Contents

Fundamental Factor Models in FactorAnalytics

R. Douglas Martin, Mido Shammaa and Avinash Navoor

February 23, 2021

1 Introduction

The overarching long-term goal of the fundamental factor model (Ffm) development in the FactorAnalytics packages is to replicate a large proportion of the non-proprietary models and model fitting and analysis methodology that is contained in commercial portfolio construction and risk management products such as MSCI Barra, Axioma, Northfield, etc. Furthermore our goals include the implementation cutting edge methods to support portfolio construction and risk management that are not much available in commercial products, such as global optimization, unequal histories and other missing data handling, highly robust covariance matrix estimators and their application to multivariate exposures and returns outliers, optimal bias robust regression, factor-model Monte Carlo, new methods for handling serial correlation that improve upon traditional HAC methods, etc.

We note that a challenge for developing an effective fundamental factor model (FFM) in factorAnalytics is that of providing an adequate set of factor exposures (also called firm characteristics, or more simply factor data). For the current development version of the FFM, the factor exposures described in Section 2 were obtained from Compustat via the University of Washington's license with Wharton Research Data Services, and the matched stock prices were obtained from Yahoo. However, in the upcoming next development phase of the FFM we anticipate to be able to include a time series of a dozen S&P CapitalIQ "scores" (simple exposures and various ratios), corresponding to a cross-section of 300 stock returns from 1990 through 2015 that are well-balanced across market-cap ranges. This will allow users a deep understanding the general statistical behavior of such fundamental factors (mean, volatility, non-stationarity, predictability), and to use the factors in meaningful "non-toy" studies of the use of FFM's in portfolio construction and risk management.

** this section will be revised in the near future, along with the entire document, RDM **

2 The Equity Fundamental Factor Model and Data Sets

The general mathematical form of equity fundamental factor model (FFM) implemented in factor-Analytics is

$$\mathbf{r}_t = \alpha_t \cdot \mathbf{1} + \mathbf{B}_{t-1}\mathbf{f}_t + \boldsymbol{\varepsilon}_t, \quad t = 1, 2, \cdots, T \tag{1}$$

where equity returns vector $\mathbf{r}_t$ and the vectors $\mathbf{1}$ and $\boldsymbol{\varepsilon}_t$ are $N \times 1$ vectors, $\mathbf{B}_{t-1}$ is an $N \times K$ exposures (risk factors) matrix, and $\mathbf{f}_t$ is a $K \times 1$ vector of random factor returns. It is assumed that the $\boldsymbol{\varepsilon}_t$ have zero mean with diagonal covariance matrix $\mathbf{D}_\mathrm{t} = diag(\sigma^2_{\epsilon t,1}, \sigma^2_{\epsilon t,2}, \cdots, \sigma^2_{\epsilon t,N})$, and are uncorrelated with the $\mathbf{f}_t$. The exposures matrix will in general have three groups of columns corresponding to: (1) style risk factors such as earnings-to-price (EP), firm size defined as the logarithm of market capitalization in \$M, and book-to-price (BP), etc., a (2) sector (or industry) factors, and (3) country factors. These groups are ordered from left to right in $\mathbf{B}_{t-1}$.

The following U.S. stock returns and data sets are included in factorAnalytics for now.

`factorDataSetDjia`: Monthly returns of 22 stocks in the DJIA from May 2004 to March 2013, with 5 corresponding style factors (MKTCAP, ENTVAL, P2B, EV2S, SIZE) and a sector factor with 8 sectors.

`factorDataSetDjia5Yrs`: A five-year segment of `factorDataSetDjia` from January 2008 to December 2012.

`wtsDjiaGmvLo`: Weight vector of dimension 22 containing the weights of a long-only global minimum variance (GMV) portfolio for the 22 stocks in the `factorDataSetDjia5Yrs` data set. The weight vector was obtained using PortfolioAnaltyics with the usual sample covariance matrix based on the 5 years of returns in `factorDataSetDjia5Yrs`.

You can view the full details of any of these data sets with the `help` function. For example use of

```
help(factorDataSetDjia5Yrs)
```

results in a display of the help file in the Rstudio Help window.

