Chapter 2. Approaches to Valuation
R Commentary

Three simple programs are illustrated supporting this chapter providing the opportunity to learn more elementary R coding techniques.

Approaches to financial valuation

Market Comparables Approach Test.R
From this chapter, one way to express the market comparables approach (MCA) is

$$V = \sum_{k=1}^{s} \alpha_k P_k \text{ (Market comparable approach),}$$

where V denotes the value of some instrument, α_k denotes number of units of instrument k held, and P_k denotes the value of some instrument k. Note, we must assume that none of the instruments in $k = 1, \ldots, s$ are themselves the instrument being valued (V).

Thus, just to illustrate simple R programming, we consider the value of say a stock portfolio where the number of shares (α_k) and prices (P_k) are contained in MCA Inputs.xlsx. Table 2R.1 provides the contents of this spreadsheet.

Table 2R.1. MCA Inputs.xlsx data

Alpha	Price
100	100.00
90	110.00
80	120.00
70	130.00
60	140.00
50	150.00
40	160.00
30	170.00
20	180.00
10	190.00

The source code is contained in two files, a setup file (with Test.R extension) and calculations file (with Calculations.R extension).

```
# Market Comparables Approach Test.R
# rmarkdown::render("Market Comparables Approach Test.R", "word_document")
rm(list = ls())  # Take out the Environment "trash"
cat("\014") # Clear Console, making error checking easier.
while (!is.null(dev.list()))  dev.off() # Clear old plots
par(family = 'Times New Roman') # Globally set fonts for graphs
defaultpar <- par() # plot global parameters
# Libraries
#   openxlsx: MS Excel file management
Packages <- c("openxlsx")
if(length(setdiff(Packages, rownames(installed.packages()))) > 0) {
  install.packages(setdiff(Packages, rownames(installed.packages())))
} # Make sure libraries are installed on this computer
lapply(Packages, library, character.only=TRUE) # Load and attach libraries
rm(Packages)
#
# Simple program illustrating reading and writing xlsx files
#
FileName = 'MCA Inputs.xlsx'
source('MCA Calculations.R')
paste0('Print of input file: MCA Inputs.xlsx')
MCA
```

```
paste0('MCA Value: $',formatC(Value, big.mark=',', format = 'f', digits = 2))
```

R calculations file for MCA.

```
# MCA Calculations.R
MCA <- read.xlsx(FileName, sheet = 1, startRow = 1, colNames = TRUE,
  rowNames = FALSE, detectDates = FALSE, skipEmptyRows = TRUE,
  rows = NULL, cols = NULL, check.names = FALSE, namedRegion = NULL)
MCA # Check to be sure data is present
is.data.frame(MCA) # Check if data frame
dimnames(MCA) # Check dimension names
# Simple way to compute the sum of the product of Alpha and Price
Value = MCA$Alpha %*% MCA$Price # Sum of product (matrix multiplication)
# More transparent and readable approach
NItems <- length(MCA$Alpha)
Value1 <- 0
for(i in 1:NItems){
  Value1 = Value1 + MCA$Alpha[i]*MCA$Price[i]
}
# Using built in functions
Value2 = crossprod(MCA$Alpha, MCA$Price)
options(scipen = 999) # Do not present in scientific notation
Value; Value1; Value2
# Illustration when you do want to print to console
print("MCA Calculation.R Value = ")
print(Value)
```

The results from running the test file:

```
> while (!is.null(dev.list()))  dev.off() # Clear old plots
> par(family = 'Times New Roman') # Globally set fonts for graphs
> defaultpar <- par() # plot global parameters
> # Libraries
> #   openxlsx: MS Excel file management
> Packages <- c("openxlsx")
> if(length(setdiff(Packages, rownames(installed.packages()))) > 0) {
+   install.packages(setdiff(Packages, rownames(installed.packages())))
+ } # Make sure libraries are installed on this computer
> lapply(Packages, library, character.only=TRUE) # Load and attach libraries
[[1]]
[1] "date"      "openxlsx"  "stats"     "graphics"  "grDevices" "utils"     "datasets"
[8] "methods"   "base"
> while (!is.null(dev.list()))  dev.off() # Clear old plots
> par(family = 'Times New Roman') # Globally set fonts for graphs
> defaultpar <- par() # plot global parameters
> # Libraries
> #   openxlsx: MS Excel file management
> Packages <- c("openxlsx")
> if(length(setdiff(Packages, rownames(installed.packages()))) > 0) {
+   install.packages(setdiff(Packages, rownames(installed.packages())))
+ } # Make sure libraries are installed on this computer
> lapply(Packages, library, character.only=TRUE) # Load and attach libraries
[[1]]
[1] "date"      "openxlsx"  "stats"     "graphics"  "grDevices" "utils"     "datasets"
[8] "methods"   "base"
> rm(Packages)
> #
> # Simple program illustrating reading and writing xlsx files
> #
> FileName = 'MCA Inputs.xlsx'
> source('MCA Calculations.R')
> paste0('Print of input file: MCA Inputs.xlsx')
[1] "Print of input file: MCA Inputs.xlsx"
> MCA
   Alpha Price
```

```
1     100   100
2      90   110
3      80   120
4      70   130
5      60   140
6      50   150
7      40   160
8      30   170
9      20   180
10     10   190
> paste0('MCA Value: $',formatC(Value, big.mark=',', format = 'f', digits = 2))
[1] "MCA Value: $71,500.00"
```

