Module 4.3: Valuation Stocks
R Commentary

See module *Ch 4.3 Valuation Stocks*. The *N*-stage dividend discount model is illustrated in *NDDM Valuation Test.R*. Applying the LSC model to the present value of dividends is illustrated in *NDDM with LSC Fit Test.R* with numerous supporting files.

NDDM Valuation Test.R (Selected Excerpts and Output)

All initial parameters are set early in this file. The program illustrates the four factor model illustrated above.

```
# NDDM Inputs
DaysPerYear <- 365.25
DividendsPerYear <- 4 # Dividend policy changes 1x per year
DeltaTau <- 1/DividendsPerYear # Time between dividends (except stub)
#
# Stub Component Inputs
#
LastDividend <- 1.0 # Quarterly dividend dollar amount
DeltaTau1 <- 0.1 # Only if on a dividend payment date, otherwise input value
StubPayments <- 2  # Number of dividends until next change in dividend amount
ForwardRate0 <- 0.10 # Continuously compounded, annualized
#
# Series Component Inputs
#
NumberOfStages <- 3
# NumberOfStages+3: Stub, last infinite series and totals
```

Model parameters managed within a dataframe.

```
Series = data.frame(matrix(vector(), NumberOfStages+3, 6, dimnames=list(c(),
  c("Stage", "ForwardRate", "GrowthRate", "YearsInStage", "SeriesValue",
    "InitialDividend"))), stringsAsFactors=F)
Series$Stage[1] <- 0
Series$ForwardRate[1] <- ForwardRate0
Series$GrowthRate[1] <- NA
Series$YearsInStage[1] <- StubPayments
Series$InitialDividend[1] <- LastDividend
Series$Stage[2] <- 1
Series$ForwardRate[2] <- 0.12
Series$GrowthRate[2] <- 0.06
Series$YearsInStage[2] <- 5
Series$InitialDividend[2] <- LastDividend
Series$Stage[3] <- 2
Series$ForwardRate[3] <- 0.09
Series$GrowthRate[3] <- 0.03
Series$YearsInStage[3] <- 5
Series$InitialDividend[3] <- LastDividend*exp(Series$GrowthRate[2] *
  Series$YearsInStage[2])
Series$Stage[4] <- 3
Series$ForwardRate[4] <- 0.07
Series$GrowthRate[4] <- 0.01
Series$YearsInStage[4] <- 5
Series$InitialDividend[4] <- Series$InitialDividend[3] *
  exp(Series$GrowthRate[3]*Series$YearsInStage[3])
Series$Stage[5] <- 4
Series$ForwardRate[5] <- 0.06
Series$GrowthRate[5] <- 0.0
Series$YearsInStage[5] <- Inf
Series$InitialDividend[5] <- Series$InitialDividend[4] *
  exp(Series$GrowthRate[4]*Series$YearsInStage[4])
# Last stage needs very large number: Need these PVDs for future research
if(identical(Series$YearsInStage[NumberOfStages+2], Inf)){
  Series$YearsInStage[NumberOfStages+2] = 150
}
```

Once the particulars of an individual stock have been appropriately acquired, we compute a variety of outputs related to each dividend payment and then write this output to two files, Dividends and Series.

```
# Infinite series, but use finite to estimate:
#   Will need each dividend for subsequent work
write.xlsx(Div, file = "Dividends.xlsx", col.names = TRUE,
  sheetName = "Dividends", showNA = TRUE, row.names = FALSE)
write.xlsx(Series, file = "Series.xlsx", col.names = TRUE,
  sheetName = "Series", showNA = TRUE, row.names = FALSE)
StockValue <- Series$SeriesValue[NumberOfStages+3]
StockValue
```

Figure 4R.3.1 illustrates excerpts from the Dividend.xlsx file produced. PV denotes the present value of $1 whereas PVD denotes the present value of the dollar dividend.