You can load `factorDataSetDjia5Yrs` , give it the shorter name `dataDjia.5Yr`, and view the list components of the data for the first two stocks in January 2008 with:

```
data("factorDataSetDjia5Yrs")
dataDjia5Yr = factorDataSetDjia5Yrs
head(dataDjia5Yr, 2)

##         DATE PERMNO GVKEY    CUSIP TICKER      NAME RETURN.OLD    RETURN
## 969 Jan 2008  24643  1356 13817101     AA ALCOA INC  -0.094665 -0.094665
## 970 Jan 2008  19561  2285 97023105     BA BOEING CO  -0.048937 -0.048937
##     RETURN.DIFF GSECTOR SECTORNAMES SECTOR   MKTCAP   ENTVAL      P2B      EV2S
## 969           0      15   Materials MATRLS 29.38908 37.50708 1.865026 1.3399214
## 970           0      20 Industrials INDUST 53.72585 52.36785 5.926081 0.8187595
##         SIZE
## 969 10.28838
## 970 10.89165
```

3 Fitting a Fundamental Factor Model

A FFM is generally fit by a two-step method, using least squares or robust regression methods in the first step, and using weighted least squares or weighted robust regression in the second step where the weights are computed in the first step. The weights are computed by estimating the error variances in $\mathbf{D}_t$ with an EWMA or GARCH model, or by using market capitalization weights. The resulting model fit is

$$\mathbf{r}_t = \hat{\alpha}_t \cdot \mathbf{1} + \mathbf{B}_{t-1}\hat{\mathbf{f}}_t + \hat{\boldsymbol{\varepsilon}}_t, \quad t = 1, 2, \cdots, T. \tag{2}$$

We use the `factorDataSetDjia5Yrs` to illustrate how such a model is fit. First, a model specification is set using the function `specFfm` which creates an object that contains all the information necessary to fit the FFM. We start by identifying which columns are the asset id's, dates, returns, weights and exposures in the data.frame. We then select whether we want an intercept to be included in the regression and whether robust statistics are to be used.

Then depending on the type of analysis you can lag the exposures, standardize the exposures and returns before running the fit and extracting the results. A key feature is the option to "retrofit" the object to the old class structure using the convert function so as to allow the new code to use the reporting facilities of the current structure.

```
asset.var = "TICKER"
ret.var = "RETURN"
date.var = "DATE"
exposure.vars = c("SECTOR", "SIZE", "P2B", "EV2S")
spec1 <- specFfm(data = dataDjia5Yr, asset.var = asset.var,
    ret.var = ret.var, date.var = date.var, exposure.vars = exposure.vars,
    weight.var = NULL, addIntercept = T, rob.stats = FALSE)
# lag the exposures
spec1 <- lagExposures(spec1)
# standardize the expsoures Cross-Sectionally
spec1 <- standardizeExposures(spec1, Std.Type = "CrossSection")
# fit the model
mdlFit <- fitFfmDT(spec1)

# extract regression results
results <- extractRegressionStats(spec1, fitResults = mdlFit)
# retrofit object
fitDjia5Yr <- FactorAnalytics::convert(SpecObj = spec1,
    FitObj = mdlFit, RegStatsObj = results)
names(fitDjia5Yr)

##  [1] "asset.names"     "r2"              "factor.names"    "asset.var"
##  [5] "date.var"        "ret.var"         "exposure.vars"   "exposures.num"
##  [9] "exposures.char"  "data"            "time.periods"    "factor.fit"
## [13] "beta"            "factor.returns"  "restriction.mat" "factor.cov"
## [17] "resid.var"       "residuals"       "g.cov"
```

See the `fitFfmDT` help file for definitions of what those named components of `fitDjia` contain.

Model Fit R-Squared Values

One of the most basic model fit statistics is the R-squared, and you can assess the goodness of your FFM fits by using the function `ffmRsq` to make a plot of the time series of R-squared values for each of the 60 monthly fits over the five-year window. This function computes and plots the time series of ordinary R-squared by default, but it can do that for the adjusted R-squared, or both. Use of `ffmRsq` as shown below results in both measures plotted as shown in Figure 3.

```
fmRsq(fitDjia5Yr, rsqAdj = T, plt.type = 2, isPrint = F,
    lwd = 0.7, stripText.cex = 0.8, axis.cex = 0.8)

##     Mean R-Squared Mean Adj R-Squared
##               0.78               0.54
```

