concordance=TRUE

Performance Analysis Measures

Mark L Labovitz

March 8, 2014

Abstract

Applying The CAPM to Performance Measurement Using Quandl data

Uses computations as interpreted from Chapter 7 of 2013 FRM Level I Manual.

Computational Focus:

1. Treynor Ratio;
2. Sharpe Ratio;
3. Jensen's Alpha;
4. Tracking Error;
5. Information Ratio;
6. Sortino Ratio.

Ancillary computation (necessary but not focus of Chapter):

1. Beta;
2. Downside Deviation.

Assumptions:

1. Computations use the PerformanceAnalytics Library.
2. There are alternative means of computing the measure which relate to such elements as:

a. The use of as recorded or excess returns;

b. The method annualization: arithmetic or geometric;

c. The order of return adjustment and averaging;

Relevant Abbreviations:

IP == Investment Portfolio;

MP == Market/Benchmark Portfolio;

RF == Risk Free Rate;

sp == Sub Period (the period of the data, e.g. day, week, month).

Contents

1 Setting Up The Analysis

1.1 Establishing RunTime Parameters

```
# Store results
Measures = rep(NULL, 4)

# If equalWts is true portfolio weights will be 1/numberOfSecurities
equalWts = TRUE

randWts = TRUE
```

```r
if (equalWts) randWts = FALSE

# Number of securities to use is
numberOfSecurities = 3

# Security Selection Method, 'randomSeq' or 'serialSeq'
secSel = "serialSeq"

# type of chaining in forming the aggregate return If geometric is TRUE use
# geometric chaining if False use arithemetic chaining
geometricB = TRUE

# Definitioned/Constant variables
relation = c("Less Than", "Equal To", "Greater Than")

# Some of run time parameters Should be amongst the input for shiny Minimum
# Acceptable Return for Sortino Ratio or Downside Deviation defaults to 0
MAR_set = 0

# Denominator for Sortino Ratio or Downside Deviation one of 'full' or
# 'subset', indicating whether to use the length of the full series or the
# length of the subset of the series below the MAR as the denominator,
# defaults to 'full'
denom = "full"

# Portfolio name used in output
PortName = "Portfolio"

# Market name used in output
MrktName = "Market"

# Market Name in data set
mrktDataName = "market"
```

```r
# Risk free rate name used in output
RFName = "Risk_Free_Rate"

# Risk Free Name in data set
rfDataName = "t90"

# Number time periods for year for data set
freQ = 12

# Names of Measures column
Measures = rbind(Measures, c("Series Name", PortName, MrktName, RFName))
colnames(Measures) = c("Measure", "Portfolio", "Market", "RiskFree")
```

1.2 Echo Parameters

Echo print run-time parameters

```r
cat("\n Random Weights: ", randWts, "\n")

##
##  Random Weights:  FALSE

cat("\n Number of securities in portfolio = ", numberOfSecurities, "\n")

##
##  Number of securities in portfolio =  3

cat("\n Security selection method: ", secSel, "\n")

##
##  Security selection method:  serialSeq

cat("\n Geometric annualization (false means arithmetic): ", geometricB, "\n")

##
##  Geometric annualization (false means arithmetic):  TRUE

cat("\n Minimum Acceptable Return for Sortino Ratio or Downside Deviation: ",
    MAR_set, "\n")
```

```
##
##  Minimum Acceptable Return for Sortino Ratio or Downside Deviation:  0

cat("\n Denominator for downside deviation (Full count or Subset < MAR_set): ",
    denom, "\n")

##
##  Denominator for downside deviation (Full count or Subset < MAR_set):  full

namesUsed = c(PortName, MrktName, RFName)
names(namesUsed) = c("Portfolio_Name", "Market/BenchMark_Name", "RiskFreeRate_name")
cat("\n Names of Portfolio, Market and RiskFree Series used in output")

##
##  Names of Portfolio, Market and RiskFree Series used in output

print(namesUsed)

##        Portfolio_Name Market/BenchMark_Name     RiskFreeRate_name
##           "Portfolio"              "Market"      "Risk_Free_Rate"

cat("\n Number of data units in a year: ", freQ, "\n")

##
##  Number of data units in a year:  12
```

