6:45 6:00-7:15 6:30-8:10 (4) 7:00-8:15 **& SOE grad courses** (T evening only)	**W** 4:00-6:30 5:30-8:00 6:00-8:30 **& SOE grad courses** (W evening only)	**R** 4:00-6:30 5:30-8:00 6:00-8:30 **& SOE grad courses** (TR & R evening)

Based on the above table, we have designed two related in-class exercises: 1) generating an R data set, and 2) writing an R function to search this data set. One example is shown below. The input is the weekday and time of the lecture.

```
>
> .exam("mw2:30")
            DATE                                  TIME                    COURSE
1    Monday.May.16 12:00-2:30pm(3)12:00-3:20pm(4)           MWF12:30-1:20
2    Monday.May.16 12:00-2:30pm(3)12:00-3:20pm(4) MW/MF/WF12:30-2:10 (4)
3 Wednesday.May.18   3:30-6:00pm(3)3:30-6:50pm(4)          MW2:30-3:45
4 Wednesday.May.18   3:30-6:00pm(3)3:30-6:50pm(4)          MW2:30-4:10(4)
>
```

The second example is related to the course listing of the next semester. Based on the Excel file offered by the Registrar Office, we generated an R data set and a related R function called `.schedule2022s()`. One example is shown below.

```
>
> .schedule2022s("fnce311")
  DEPT   CRN  SUBJ_CRS              TITLE CrHr DAYS          TIMES  BLDG  RM
1 SBUS 57348 FNCE311_1 Managerial Finance    3   TR 08:30am-09:45am SOUTH 338
2 SBUS 57349 FNCE311_2 Managerial Finance    3   TR 10:00am-11:15am SOUTH 338
3 SBUS 57350 FNCE311_3 Managerial Finance    3   MW 04:00pm-05:15pm SOUTH 338
             INSM PRIMARY_INST CAP  PART_OF_TERM
1 Face-to-Face (1)     Pan, Ken  35 Full Semester
2 Face-to-Face (1)     Pan, Ken  35 Full Semester
3 Face-to-Face (1)    Yan, Paul  35 Full Semester
>
```

Students could use several ways to search, such as subject, an instructor's name, credit hours, department and even the school's name. Since those data sets are directly related to students' lives, they are quite impressed by the power of R.

Potential projects

It is a great idea to do a term project to test a learner's grasp of R or/and Python knowledge and skills. For our teaching, this part is treated as extra credit since learning both R and Python within one semester is already a heavy task. After typing `.project`, the following list would pop up.

```
>
> .project
function(i){
"A list of potential interesting projects.
  [optional and could be used as extra credit]
 i Project examples                i  Projects                          i  Names
-- ------------------------------ --- ---------------------------- --- ----------------------
 1 Three types of projects        21 Issues with import cv2        41 Combine 2 videos    [3 lines]
 2 os.system() function           22 Type II: example #13: ASCII art  42 Images to a video   [4 lines]
 3 Type I: example: text to voice 23          [original code]       43 Add a text to video [5 lines]
 4              [original code]   24 Speech to text (good)          44 GIF from jpg images [5 lines]
 5 Speech to text:(School of Business) 25 issues with import PyAudio[not Done] 45 Movie to one image
 6 Type I:example Scholarly module 26 sys.argv explanation          46 Make a short video
 7 Type I:example:sentiment score  27 Type III: example 14:Google Scholar  47 Cool effect
 8                [original code]  28          [more complete code]  10  48 Generate a sin video
 9 Type II:example: playsound      29 Screenshot every 4 seconds      49 Free vidoes,images,music,and sound
10 Type II:example: fetch http status  30           [II]              50 retrieve text from image
11 Type II:example: text to morse code 31           [original code]
12 Type II:example: PDF to text   32 find dominant color for a given image
13              [original code]   33          [original code]
14 Type II:example add a water mark  34 password generator
15              [original code]   35          [original code]
16 Type II:example tic-tac-toe    36 moviepy:installation
17 Type II:example Sine game      37       :generate a png    [2 lines]
18 Type ii:example check spelling 38       :cut your video    [2 lines]
19 Type II:example: JPEG to png image  39  :rotate 180 degree [2 lines]
20 Type II:example: digital clock 40       :GIF from a video  [2 lines]

  .pp  # short-cut
```