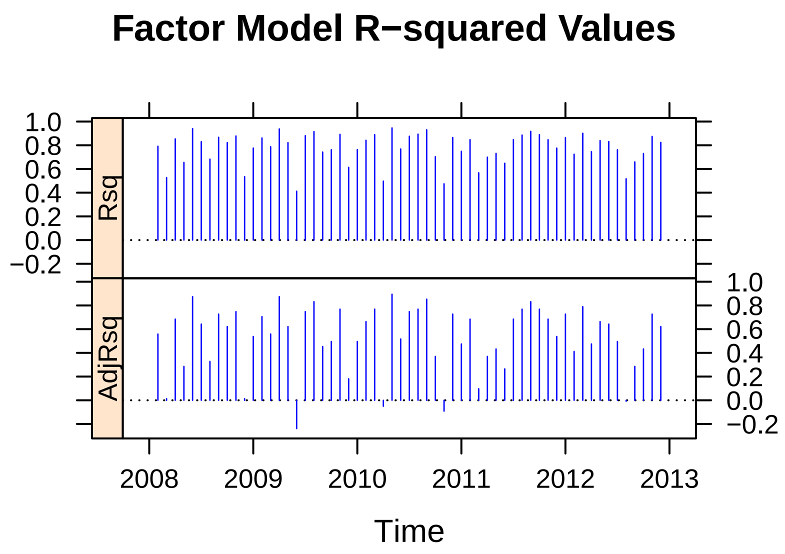

Figure 1: DJIA 5-Year Fundamental Factor Model R-Squared Values for 60 Months

It should be noted that you can print the values by using the `fmRsq` optional argument choice `isPrint = T`.

Model Fit Variance Inflation Factors

When your model includes continuous style factor variables the function, `ffmRsq` also allows you to compute and display the time series of *variance inflation factors* (VIF's). These can help you determine whether or not there are any regression collinearity problems. See Figure 3.

```
vif(fitDjia5Yr, isPlot = T, isPrint = F, lwd = 0.7,
    stripText.cex = 0.8, axis.cex = 0.8)

## $Mean.VIF
## SIZE  P2B EV2S
## 1.14 1.09 1.15
```

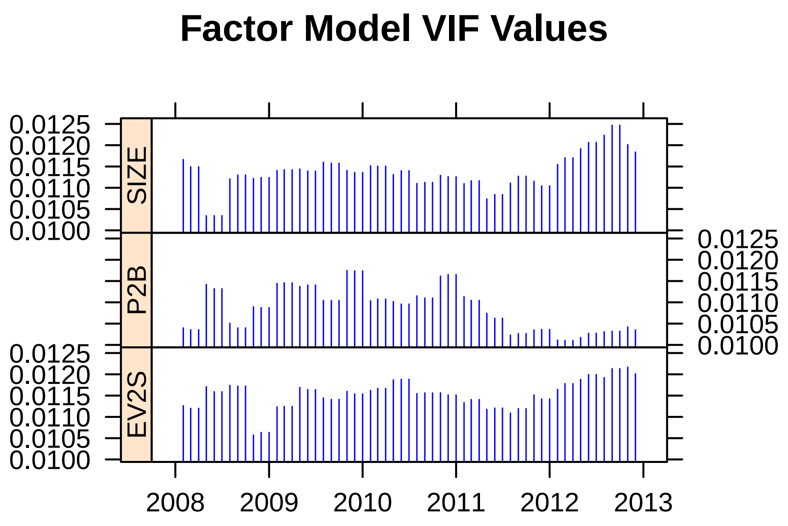

Figure 2: DJIA 5-Year Fundamental Factor Model Inflation Factors for 60 Months

Model Fit t-Statistics

Plots of the time series of t-statistics for the factors in an FFM fitted model, with horizontal dashed lines provided to judge whether or not a factor is significant at the 5% level, may be obtained with the function `ffmTstats`. See Figure 3.

```r
fmTstats(fitDjia5Yr, whichPlot = "tStats", color = "blue",
    lwd = 0.7, layout = c(3, 4), stripText.cex = 0.8,
    axis.cex = 0.8)
```