1.3 Extracting and Organizing Data

```
# 'Load the GARPFRM package and the CAPM dataset.
suppressMessages(library(GARPFRM))
options(digits = 5)
data(crsp.short)

# Make the assumption that the last two time series are the Market and RF
# series Tickers/Names of time series in the test data set
cat("\n List column headers predominantly tickers/name of securities")
```

```
##
##  List column headers predominantly tickers/name of securities

colnames(largecap.ts)

##  [1] "AMAT"   "AMGN"   "CAT"    "DD"     "G"      "GENZ"   "GM"
##  [8] "HON"    "KR"     "LLTC"   "MSFT"   "ORCL"   "PG"     "PHA"
## [15] "SO"     "TXN"    "UTX"    "WM"     "WYE"    "YHOO"   "market"
## [22] "t90"
```

Summarize the first and last data values corresponding to the first 5 dates for the first 5 returns, followed by number of periods in time series

```
head(largecap.ts)

##                  AMAT       AMGN       CAT          DD          G       GENZ
## 1997-01-31  0.373913  0.036782 0.0368771  0.1646746  0.063408  0.287356
## 1997-02-28  0.025316  0.084257 0.0080515 -0.0164652 -0.040909 -0.080357
## 1997-03-31 -0.083951 -0.085890 0.0255591 -0.0116550 -0.082148 -0.126214
## 1997-04-30  0.183288  0.053691 0.1140187  0.0011792  0.173356  0.027778
## 1997-05-30  0.189066  0.135881 0.0969101  0.0294935  0.045588  0.032432
## 1997-06-30  0.085249 -0.130841 0.0998720  0.1576525  0.066104  0.162304
##                    GM        HON         KR        LLTC       MSFT        ORCL
## 1997-01-31  0.058296  0.050373  0.026882  0.1122507  0.234493 -0.0688623
## 1997-02-28 -0.010593  0.030337  0.109948 -0.0666667 -0.044118  0.0096463
## 1997-03-31 -0.043197 -0.013841 -0.037736 -0.0274725 -0.059615 -0.0175159
## 1997-04-30  0.045147  0.014035  0.078431  0.1367232  0.325153  0.0307942
## 1997-05-30  0.000000  0.065882 -0.059091 -0.0024876  0.020576  0.1729560
## 1997-06-30 -0.028322  0.094463  0.120773  0.0324190  0.019153  0.0804290
##                    PG        PHA          SO         TXN         UTX         WM
## 1997-01-31  0.074983 -0.028939 -0.0187845  0.2294118  0.0528302  0.259740
## 1997-02-28  0.038919 -0.032450 -0.0057143 -0.0159490  0.0832975 -0.026467
## 1997-03-31 -0.044745  0.051546 -0.0287356 -0.0269692  0.0066445 -0.086288
## 1997-04-30  0.099782  0.114379 -0.0355030  0.1919867 -0.0016502  0.027374
## 1997-05-30  0.096421  0.036012  0.0527607  0.0056022  0.0669091  0.126582
## 1997-06-30  0.024479 -0.021307  0.0355030 -0.0614763  0.0326594  0.074157
##                     WYE       YHOO    market       t90
## 1997-01-31  0.0810234  0.992647  0.053032 0.004517
## 1997-02-28  0.0163314 -0.107011 -0.000885 0.003902
## 1997-03-31 -0.0625000 -0.070248 -0.044399 0.004282
## 1997-04-30  0.1020833  0.213333  0.042472 0.004844
## 1997-05-30  0.1555388 -0.054945  0.071262 0.004837
## 1997-06-30  0.0065789  0.093023  0.044201 0.004228
```

```
tail(largecap.ts)
```

```
##                  AMAT      AMGN        CAT        DD         G      GENZ
## 2001-07-31 -0.0659877  0.033454  0.1078921 -0.112355 -0.033029 -0.081967
## 2001-08-31 -0.0604012  0.025355 -0.0925590 -0.035030  0.099749  0.011429
## 2001-09-28 -0.3399861 -0.086003 -0.1040000 -0.084208 -0.027732 -0.198093
## 2001-10-31  0.1993671 -0.033180  0.0060268  0.065832  0.048742  0.187803
## 2001-11-30  0.1650543  0.169131  0.0603756  0.117529  0.051785  0.012419
## 2001-12-31  0.0090588 -0.150384  0.1018558 -0.041272  0.021407  0.095936
##                   GM       HON         KR      LLTC      MSFT      ORCL
## 2001-07-31 -0.011655  0.059088  0.0544000 -0.014473 -0.093288 -0.048421
## 2001-08-31 -0.131289  0.015663  0.0098634 -0.056448 -0.138087 -0.324668
## 2001-09-28 -0.216438 -0.291465 -0.0743802 -0.201558 -0.103068  0.030303
## 2001-10-31 -0.036830  0.119318 -0.0073052  0.184146  0.136408  0.077901
## 2001-11-30  0.214908  0.127834  0.0351595  0.057474  0.104213  0.034661
## 2001-12-31 -0.022133  0.020519 -0.1757503 -0.048501  0.031771 -0.015681
##                   PG        PHA         SO       TXN       UTX        WM
## 2001-07-31  0.119122 -0.0260066  0.0107527  0.082171  0.001911  0.085220
## 2001-08-31  0.044072 -0.1125056  0.0002130 -0.040580 -0.065054 -0.076012
## 2001-09-28 -0.018341  0.0242425  0.0349590 -0.245317 -0.320175  0.027778
## 2001-10-31  0.018821  0.0023422 -0.0033361  0.121347  0.158925 -0.209200
## 2001-11-30  0.049878  0.0957552 -0.0341004  0.145052  0.121266  0.036105
## 2001-12-31  0.021559 -0.0394144  0.1142857 -0.126365  0.073588  0.045396
##                  WYE     YHOO    market      t90
## 2001-07-31  0.026553 -0.11856 -0.017827 0.003305
## 2001-08-31 -0.067651 -0.32690 -0.059049 0.003284
## 2001-09-28  0.040179 -0.25717 -0.090733 0.004224
## 2001-10-31 -0.041545  0.23496  0.027219 0.002545
## 2001-11-30  0.080602  0.43107  0.078237 0.002106
## 2001-12-31  0.020965  0.13937  0.017569 0.001592

nRow = nrow(largecap.ts)
nRow

## [1] 60
```