Figure 4R.3.1 Excerpts from Dividend.xlsx file

	A	B	C	D
1	**MaturityTime**	**DollarDividend**	**PV**	**PVD**
2	0.10	$1.00	$0.99005	$0.99005
3	0.35	$1.00	$0.96561	$0.96561
4	0.60	$1.06	$0.93707	$0.99501
5	0.85	$1.06	$0.90937	$0.96561
6	1.10	$1.06	$0.88250	$0.93707
7	1.35	$1.06	$0.85642	$0.90937
8	1.60	$1.13	$0.83110	$0.93707
9	1.85	$1.13	$0.80654	$0.90937
10	2.10	$1.13	$0.78270	$0.88250
653	162.85	$1.65	$0.00003	$0.00006
654	163.10	$1.65	$0.00003	$0.00006
655	163.35	$1.65	$0.00003	$0.00005
656	163.60	$1.65	$0.00003	$0.00005
657	163.85	$1.65	$0.00003	$0.00005
658	164.10	$1.65	$0.00003	$0.00005
659	164.35	$1.65	$0.00003	$0.00005
660	164.60	$1.65	$0.00003	$0.00005
661	164.85	$1.65	$0.00003	$0.00005
662	165.10	$1.65	$0.00003	$0.00005
663	165.35	$1.65	$0.00003	$0.00005

Recall Figure 4.3.3 plots the present value of dividends with respect to maturity. We clearly see the influence of annual dividend growth as well as the significant downward pressure on the present value of dividends as we move out in maturity time.

Figure 4R.3.2 illustrates excerpts from the Dividend.xlsx file produced. PV denotes the present value of $1 whereas PVD denotes the present value of the dollar dividend.

Figure 4R.3.2 Results contained in the Series.xlsx file

	A	B	C	D	E	F
1	**Stage**	**ForwardRate**	**GrowthRate**	**YearsInStage**	**SeriesValue**	**InitialDividend**
2	0	10%		2	$1.95566	$1.00000
3	1	12%	6%	5	$16.94363	$1.00000
4	2	9%	3%	5	$12.40868	$1.34986
5	3	7%	1%	5	$9.12275	$1.56831
6	4	6%	0%	150	$25.97338	$1.64872
7					$66.40409	

Therefore, the stock is valued at $66.40.

NDDM with LSC Fit Test.R (Selected Excerpts and Output)

All initial parameters related to applying the LSC model are set early in this file. We also read in the output from the previous program as well as set some switches for plots to produce.

```
# Plots to produce:
PlotPVD <- TRUE
PlotlnPVD <- FALSE
# Number of factors to fit:
LSC1 <- FALSE
LSC2 <- TRUE
LSC3 <- TRUE
LSC4 <- TRUE
LSC5 <- FALSE
LSC6 <- FALSE
# Fit forward version of LSC
ForwardLSC <- FALSE
# NDDM Inuts from file created by 4.3 NDDM with LSC Fit Test.R
Div <- read.xlsx(xlsxFile = "Dividends.xlsx", sheet = 1, skipEmptyRows=FALSE)
Series <- read.xlsx(xlsxFile = "Series.xlsx", sheet = 1, skipEmptyRows=FALSE)
LengthDiv <- length(Div$PVD)
LengthSeries <- length(Series$SeriesValue)
StockValue <- Series$SeriesValue[LengthSeries]
SumDiv <- 0
SumWeight <- 0
for(i in 1:LengthDiv){
  Div$lnPVD[i] <- log(Div$PVD[i])
  Div$Weight[i] = Div$PVD[i] / StockValue
  SumDiv = SumDiv + Div$PVD[i]
  SumWeight = SumWeight + Div$Weight[i]
}
SumDiv; StockValue; SumWeight
# LSC inputs
NumberOfFactors <- 6
N <- NumberOfFactors - 2
Scalar <- c(1:N)
Scalar[1] <- 10
Scalar[2] <- 20
Scalar[3] <- 50
Scalar[4] <- 80
#
# Work on LSC fitting
#
NumberOfMaturities <- LengthDiv
Tau <- c(1:N)
Tau <- Scalar
NumberOfDates <- 1
Factors <- matrix(nrow = NumberOfFactors - 1, ncol = NumberOfMaturities)
Factors2 <- matrix(nrow = NumberOfFactors - 1, ncol = NumberOfMaturities)
NP = NumberOfFactors - 1
#
# Factors: Standard LSC factors based originally on spot rates
# Factors2: Alternative LSC factors based originally on forward rates
#  Does not appear to make a difference
#
for (j in 1:NP) {
  for (i in 1:NumberOfMaturities) {
    if (j == 1) {
      Factors[j,i] = (1.0 - exp(-Div$MaturityTime[i]/Tau[j])) /
        (Div$MaturityTime[i]/Tau[j])
      Factors2[j,i] = exp(-Div$MaturityTime[i]/Tau[j])
    } else {
      Factors[j, i] = ((1.0 - exp(-Div$MaturityTime[i]/Tau[j-1])) /
        (Div$MaturityTime[i]/Tau[j-1])) - exp(-Div$MaturityTime[i]/Tau[j-1])
      Factors2[j, i] = (Div$MaturityTime[i]/Tau[j-1]) *
          exp(-Div$MaturityTime[i]/Tau[j-1])
    }
  }
}
```

This program is set to produce the LSC model for 2, 3, and 4 factor models. The four factor model captures the maturity-varying present value of dividends except for year variation.