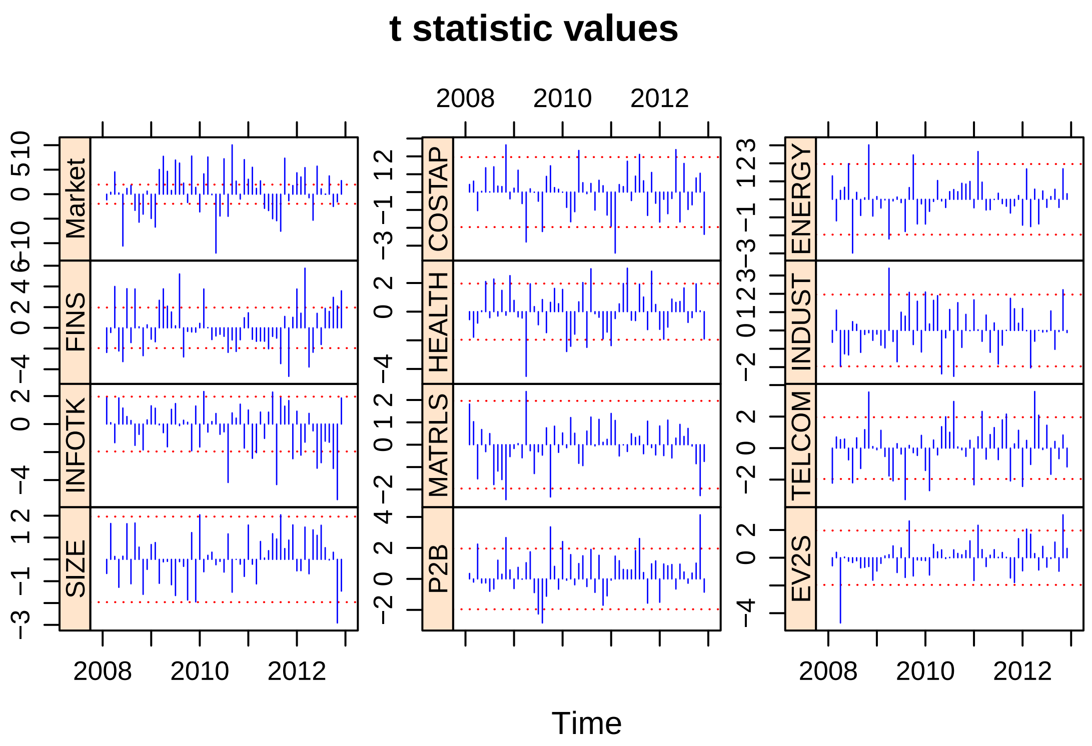

Figure 3: DJIA Data FFM t-Statistics for 3 Style Factors Plus Market and 8 Sectors

The function `fmTstats` can also compute the number of significant t-statistics, the number of positive significant t-statistics and the number of negative significant t-statistics. See Figure 4:

```r
fmTstats(fitDjia5Yr, whichPlot = "significantTstatsV",
    color = "blue", stripText.cex = 0.8, axis.cex = 0.8,
    layout = c(3, 4))
```