1.4 Extract Market (Benchmark) And Risk Free Series

Determine pointer to column with the market returns and pointer to column of Risk Free Ratw

```
ptrMkt = which(colnames(largecap.ts) == mrktDataName, arr.ind = TRUE)
ptrMkt

## [1] 21

ptrRF = which(colnames(largecap.ts) == rfDataName, arr.ind = TRUE)
ptrRF

## [1] 22

# Market returns
R.market <- largecap.ts[, ptrMkt]


# risk free rate
rf <- largecap.ts[, ptrRF]
##
```

1.5 Generate Portforlio For Further Analysis

Generate Portfolio weights based upon selected method and create portfolio.

```
# Number of time series in data set
nSec = ncol(largecap.ts)


if (numberOfSecurities > (nSec - 2)) numberOfSecurities = nSec - 2
cat("\n Number of Securities being used in portfolio = ", numberOfSecurities,
    "\n")
```

```
##
##  Number of Securities being used in portfolio =  3

# Security position numbers of securities in portfolio
secNbrs = seq(1, numberOfSecurities)
if (secSel == "randomSeq") secNbrs = sample.int(nSec, size = numberOfSecurities,
    replace = FALSE)

cat("\n Selected Security position indicies")

##
##  Selected Security position indicies

print(secNbrs)

## [1] 1 2 3

# Created random wts from the non-market non RF series if selected
if (randWts) {
    coefs = runif(numberOfSecurities)
    alphas = coefs/sum(coefs)
    sum(alphas)
}

# Generate portfolio from selected securities and associated weigts
if (equalWts) {
    R.portfolio <- Return.portfolio(largecap.ts[, secNbrs], geometric = geometricB)
} else {
    R.portfolio = Return.portfolio(largecap.ts[, secNbrs], weights = alphas,
        geometric = geometricB)
}

## Warning:  weighting vector is null, calulating an equal weighted portfolio

# Print names of securities in portfolio
cat("\n Securities in Portfolio")

##
##  Securities in Portfolio
```

```
print(colnames(largecap.ts)[secNbrs])

## [1] "AMAT" "AMGN" "CAT"

colnames(R.portfolio) = "PortRets"

# Compute initial ancillary statistics and set them in the summary matrix
salient = matrix(data = c("Mean", mean(R.portfolio), mean(R.market), mean(rf),
    "Stdev", sd(R.portfolio), sd(R.market), sd(rf), "NbrSecuritiesInPort", numberOfSecurities,
    "", "", "AnnuallyFrequency", rep(freQ, 3), "Number of Samples", rep(nRow,
        3)), ncol = 4, byrow = TRUE)

Measures = rbind(Measures, salient)

# Results used later in computation Mean of risk free rate over sampling
# time series
meanRF = mean(rf)

# Precompute excess returns
xCessP1 = R.portfolio - rf
xCessM1 = R.market - rf

# Precompute relative return [RR] = (Investment Portfolio [IP] - Market
# Portfolio [MP]) and average relative return [ARR] = mean(RR)

RR = R.portfolio - R.market
ARR = mean(RR)
```

2 Treynor Ratio

(Add more commentary) Where TR denotes the Treynor Ratio and IP and MP denote the Investment Portfolio and Market Portfolio

2.1 Beta

Compute Beta using excess returns.

```
# Compute Beta from excess returns.
betaRF = cov((R.portfolio[, 1] - rf), (R.market - rf))/var((R.market - rf))

# Output value of Beta
cat("\n Beta =", betaRF, ", between the excess portfolio represented by ", PortName,
    " and the excess market represented by ", MrktName, " \n")

##
##  Beta = 1.4644 , between the excess portfolio represented by  Portfolio  and the excess market

Measures = rbind(Measures, c("Beta", betaRF, "", ""))
```

2.2 Treynor Ratio

Compute the Traynor Ratio.

```
# Treynor ratio for portfolio and market Recall that Beta for the market is
# 1
tr = TreynorRatio(R.portfolio, R.market, rf, freQ)
trMarket = TreynorRatio(R.market, R.market, rf, freQ)

# Output salient results from Treynor Ratio computation
cat("\n PA computed for ", PortName, " Treynor Ratio = ", tr, "\n")

##
##  PA computed for  Portfolio  Treynor Ratio =  0.14602

cat("\n PA computed for ", MrktName, " Treynor Ratio = ", trMarket, "\n")

##
##  PA computed for  Market  Treynor Ratio =  0.041322

Measures = rbind(Measures, c("Treynor Ratio", tr, trMarket, ""))
```

As per the Chapter comparing the Treynor Ratio for the investment portfolio versus the Treynor Ratio for the market portfolio allows a check on whether or not the investment portfolio is being sufficiently rewarded for its level of risk, i.e. the TR(IP) greater than or equal TR(MP).

```r
# Compare Treynor Ratio for market returns against Treynor Ratio for
# portfolio returns
TR_Mrkt2Port = relation[sign(trMarket - tr) + 2]
cat("\n Treynor Ratio of ", MrktName, " is ", TR_Mrkt2Port, " Treynor Ratio of ",
    PortName, "\n")

##
##  Treynor Ratio of  Market  is  Less Than  Treynor Ratio of  Portfolio
```