With this framework as well as the ability to code in R, you will be able to see numerous applications for better understanding stock prices and related risk management assessments.

LSC model applied to growth and discount rates (Selected Excerpts and Output)

The R code for this material is found in the subfolder of module 4.3. The test file is rather simple.

```
# LSC Model Growth and Discount Rate Test.R
…
# Identify spreadsheet containing inputs
InputFileName <- 'LSC Input File.xlsx'
#
# Core modeling inputs
#
NumberOfMaturities <- 500
# SPY: SP 500 ETF
# XLK: SP 500 Sector ETF: Technology
# XLF: SP 500 Sector ETF: Financial
# XLI: SP 500 Sector ETF: Industrial
# XLY: SP 500 Sector ETF: Consumer Discretionary
# XLB: SP 500 Sector ETF: Materials
# XLV: SP 500 Sector ETF: Healthcare
# XLU: SP 500 Sector ETF: Utilities
# XLP: SP 500 Sector ETF: Consumer Staples
# XLE: SP 500 Sector ETF: Energy
#
# Calibrate using LSC model
#
source("LSC Model Calibration.R")
source("LSC Model Calibration Plots.R")
source("LSC Model Calibration Plots V2.R")
```

Model calibration proceeds in several phases contained in the LSC Model Calibration.R file.

```
# LSC Model Calibration.R
# Import as data.frame each tab of data in input spreadsheet
# II - industry input data.frame
# MI - model inputs
II <- read.xlsx(xlsxFile = InputFileName, sheet = "Industry Inputs",
  skipEmptyRows = FALSE)
MI <- read.xlsx(xlsxFile = InputFileName, sheet = "Model Inputs",
  skipEmptyRows = FALSE)
#
# Phase 1: Calibrate LSC model based on II and MI
#
# Core model inputs
NumberOfFactors <- MI$Value[1]
N <- pmax(1,NumberOfFactors - 2)
ScalarG <- c(1:N)
ScalarDR <- c(1:N)
for(i in 1:N){
  ScalarG[i] <- MI$Value[i+1]
  ScalarDR[i] <- MI$Value[N+i+1]
}
Lg <- MI$Value[2*N+2]
Damper <- MI$Value[2*N+3]
LengthII <- length(II$Price) # Number of industries
#
# Phase 2: Solve for implied expected cash flow growth rate
#
source("LSC Model Functions.R")
inputInstrumentValue <- -99
inputGLevel = Lg
```

```
inputGSlope = -99 # Placeholer
inputGCurve1 = 0
inputDRLevel = -99
inputDRSlope = -99
inputDRCurve1 = 0
inputScalarG = ScalarG
inputScalarDR = ScalarDR
inputNumberOfYears <- NumberOfMaturities
inputDR <- -99 # Placeholder
inputInitialCF <- 1.0
inputImpliedLowerBound <- -10
inputImpliedUpperBound <- 10
#  Input matrix
#  LSCInputData - eventually generalize
LSCModelInputs <- list(inputGLevel, inputGSlope, inputGCurve1,
  inputDRLevel, inputDRSlope, inputDRCurve1,
  inputScalarG, inputScalarDR,
  inputNumberOfYears, inputDR, inputInitialCF,
  inputImpliedLowerBound, inputImpliedUpperBound)
names(LSCModelInputs) <- c("LSCGLevel", "LSCGSlope", "LSCGCurve1",
  "LSCDRLevel", "LSCDRSlope", "LSCDRCurve1",
  "LSCScalarG", "LSCScalarDR",
  "LSCNumberOfYears", "LSCDR", "LSCInitialCF",
  "ImpliedLowerBound", "ImpliedUpperBound")
II$CF <- II$Price*II$DY # Initial dollar dividend amount
II$PCF <- II$Price/II$CF
for(i in 1:LengthII){
  inputInstrumentValue <- II$Price[i]
  LSCModelInputs$LSCInitialCF <- II$CF[i]
  LSCModelInputs$LSCDR <- II$DR[i]
  # LSCModelInputs$LSCGLevel = Lg
  II$GSlope[i] = ImpliedLSCGSlope(LSCModelInputs, inputInstrumentValue)
  LSCModelInputs$LSCGSlope <- II$GSlope[i]
  II$ILSCGValue[i] <- LSCInstrumentValueG(LSCModelInputs)
  II$GLevel[i] <- LSCModelInputs$LSCGLevel
  II$ScalarG[i] <- LSCModelInputs$LSCScalarG
  II$ScalarDR[i] <- LSCModelInputs$LSCScalarDR
}
#
# Phase 3: Now solve for implied forward DR slope
#
# Compute Lf for each industry based on damper
#
AvPCF <- mean(II$PCF)
II$PCFL <- II$PCF + Damper*(AvPCF - II$PCF)
II$DRLevel <- log(1 + 1/II$PCF) + Lg
for(i in 1:LengthII){
  LSCModelInputs$LSCGLevel = II$GLevel[i]
  LSCModelInputs$LSCGSlope = II$GSlope[i]
  LSCModelInputs$LSCDRLevel = II$DRLevel[i]
  inputInstrumentValue <- II$Price[i]
  II$DRSlope[i] = ImpliedLSCDRSlope(LSCModelInputs, inputInstrumentValue)
  LSCModelInputs$LSCDRSlope <- II$DRSlope[i]
  II$ILSCDRValue[i] <- LSCInstrumentValueGDR(LSCModelInputs)
}
# Phase 4: Export
II$PCF <- round(II$PCF, 4)
II$DR <- round(II$DR*100, 4)
II$GSlope <- round(II$GSlope*100, 4)
II$ILSCGValue <- round(II$ILSCGValue, 4)
II$GLevel <- round(II$GLevel*100, 4)
II$DRSlope <- round(II$DRSlope*100, 4)
II$ILSCDRValue <- round(II$ILSCDRValue, 4)
II$DRLevel <- round(II$DRLevel*100, 4)
```

```
write.xlsx(II, "LSC Output File.xlsx", sheetName = "Industry Outputs")
```