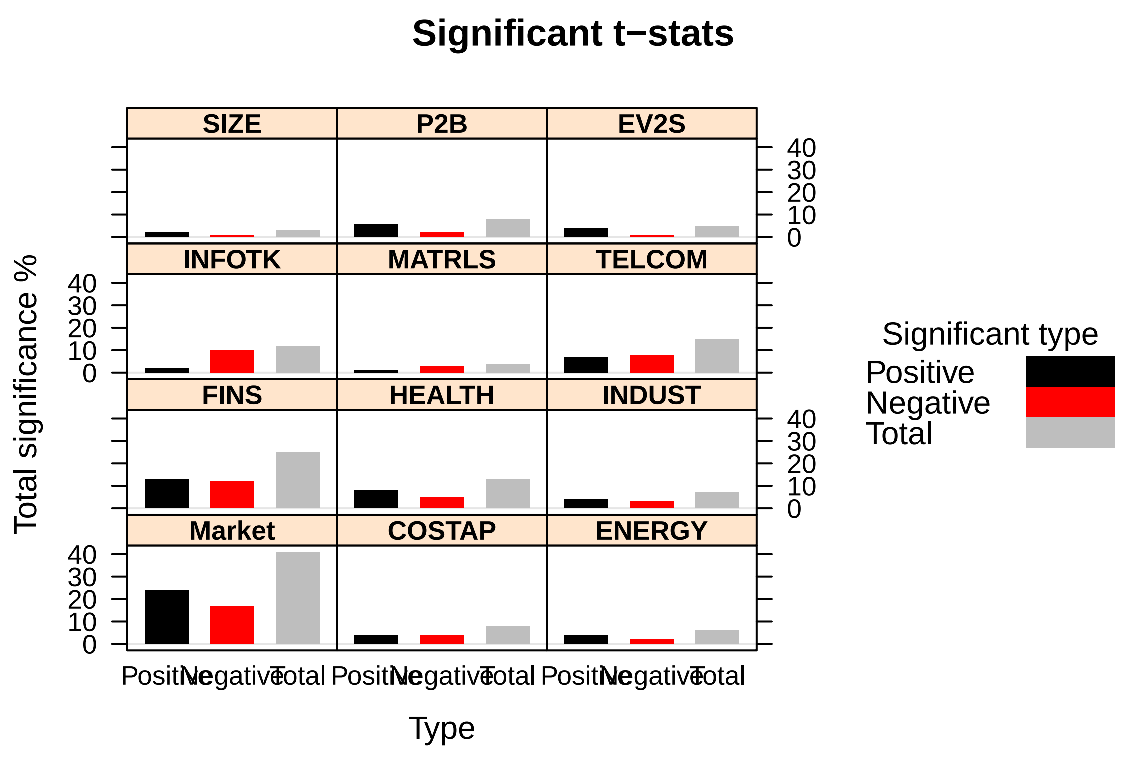

Figure 4: Number of Significant t-Statistics Each Month

4 Risk and Performance Reports

The factorAnalytics package contains the following risk and performance reporting functions:

- `repExposures`
- `repReturn`
- `repRisk`

We illustrate the use of each of these in turn using the FFM fitted object `fitDjia5Yr` and the corresponding global minimum variance long-only portfolio weights object `wtsDjiaGmvLo`. We first load the weight vector and give it the short name `wtsDjia`.

```
data(wtsDjiaGmvLo)
wtsDjia = wtsDjiaGmvLo
```

repExposures

The portfolio exposure to a given risk factor is the inner (dot) product of the portfolio weight vector with the column of the exposures matrix $\mathbf{B}_{t-1}$ corresponding to the given factor. The style factors vary over time, but the sector factors are fixed and each sector is represented by a column of zero's and ones. Thus the portfolio exposure to style factors will vary over time and thus have a distribution with a mean and volatility. On the other hand the portfolio exposure to sector factors will have a fixed value depending on the portfolio weights and number of firms in a given sector.

You compute portfolio exposures using the function `repExposures`, whose two main arguments are a FFM fitted model, and a portfolio weights vector. We first use this function to compute and print the volatilizes of the style factors and sector factors, and then use it to make barplots of those exposures. See Figure 5 for the latter.

```r
repExposures(fitDjia5Yr, wtsDjia, isPlot = FALSE, digits = 1,
    stripText.cex = 0.8, axis.cex = 0.8)

## $Style.Exposures
##       Mean Volatility
## SIZE  89.6        4.6
## P2B   25.2       20.0
## EV2S -31.5        6.8
##
## $Sec.Exposures
##
## Market 100.0
## COSTAP  47.3
## ENERGY  17.7
## FINS     0.0
## HEALTH  10.8
## INDUST   0.0
## INFOTK  16.7
## MATRLS   0.0
## TELCOM   7.5
```

```r
repExposures(fitDjia5Yr, wtsDjia, isPrint = F, isPlot = T,
    which = 3, add.grid = F, zeroLine = F, color = "Cyan")
```

Next we plot the time series of the six style factor exposures in Figure 6, and display boxplots of those exposures in Figure 7. These options are controlled by the which argument.

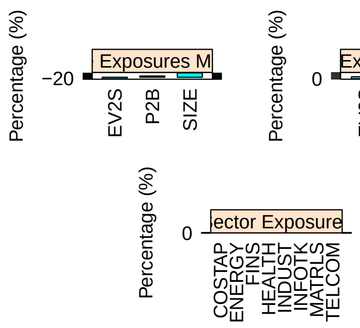

Figure 5: DJIA GMV Long-Only Portfolio Style and Sector Exposures Means and Volatilities

```r
repExposures(fitDjia5Yr, wtsDjia, isPrint = F, isPlot = T,
    which = 1, add.grid = F, zeroLine = T, color = "Blue",
    stripText.cex = 0.8, axis.cex = 0.8)
```