We now turn to a simple R program to compute CFAA-based value.

Cash Flow Adjusted Approach Test.R

From this chapter, one way to express the cash flow adjusted approach (CFAA) is

$$V = \sum_{t=1}^{T} \frac{1}{(1+r_t)^t} \sum_{j=1}^{m} q_{t,j} CF_{t,j} \text{ (Cash flow adjusted approach).}$$

where V denotes the value of some instrument, r_t denotes the annually compounded risk-free interest rate, $CF_{i,t,}$ denotes the cash flow on this instrument at time t, when state j occurs, $q_{t,j}$ denotes the equivalent martingale measure probability of state j occuring at time t.

The data inputs are contained in CFAA Inputs.xlsx, where to keep the coding simple, the values of T and m are specified in the R code.

Thus, just to illustrate simple R programming, we consider the value of some project where the appropriate inputs are contained in CFAA Inputs.xlsx (reproduced in Table 2R.2). In this case, the project lasts five years (t) and results in end of year cash flows depending on five different potential states of the world (j). The appropriate discount rate (r), probability (q), and cash flow (CF) are also specified.

Table 2R.2. CFAA Inputs.xlsx data

t	j	r	q	CF
1.000	1	2.00%	1.0%	$1.00
1.000	2	2.00%	24.0%	$2.00
1.000	3	2.00%	50.0%	$3.00
1.000	4	2.00%	24.0%	$4.00
1.000	5	2.00%	1.0%	$5.00
2.000	1	3.00%	8.0%	($5.00)
2.000	2	3.00%	22.0%	$5.00
2.000	3	3.00%	40.0%	$15.00
2.000	4	3.00%	22.0%	$25.00
2.000	5	3.00%	8.0%	$35.00
3.000	1	3.50%	15.0%	($15.00)
3.000	2	3.50%	20.0%	$0.00
3.000	3	3.50%	30.0%	$15.00
3.000	4	3.50%	20.0%	$30.00
3.000	5	3.50%	15.0%	$45.00
4.000	1	4.00%	20.0%	($25.00)
4.000	2	4.00%	20.0%	$0.00
4.000	3	4.00%	20.0%	$25.00
4.000	4	4.00%	20.0%	$50.00
4.000	5	4.00%	20.0%	$75.00
5.000	1	4.25%	20.0%	($50.00)
5.000	2	4.25%	20.0%	$0.00
5.000	3	4.25%	20.0%	$50.00
5.000	4	4.25%	20.0%	$100.00
5.000	5	4.25%	20.0%	$195.65

The source code is contained in two files, a setup file (with Test.R extension) and calculations file (with Calculations.R extension).

```
# Cash Flow Adjusted Approach Test.R
# rmarkdown::render("Cash Flow Adjusted Approach Test.R", "word_document")
rm(list = ls())  # Take out the Environment "trash"
```

```
cat("\014") # Clear Console, making error checking easier.
while (!is.null(dev.list()))  dev.off() # Clear old plots
par(family = 'Times New Roman') # Globally set fonts for graphs
defaultpar <- par() # plot global parameters
# Libraries
#   openxlsx: MS Excel file management
Packages <- c("openxlsx")
if(length(setdiff(Packages, rownames(installed.packages()))) > 0) {
  install.packages(setdiff(Packages, rownames(installed.packages())))
} # Make sure libraries are installed on this computer
lapply(Packages, library, character.only=TRUE) # Load and attach libraries
rm(Packages)
#
# Simple program illustrating reading and writing xlsx files
#
T = 5 # Number of future points in time
m = 5 # Number of states at each future point in time
FileName = 'CFAA Inputs.xlsx'
source('CFAA Calculations.R')
paste0('Print of input file: CFAA Inputs.xlsx')
CFAA
paste0('CFAA Value: $',formatC(Value, big.mark=',', format = 'f', digits = 2))
```