3 Sharpe Ratio

Developed by Nobel Laureate William F. Sharpe, the Sharpe ratio is a risk-adjusted measure of performance, also known as the reward-to-volatility or reward per unit of risk ratio. It is calculated as the average sp excess return divided by the standard deviation of sp excess returns over a period of interest. The excess return is the difference between the IP return and the RF return for an sp. A higher Sharpe ratio means better fund performance relative to the risk-free rate on a risk-adjusted basis.

```r
# Compute Sharpe and annualized Sharpe Ratio
shr = SharpeRatio(R.portfolio, rf, FUN = "StdDev")
shrAnn = SharpeRatio.annualized(R.portfolio, rf, scale = freQ, geometric = TRUE)

# Output salient results for Sharpe Ratio
cat("\n PA computed ", PortName, " Sharpe Ratio = ", shr, "\n")

##
##  PA computed  Portfolio  Sharpe Ratio =  0.20392

cat("\n PA computed ", PortName, " Annualized Sharpe Ratio = ", shrAnn, "\n")

##
##  PA computed  Portfolio  Annualized Sharpe Ratio =  0.5734

Measures = rbind(Measures, matrix(c("Sharpe Ratio", shr, "", "", "Sharpe Ratio Annualized",
    shrAnn, "", ""), ncol = 4, byrow = TRUE))
```

4 Jensen's Alpha

(Add more commentary)

```
# Compute Jensen's Alpha

# Produces one value
jaStatic = CAPM.jensenAlpha(R.portfolio, R.market, meanRF)

# Produces multiple values
jaTV = CAPM.jensenAlpha(R.portfolio, R.market, rf)

# Regression computation and salient results
lm_ja1.fit = lm(xCessP1 ~ xCessM1)
values1 = summary(lm_ja1.fit)
values1
```

```
##
## Call:
## lm(formula = xCessP1 ~ xCessM1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.15459 -0.04258  0.00123  0.04399  0.20033
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.01484    0.00952    1.56     0.12
## xCessM1      1.46440    0.17653    8.30    2e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0734 on 58 degrees of freedom
## Multiple R-squared:  0.543,Adjusted R-squared:  0.535
## F-statistic: 68.8 on 1 and 58 DF,  p-value: 1.96e-11
```

```
analT1 = values1[[4]]
```

```r
# Annualize Jensen's Alpha
jaHandA = (1 + (mean(R.portfolio) - meanRF - betaRF * (mean(R.market) - meanRF)))^freQ -
    1

# Use Delta Method to approximate standard error of annualized Jensen's
# Alpha
deltaCoef = (freQ * (1 + analT1[1, 1])^(freQ - 1))^2

# Output salient results for Jensen's Alpha
cat("\n PA computed ", PortName, " Static Jensen's Alpha = ", jaStatic, "\n")

##
##  PA computed  Portfolio  Static Jensen's Alpha =  0.13828

cat("\n Hand computed Annualized ", PortName, " Jensen Alpha = ", jaHandA, "\n")

##
##  Hand computed Annualized  Portfolio  Jensen Alpha =  0.19336

cat("\n Delta Method Coefficient Value = ", deltaCoef, "\n")

##
##  Delta Method Coefficient Value =  199.12

# t stat and pvalue under H0: alpha = 0 against HA: alpha != 0
tStat = jaHandA/(analT1[1, 2] * deltaCoef^0.5)
pValue = 2 * (1 - pt(tStat, nRow - 2))
cat("\n H0: alpha = 0, HA: alpha != 0  p-value: ", pValue, "\n")

##
##  H0: alpha = 0, HA: alpha != 0  p-value:  0.15542

Measures = rbind(Measures, matrix(c("Jensen's Alpha", jaStatic, "", "", "Jensen's Alpha Annualize
    jaHandA, "", "", "Jensen's Alpha P Value", pValue, "", ""), ncol = 4, byrow = TRUE))
```

5 Tracking Error

(Add more commentary)

```r
# Compute Tracking Error

te = TrackingError(R.portfolio, R.market, scale = freQ)

# Output salient results from Tracking Error computation
cat("\n PA computed ", PortName, " Tracking Error = ", te, "\n")
```

```
##
##  PA computed  Portfolio  Tracking Error =  0.26687
```

```r
Measures = rbind(Measures, c("Tracking Error", te, "", ""))
```