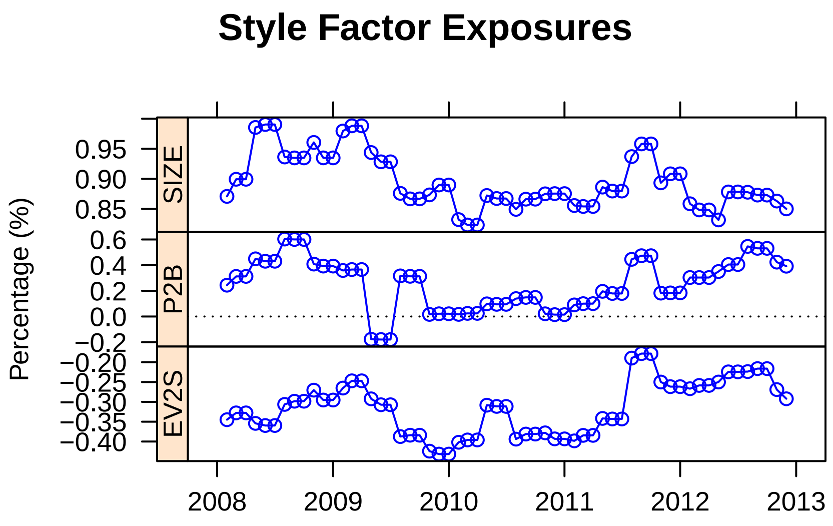

Figure 6: DJIA GMV Long-Only Portfolio Style Exposures Time Series

```r
repExposures(fitDjia5Yr, wtsDjia, isPrint = FALSE,
    isPlot = TRUE, which = 2, notch = F, layout = c(3,
        3))
```

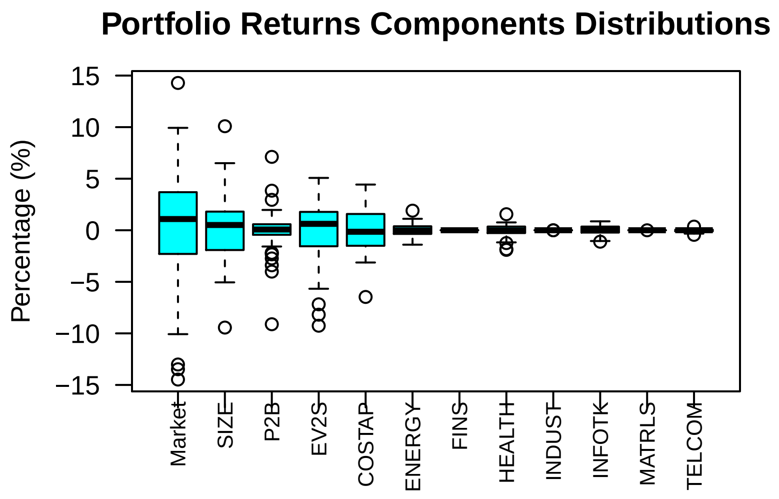

Figure 7: DJIA GMV Long-Only Portfolio Style Exposures Boxplots

repReturn

The function `repReteurn` provides you with the following choices of graphical reports, the results of which will also be printed because of the default printing option `isPrint = T`:

1. Time Series plot of portfolio returns decomposition
2. Time Series plot of portfolio style factors returns
3. Time Series plot of portfolio sector returns
4. Boxplot of Portfolio Returns Components.

We illustrate their use on the FFM fitted object `fitDjia5Yr` with GMV long-only portfolio weights object `wtsDjia`. If you just want a printout of the mean and volatility of the various portfolio return components without the graphical display, just change the default `isPlot=T` to `isPlot=F`.

```
repReturn(fitDjia5Yr, wtsDjia, isPlot = FALSE, digits = 2)

##            Mean Volatility
## PortRet    0.40       6.44
## ResidRet   0.10       1.46
## FacRet     0.31       5.93
## Market     0.48       5.78
## SIZE       0.03       3.22
## P2B       -0.11       2.01
## EV2S      -0.15       3.09
## COSTAP     0.07       2.02
## ENERGY     0.01       0.60
## FINS       0.00       0.00
## HEALTH    -0.04       0.60
## INDUST     0.00       0.00
## INFOTK     0.03       0.45
## MATRLS     0.00       0.00
## TELCOM    -0.02       0.14
```

Now you can get the graphical displays 1, 2, 3 and 4 with the following code. See Figures 8, 9, 10 and 11, respectively.

```
repReturn(fitDjia5Yr, wtsDjia, isPrint = FALSE, isPlot = TRUE,
    which = 1, add.grid = TRUE, scaleType = "same",
    color = "Blue", stripText.cex = 0.8, axis.cex = 0.8)
```