R calculations file for CFAA.

```
# CFAA Calculations.R
CFAA <- read.xlsx(FileName, sheet = 1, startRow = 1, colNames = TRUE,
  rowNames = FALSE, detectDates = FALSE, skipEmptyRows = TRUE,
  rows = NULL, cols = NULL, check.names = FALSE, namedRegion = NULL)
CFAA # Check to be sure data is present
is.data.frame(CFAA) # Check if data frame
dimnames(CFAA) # Check dimension names
# Check if inputs correct
NItems <- length(CFAA$t)
NItemsCheck <- T * m
if(NItems != NItemsCheck)paste0('ERROR: T and m do not match file. Results wrong!!')
Value <- 0
Counter <- 0
for(t in 1:T){
  for(j in 1:m){
    Counter <- Counter + 1
    Value = Value + CFAA$q[Counter]*CFAA$CF[Counter]/(1 +
CFAA$r[Counter])^CFAA$t[Counter]
  }
}
options(scipen = 999) # Do not present in scientific notation
Value
# Illustration when you do want to print to console
print("CFAA Calculation.R Value = ")
print(Value)
```

The results from running the test file:

```
> #
> # Simple program illustrating reading and writing xlsx files
> #
> T = 5 # Number of future points in time
> m = 5 # Number of states at each future point in time
> FileName = 'CFAA Inputs.xlsx'
> source('CFAA Calculations.R')
> paste0('Print of input file: CFAA Inputs.xlsx')
[1] "Print of input file: CFAA Inputs.xlsx"
> CFAA
   t j      r    q     CF
1  1 1 0.0200 0.01   1.00
```

```
2  1 2 0.0200 0.24   2.00
3  1 3 0.0200 0.50   3.00
4  1 4 0.0200 0.24   4.00
5  1 5 0.0200 0.01   5.00
…
25 5 5 0.0425 0.20 195.65
> paste0('CFAA Value: $',formatC(Value, big.mark=',', format = 'f', digits = 2))
[1] "CFAA Value: $100.00"
```

Discount Factor Adjusted Approach Test.R

From this chapter, one way to express the discount factor adjusted approach (DFAA) is

$$V = \sum_{t=1}^{T} \frac{1}{\left(1 + r_t + RP_t\right)^t} \sum_{j=1}^{m} p_{t,j} CF_{t,j} \text{ (Discount factor adjusted approach).}$$

where the notation is as described for CFAA, except RP_t notes the risk premium on this instrument at time t, when state j occurs, and $p_{t,j}$ denotes the analysts subjective probability of state j occuring at time t.

The data inputs are contained in DFAA Inputs.xlsx, where to keep the coding simple, the values of T and m are specified in the R code.

Thus, just to illustrate simple R programming, we consider the value of the same project presented with CFAA where the appropriate inputs are contained in DFAA Inputs.xlsx (reproduced in Table 2R.3). In this case, the project lasts five years and results in end of year cash flows depending on five different potential states of the world. In this case, we have inserted an additional time varying risk premium.

Table 2R.3. DFAA Inputs.xlsx data

t	j	r	RP	q	CF
1.000	1	2.00%	0.50%	1.0%	$1.00
1.000	2	2.00%	0.50%	24.0%	$2.00
1.000	3	2.00%	0.50%	50.0%	$3.00
1.000	4	2.00%	0.50%	24.0%	$4.00
1.000	5	2.00%	0.50%	1.0%	$5.00
2.000	1	3.00%	0.75%	8.0%	($5.00)
2.000	2	3.00%	0.75%	22.0%	$5.00
2.000	3	3.00%	0.75%	40.0%	$15.00
2.000	4	3.00%	0.75%	22.0%	$25.00
2.000	5	3.00%	0.75%	8.0%	$35.00
3.000	1	3.50%	1.00%	15.0%	($15.00)
3.000	2	3.50%	1.00%	20.0%	$0.00
3.000	3	3.50%	1.00%	30.0%	$15.00
3.000	4	3.50%	1.00%	20.0%	$30.00
3.000	5	3.50%	1.00%	15.0%	$45.00
4.000	1	4.00%	1.50%	20.0%	($25.00)
4.000	2	4.00%	1.50%	20.0%	$0.00
4.000	3	4.00%	1.50%	20.0%	$25.00
4.000	4	4.00%	1.50%	20.0%	$50.00
4.000	5	4.00%	1.50%	20.0%	$75.00
5.000	1	4.25%	2.00%	20.0%	($50.00)
5.000	2	4.25%	2.00%	20.0%	$0.00
5.000	3	4.25%	2.00%	20.0%	$50.00
5.000	4	4.25%	2.00%	20.0%	$100.00
5.000	5	4.25%	2.00%	20.0%	$195.65