6 Information Ratio

(Add more commentary)

```r
# Crompute Information Ratio Definition PA, InformationRatio =
# ActivePremium/TrackingError Active Premium = Investment's annualized
# return - Benchmark's annualized return
ir = InformationRatio(R.portfolio, R.market, scale = freQ)

# Output salient results from Information Ratio computation
cat("\n PA computed ", PortName, " Information Ratio = ", ir, "\n")
```

```
##
##  PA computed  Portfolio  Information Ratio =  0.67837
```

```r
Measures = rbind(Measures, c("Information Ratio", ir, "", ""))
```

7 Downside Deviation and Sortino Ratio

(Add more commentary)

7.1 Downside Deviation

Downside Deviation is a measure of risk which just captures the downside or returns below a minimal acceptible return or MAR. Consequently the computation only includes as non-zero those returns less than the MAR. MAR in the present case is set as single value, in other calculations it itself is a series, often the market/benchmark performance. As downside deviation increases it means higher risk on the downside relative to the MAR. Beyond the MAR, an important parametric consideration is the decision to normalize the deviations by the total number of values in the time series or just the number of deviations which are not equal to zero (in either case the denominator count is maybe reduced by 1)

```
# Compute Downside Deviation

# Set value of denominator for Downside Deviation from the parameter set
# earlier
denomVal = nRow
if (denom != "full") {
    denomVal = sum(ifelse((R.portfolio - MAR_set) < 0, 1, 0))
}
cat("\n Denominator parameter is set to: ", denom, " which results in a denominator value of ",
    denomVal, "\n")

##
##  Denominator parameter is set to:  full  which results in a denominator value of  60

cat("\n MAR == Minimal Acceptable Return, number of MAR values = ", length(MAR_set))

##
##  MAR == Minimal Acceptable Return, number of MAR values =  1

head(as.vector(MAR_set))

## [1] 0

if (length(MAR_set) > 5) tail(as.vector(MAR_set))

# PA computation of Downside Deviation
dwn = DownsideDeviation(R.portfolio, MAR = MAR_set, method = denom)
```

```
# Output salient computation of Downside Deviation
cat("\n PA computed ", PortName, "  Downside Deviation = ", dwn, "\n")

##
##  PA computed  Portfolio   Downside Deviation =  0.061658

Measures = rbind(Measures, c("Downside Deviation", dwn, "", ""))
Measures = rbind(Measures, c("Denominator DD", denomVal, "", ""))
```

7.2 Sortino Ratio

(Add more commentary)

```
# Compute Sortino Ratio
sr = SortinoRatio(R.portfolio, MAR = MAR_set, weights = NULL)

# Output salient computations of Sortino Ratio
cat("\n PA computed ", PortName, "  Sortino Ratio = ", sr, "\n")

##
##  PA computed  Portfolio   Sortino Ratio =  0.42558

Measures = rbind(Measures, c("Sortino Ratio", sr, "", ""))
```

8 Summary of Performance Measures

Summary of values and statistics computed from input IP, MP and risk free rate series.

```
Measures.few = cbind(Measures[, 1], substr(Measures[, 2:4], 1, 8))
Measures.few = rbind(Measures[1, ], Measures.few[-1, ])

Measures.df = as.data.frame(Measures.few, stringsAsFactors = FALSE)
print(Measures.df)

##                         Measure Portfolio   Market       RiskFree
## 1                   Series Name Portfolio   Market Risk_Free_Rate
## 2                          Mean  0.026240 0.009144       0.004287
## 3                       Stdev  0.107653 0.054107       0.000857
## 4        NbrSecuritiesInPort         3
## 5          AnnuallyFrequency        12       12             12
## 6          Number of Samples        60       60             60
## 7                        Beta  1.464402
## 8              Treynor Ratio  0.146020 0.041322
## 9               Sharpe Ratio  0.203923
## 10   Sharpe Ratio Annualized  0.573395
## 11            Jensen's Alpha  0.138276
## 12 Jensen's Alpha Annualized  0.193362
## 13    Jensen's Alpha P Value  0.155417
## 14            Tracking Error  0.266872
## 15         Information Ratio  0.678369
## 16        Downside Deviation  0.061658
## 17            Denominator DD        60
## 18             Sortino Ratio  0.425579
```