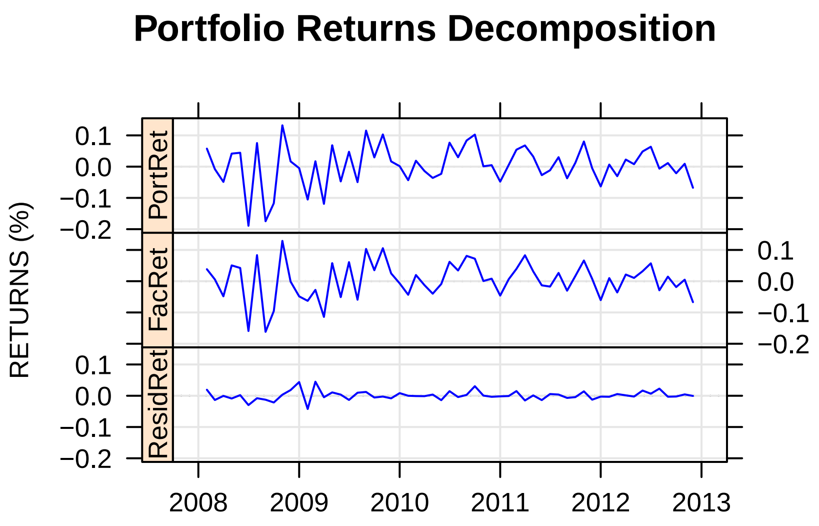

Figure 8: DJIA FFM Portfolio Factor Return and Specific Return Decomposition

```r
repReturn(fitDjia5Yr, wtsDjia, isPrint = FALSE, isPlot = TRUE,
    which = 2, add.grid = TRUE, zeroLine = T, color = "Blue",
    scaleType = "same", stripText.cex = 0.8, axis.cex = 0.8)
```

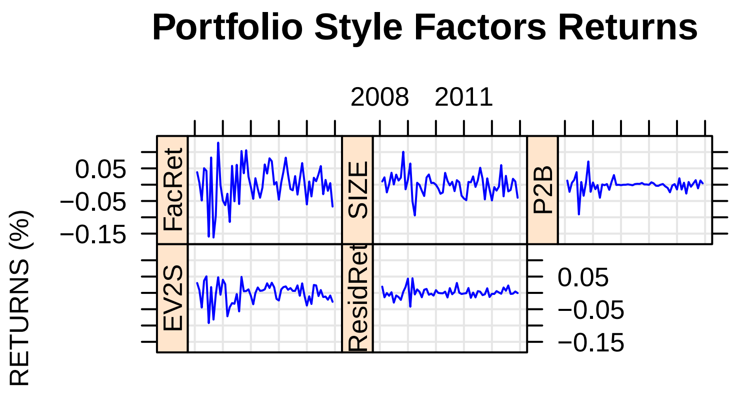

Figure 9: DJIA Fundamental Factor Model Portfolio Style Factor Returns

```r
repReturn(fitDjia5Yr, wtsDjia, isPrint = FALSE, isPlot = TRUE,
    which = 3, add.grid = TRUE, zeroLine = T, color = "Blue",
    scaleType = "same", stripText.cex = 0.8, axis.cex = 0.8)
```

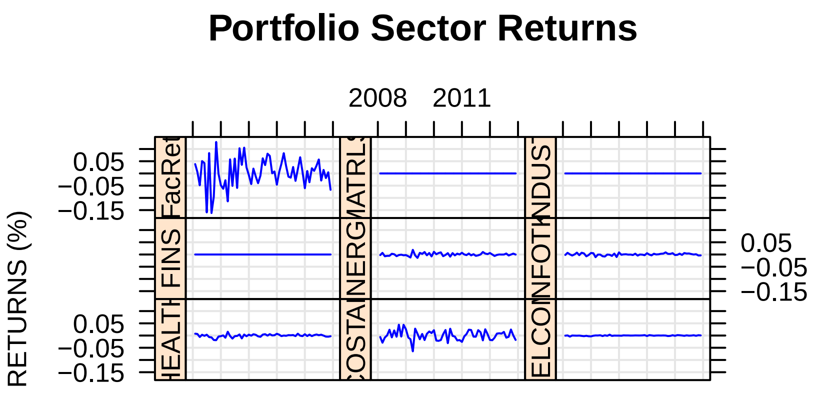

Figure 10: Fit145 Portfolio Sector Factor Returns

```r
repReturn(fitDjia5Yr, wtsDjia, isPrint = FALSE, isPlot = TRUE,
    which = 4)
```