The complete output file is given in Figure 4R.3.3 below.

Figure 4R.3.3 LSC model application to sector ETFs

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	**Industry**	**Ticker**	**Price**	**DY**	**DR**	**CF**	**PCF**	**GSlope**	**ILSCGValue**	**GLevel**	**ScalarG**	**ScalarDR**	**PCFL**	**DRLevel**	**DRSlope**	**ILSCDRValue**
2	Broad Market	SPY	$311.17	1.90%	8.00	$5.9122	$52.63	9.6179	$311.17	4	3	10	$50.0815	5.8822	-2.3816	$311.17
3	Technology	XLK	$105.09	1.18%	9.00	$1.2401	$84.75	20.1333	$105.09	4	3	10	$66.1385	5.1731	9.8600	$105.09
4	Financial	XLF	$22.99	2.71%	10.00	$0.6230	$36.90	12.0371	$22.99	4	3	10	$42.2158	6.6739	13.0265	$22.99
5	Industrials	XLI	$68.70	2.30%	7.80	$1.5801	$43.48	6.1766	$68.70	4	3	10	$45.5048	6.2739	2.5642	$68.70
6	Consumer Discretionary	XLY	$129.00	1.24%	8.20	$1.5996	$80.65	16.0797	$129.00	4	3	10	$64.0882	5.2324	6.7896	$129.00
7	Materials	XLB	$56.10	2.21%	12.00	$1.2398	$45.25	21.5877	$56.10	4	3	10	$46.3901	6.1859	11.6216	$56.10
8	Healthcare	XLV	$101.04	2.33%	9.80	$2.3542	$42.92	13.6402	$101.04	4	3	10	$45.2249	6.3033	3.7498	$101.04
9	Utilities	XLU	$57.61	3.52%	7.50	$2.0279	$28.41	-1.0676	$57.61	4	3	10	$37.9702	7.4595	-2.7163	$57.61
10	Consumer Stables	XLP	$58.92	2.83%	6.50	$1.6674	$35.34	-2.5747	$58.92	4	3	10	$41.4335	6.7907	-1.9987	$58.92
11	Energy	XLE	$37.11	4.00%	14.00	$1.4844	$25.00	17.0437	$37.11	4	3	10	$36.2657	7.9221	8.9215	$37.11

The remainder of the program produces numerous plots.