Many examples are very simple with just a few lines. Below is one example.

```
>
> .project(3)
Example #1: text to voice
////////////////////////////////////////////////

from gtts import gTTS
string ="Prof. Yan is teaching Programming for Data Analytics"
x = gTTS(text=string, lang='en', slow=False)
x.save("test.mp3")

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

An interested learner/student could extend it to a whole speech.

Fun programs

When teaching this course, this part is optional because of time limitation and various levels of our students. However, including at least a few such examples from this part could definitively impress many students, and this is especially true for more advanced students. Actually, many examples are very simple. In other words, for most such fun programs, readers and students could apply the copy-and-paste method. The following table shows a few examples.

#	Purpose	Language	Number of lines
1	Generating a RQ code	R	2
2	Generating a Gif rom a video	Python	2
3	Converting one sentence to voice	Python	4
4	Sentiment of a sentence	Python	4
5	Rotating your video	Python	4
6	Cut a long video to a short one	Python	4
7	Shrinking a video to a PNG file	Python	5
8	Adding a text to a video	Python	9
9	Converting PDF to text	Python	11

Individualized exams

This part is for a potential instructor. With R, we designed an individualize exam. For example, students could use the function called `.getFinal()` to get his/her final exam by using his/her first name plus the last two digits of the student ID.

```
> .getFinal()

  Input your first name
   john

  Input the last two digits of  your student ID
   89
```

After hitting the Enter-Key, John will get his final exam. The next section will offer more information.

Paperless exams

This part is for a potential instructor. To make programming more appealing, R is used to design a paperless exam. The menu is shown below.