The source code is contained in two files, a setup file (with Test.R extension) and calculations file (with Calculations.R extension).

```
# Discount Factor Adjusted Approach Test.R
# rmarkdown::render("Discount Factor Adjusted Approach Test.R", "word_document")
rm(list = ls())  # Take out the Environment "trash"
cat("\014") # Clear Console, making error checking easier.
while (!is.null(dev.list()))  dev.off() # Clear old plots
par(family = 'Times New Roman') # Globally set fonts for graphs
defaultpar <- par() # plot global parameters
# Libraries
#   openxlsx: MS Excel file management
```

```
Packages <- c("openxlsx")
if(length(setdiff(Packages, rownames(installed.packages()))) > 0) {
  install.packages(setdiff(Packages, rownames(installed.packages())))
} # Make sure libraries are installed on this computer
lapply(Packages, library, character.only=TRUE) # Load and attach libraries
rm(Packages)
#
# Simple program illustrating reading and writing xlsx files
#
T = 5 # Number of future points in time
m = 5 # Number of states at each future point in time
FileName = 'DFAA Inputs.xlsx'
source('DFAA Calculations.R')
paste0('Print of input file: DFAA Inputs.xlsx')
DFAA
paste0('DFAA Value: $',formatC(Value, big.mark=',', format = 'f', digits = 2))
```

R calculations file for DCFAA.

```
# DFAA Calculations.R
DFAA <- read.xlsx(FileName, sheet = 1, startRow = 1, colNames = TRUE,
  rowNames = FALSE, detectDates = FALSE, skipEmptyRows = TRUE,
  rows = NULL, cols = NULL, check.names = FALSE, namedRegion = NULL)
DFAA # Check to be sure data is present
is.data.frame(DFAA) # Check if data frame
dimnames(DFAA) # Check dimension names
# Check if inputs correct
NItems <- length(DFAA$t)
NItemsCheck <- T * m
if(NItems != NItemsCheck)paste0('ERROR: T and m do not match file. Results wrong!!')
Value <- 0
Counter <- 0
for(t in 1:T){
  for(j in 1:m){
    Counter <- Counter + 1
    Value = Value + DFAA$q[Counter]*DFAA$CF[Counter] /
      (1 + DFAA$r[Counter] + DFAA$RP[Counter])^DFAA$t[Counter]
  }
}
options(scipen = 999) # Do not present in scientific notation
Value
# Illustration when you do want to print to console
print("MCA Calculation.R Value = ")
print(Value)
```

The results from running the test file:

```
#
> # Simple program illustrating reading and writing xlsx files
> #
> T = 5 # Number of future points in time
> m = 5 # Number of states at each future point in time
> FileName = 'DFAA Inputs.xlsx'
> source('DFAA Calculations.R')
> paste0('Print of input file: DFAA Inputs.xlsx')
[1] "Print of input file: DFAA Inputs.xlsx"
> DFAA
   t j      r     RP    q     CF
1  1 1 0.0200 0.0050 0.01   1.00
2  1 2 0.0200 0.0050 0.24   2.00
3  1 3 0.0200 0.0050 0.50   3.00
4  1 4 0.0200 0.0050 0.24   4.00
5  1 5 0.0200 0.0050 0.01   5.00
6  2 1 0.0300 0.0075 0.08  -5.00
7  2 2 0.0300 0.0075 0.22   5.00
```

```
8  2 3 0.0300 0.0075 0.40  15.00
9  2 4 0.0300 0.0075 0.22  25.00
10 2 5 0.0300 0.0075 0.08  35.00
11 3 1 0.0350 0.0100 0.15 -15.00
12 3 2 0.0350 0.0100 0.20   0.00
13 3 3 0.0350 0.0100 0.30  15.00
14 3 4 0.0350 0.0100 0.20  30.00
15 3 5 0.0350 0.0100 0.15  45.00
16 4 1 0.0400 0.0150 0.20 -25.00
17 4 2 0.0400 0.0150 0.20   0.00
18 4 3 0.0400 0.0150 0.20  25.00
19 4 4 0.0400 0.0150 0.20  50.00
20 4 5 0.0400 0.0150 0.20  75.00
21 5 1 0.0425 0.0200 0.20 -50.00
22 5 2 0.0425 0.0200 0.20   0.00
23 5 3 0.0425 0.0200 0.20  50.00
24 5 4 0.0425 0.0200 0.20 100.00
25 5 5 0.0425 0.0200 0.20 195.65
> paste0('DFAA Value: $',formatC(Value, big.mark=',', format = 'f', digits = 2))
[1] "DFAA Value: $93.85"
>
```