```
*-------------------------------------------------------------------*
* Individualized Exams            2021 by Yan paulyxy@gmail.com *
*-------------------------------------------------------------------*
* Function              short-cut  Function           short-cut *
*-------------------------------------------------------------------*
* .readme                    .rm    .upload2remoteDirs    .s12   *
*-------------------------------------------------------------------*
* .setup                     .s1    .backupPlans          .b     *
* .downloadInputFiles        .s2                                 *
*-------------------------------------------------------------------*
* .fromInput7                .s3    .solution2gradingDir  .s13   *
* .generateGetExamRData      .s4    .getVersionSolution   .s14   *
* .generate_f888txt          .s5    .idSolutionTxtRData   .s15   *
* .generate35emptyTxt        .s6    .checkInput1          .c1    *
* .generate35emptyRData      .s7    .showVersions         .sv    *
* .generate35oneLineRemote1  .s8    .showVersionSolution  .svs   *
* .generate35testTXT         .s9                                 *
* .generate35testRData       .s10   .gradeMC              .mc    *
* .generate35done            .s11   .showID               .si    *
* .runAllStep3to11           .ra    .showExam             .se    *
* .start                     .st                                 *
*-------------------------------------------------------------------*
* >.readme                   # readme (a dot in front of readme) *
* >.setup                    # see the uages of the fucntion     *
* >.ind                      # back to this menu                 *
*-------------------------------------------------------------------*
```

Any student and learner could design their own exam as well.

Research projects

This part is optional if this textbook is used for just one semester. For a business school student or professional who finished the first 22 chapters, one obvious question is how to apply what have learned to his/her job and professional areas. To help those readers, students, and professionals, we have the last three chapters. The first example is to parse the SEC Financial Statement Data Sets which has data from 2009 up to today. The second example is to parse the Census 2020 data.

What this book covers

Chapter 1: R Basics explains how to install R, its basics and value assignments. Those are the most basic concepts for learning a computer language. Later in the book, we will use those concepts repeatedly. Thus, a new user should have certain confidence if he/she feels overwhelmed by those new concepts.

Chapter 2: Simple Functions using R discusses how to write one-line R function. Then, we will explain how to extend it to a multi-line function. In addition, we explain how to add comments to make our functions more readable. It means that we offer the objective of the functions, definitions of input variables, and any default values, plus a

few examples. Because of those extra comments, our functions become self-explanatory.

Chapter 3: Open Data shows many open-source data related to business education, such as the SEC filings, SEC financial statement data sets, Census data (2010), the MCDC (Missouri Census Data Center) data platform, Professor French's Data Library, and FRED (Federal Research Economics Data Library).

Chapter 4: Data Input using R explains several ways to retrieve various types of data, such as CSV (Comma Separated Values), text, and Excel files. The related R functions are `read.csv()`, `read.table()`, `load()`, `read(url())`, `readRDS()`, `read_excel()`, and the like.

Chapter 5: Simple Data Manipulation using R discusses issues related to data processing. First, we will introduce some basic concepts, such as scalar, vector and matrix. Then, we show how to convert vectors into a matrix and how to get a subset from a matrix. The related R functions are `matrix()`, `as.matrix()`, `c()`, `cbind()`, `rbind()`, `data.frame()`, `nrow()`, `dim()`, `ncol()`, `colnames()`, and `rownames()`.

Chapter 6: Data Output using R explains various ways to output our data and results. For text files and CSV formats, two of the most frequently used R functions are `write.csv()` and `write.table()`. To save an R data set, we could use the `save()` and `saveRDS()` functions. In addition, we also discussed other formats such as writing to a clipboard. To save all our R objects, the `save.image()` function is applied.

Chapter 7: Loops and Conditions introduces `for` loops and `while` loops. After that, we will discuss double loops. For example, we are working with just one company's data. After we are satisfied with our programs, we include our program within a loop. Thus, processing 500 companies will be equivalent to processing just one company's data. This is the power of loops. In addition, we discuss various conditions: `if`, `if-else`, `if`, `else if` and `else`. Those conditions are used to control the flow of our programs. To mimic the Excel function `if()`, we have the equivalent R `ifelse()` function.

Chapter 8: Simple Plots and Graphs using R generates various plots and graphs. It is quite important for visual representation, and this is true for our data sets. The related R functions include `plot()` and `hist()`. For a better visual presentation, we mention a few related R packages such as `png`, `ggplot2`, and `ggridges`. As an optional part, we cover how to a generate moving gif.

Chapter 9: A True Date Variable, R data frame and List discusses why we need a true date variable and how to define it. Then, we will show how to use this true date variable to cut our data and merge two or three data sets. Then, we discuss the applications of the `data.frame()` and `list()` functions. With the `data.frame()` function, we can include different types of data, such as string and numerical variables. In addition, we will learn how to convert a list into a matrix by using various R functions, such as `as.matrix()`, `matrix()`, and `delist()`.

Chapter 10: Matrix Manipulation shows how to define a matrix and the applications of a matrix. More importantly, we will explain several most widely used R functions such as `matrix()` and `as.matrix()`, and how to use a matrix to manipulate data to simplify our code. For example, the code to estimate portfolio returns is much simpler by using matrix multiplication.

Chapter 11: Simple String Manipulation using R discusses many functions associated to string manipulation, such as using `paste()` and `paste0()` to merge two strings, and using the `sub()` and `gsub()` functions to substitute different phrases. Other functions include `length()`, `nchar()`, `grep()`, `abbreciate()` and a few embedded variables, such as `letters` and `LETTERS`.

Chapter 12: Introduction to R Packages first explains why the knowledge related to R packages is important. Then, we show how to install a package, find a list of all available R packages today, and how to find the manual for a specific package.

Chapter 13: Zip, Pickle, SAS Data and Google Drive first explains why users compress their data sets, then discusses how to use the `zip()` and `unzip()` functions, contained in the R `zip` package, to compress and un-compress our data. Since many available data sets are in SAS and pickle (Python data sets) formats, we discuss functions to retrieve those types of data. There is no doubt that Google Drive is widely used for containing or sharing data, so we explain the related R functions to retrieve data from others' Google Drive with permission.

Chapter 14: Python Basics switches gears from R to Python. First, we will explain why the knowledge related to Python is important. We then show several ways to install Python, and some basic concepts, such as how to assign values to a new variable, whether Python is case sensitive, and how to write simple Python functions. To make it a little easier to learn Python, we will constantly compare it with R.

Chapter 15: Introduction to Python Modules first discusses the importance of Python modules. More specifically, we will discuss how to find some relevant Python Modules, and how to install/update individual

Python modules. To make our introduction easy to understand, we will more time on the three most frequently used modules, such as `numPy`, `pandas`, and `matplotlib`. In *Chapter 19: Data manipulation via Python,* we will spend more time on `pandas`, while for *Chapter 24: Simple graphs via Python,* we will spend more time on `matplotlib`.

Chapter 16: Data Input using Python discusses various ways to input data such as CSV (Comma Separated Values), text, pickle, and other types. To achieve this goal, we will focus on the `pandas` Python module. For data input, we will have the functions `pd.read_csv()`, `pd.read_excel()`, `pd.read_pickle()`, `pd.read_sas()`, and `pd.read_table()`. For output, we will have `pd.to_pickle()`, `pd.to_csv()`, and the like. Here, `pd` is a common short name for the `pandas` Python module.

Chapter 17: A True Date Variable and Data Output using Python discusses three parts. Firstly, we discuss how to generate a true date variable, such as how to assign a date variable, add `n` days to a date variable, calculate the difference between two date variables, and get the year, month, and day from a date variable. Secondly, we discuss the Python Data Frame and compare it with that in R. Thirdly, we show how to export our data to a CSV, text, or pickle data set. We also discuss the `df.to_csv()` and `df.to_pickle()` functions. And finnally, we offered a few examples, such as how to generate `ff3Monthly.pickle`, `titanic.pickle`, and `covid19.pickle`.

Chapter 18: Python Loops and Conditions discusses various loops, such as `for` and `while` loops. In addition, we will explain how to use various conditions to direct and re-direct the flows of our programs.

Chapter 19: Simple Data Manipulation using Python will discuss many basic techniques for data manipulation such as how to generate a `pandas.DataFrame`, how to change the column names, add and delete columns. After that, we explain how to get a list of all the possible attached functions for `pandas.DataFrame` by typing the name of a `pandas.DataFrame` followed by a dot (`.`), and then hitting the Tab-key. A `pandas.DataFrame` has many functions attached to it, such as `sum`, `min`, `max`, `median`, `sd`, `sort_values`, and `groupby`. After that, we explain 4 types of merges: inner, left, right, and outer.

Chapter 20: Simple plots and graphs using Python will discuss how to use the `matplotlib` Python module to generate various types of graphs, such as a drawing a normal distribution, a straight-line, a histogram, and the volatility graphs for a given stock.

Chapter 21: Simple String Manipulation using Python will discuss many concepts and issues related to string manipulations, such as how to

replace an old substring with a new one, search a fixed substring, convert a number to a string, and vice versa. Other functions include `s.replace()`, `s.index()`, `s.rindex()`, `s.join()`, `s.split()`, and `s.strip()`. In addition, we introduce the concept of Regular Expressions.

Chapter 22: Project: Types I and II will discuss three types of projects students could do as a term project. This chapter focuses on the first two types: Type I: study and explain one R or Python package by searching *http://r-project.org* and *http://pypi.org*, and Type II: search GitHub at *https://github.com* or online to finish a simple project. Over 20 such simple projects are offered in this chapter.

Chapter 23: Project Type III: SEC Filings and Financial Statement Analysis first shows how to download the SEC index quarterly zip and text files. The SEC filings is a gold mine, and this is especially true for business school students. Another good project is to download the SEC Financial Statement Data Sets and process them to generate Balance-sheet and Income Statement data sets with `RData` and `pickle` (Python data) extensions. As for the second part (not related to the first part), we will discuss how to download many data sets from various sources, such as the SEC (US Securities and Exchange Commission), Census, Prof. French's Data Library and other sources.

Chapter 24: Project Type III: Census Data shows how to download data from the Census 2010 and 2020 databases. This is a great exercise since the data sets are relatively large. In addition, the programs are more complex to merge different data sets.