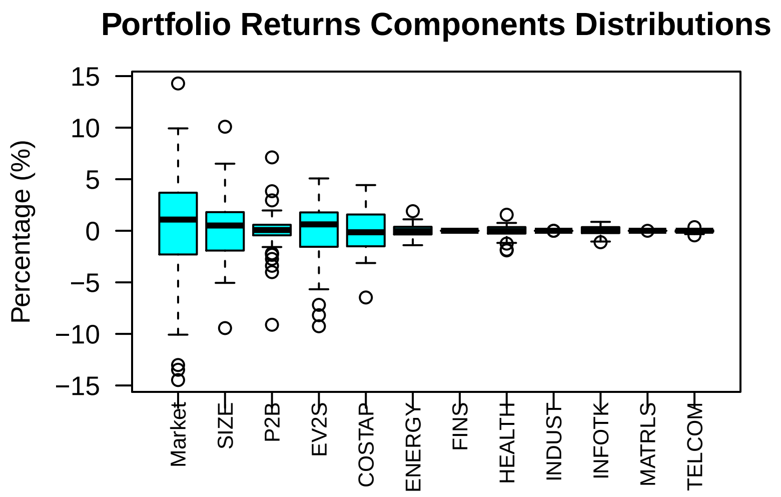

Figure 11: Fit145 Portfolio Return Components Boxplots

repRisk

The function `repRisk` allows one to compute and display (graphically and in tabular form) factor decompositions of risk for a portfolio and for each of the assets used to fit a fundamental factor model with `fitFfmDT`. The risk measure can be chosen as standard deviation/volatility (SD), expected shortfall (ES) or value-at-risk (VaR). and the factor risk decomposition can be chosen as factor percent contribution to risk (FPCR), factor contribution to risk (FCR) or factor marginal contribution to risk (FMCR). FPCR is the most commonly used by practitioners. The function `repRisk` has the has the following arguments:

> repRisk(object, weights = NULL, risk = c("Sd", "VaR", "ES"), decomp = c("RM", "FMCR", "FCR", "FPCR"), digits = NULL, invert = FALSE, nrowPrint = 20, p = 0.05, type = c("np", "normal"), sliceby = c("factor", "asset"), isPrint = TRUE, isPlot = FALSE, layout = NULL, portfolio.only = FALSE, ...)

The ES and VaR risk estimators can be computed using either a non-parametric risk measure estimator (often called a "historical" estimator) specified by the optional argument default risk = "np" above, or a parametric normal distribution risk measure estimator specified by risk = "normal". For VaR and ES the default tail probability of 5%, specified by the argument p = .05. The choice of factor risk decomposition type is specified by the decomp = optional argument. The other arguments will be explained in the examples below.

We illustrate using the fitted model `fitDjia5YrIntStyle` with and intercept (alpha) and style factors only for the DJIA returns data set `dataDjia.5Yr`.

```r
asset.var = "TICKER"
ret.var = "RETURN"
date.var = "DATE"
exposure.vars = c("SIZE", "P2B", "EV2S")
spec1 <- specFfm(data = dataDjia5Yr, asset.var = asset.var,
    ret.var = ret.var, date.var = date.var, exposure.vars = exposure.vars,
    weight.var = NULL, addIntercept = T, rob.stats = FALSE)
# lag the exposures
spec1 <- lagExposures(spec1)
# standardize the expsoures Cross-Sectionally
spec1 <- standardizeExposures(spec1, Std.Type = "CrossSection")
# fit the model
```

```
mdlFit <- fitFfmDT(spec1, fit.method = "WLS")

# extract regression results
results <- extractRegressionStats(spec1, fitResults = mdlFit)
# retrofit object
fitDjia5YrIntStyle <- convert(SpecObj = spec1, FitObj = mdlFit,
    RegStatsObj = results)
```

We also need the long-only global minimum variance portfolio weights:

```
data(wtsDjiaGmvLo)
wtsDjia = wtsDjiaGmvLo
```

First we compute an FPCR decomposition of the portfolio and individual assets using Sd as the risk measure, and provide both Lattice visualization and tabular displays. For the Lattice display, the argument `sliceby = “factor”` specifies that the panel conditioning is by risk facto,r and the choice `layout = c(5,1)` results in a single row with five panels. We used nrowPrint = 10 to shorten the printed output from one row for the portfolio factor risk decomposition and 22 rows (we are only using 22 of the DJIA stocks at the moment) for the stock factor risk decompositions to one row for the portfolio and 9 rows for the assets. The result is shown in Figure 12.

```
repRisk(fitDjia5YrIntStyle, wtsDjia, risk = "Sd", decomp = "FPCR",
    nrowPrint = 10, sliceby = "factor", isPrint = T,
    isPlot = T, layout = c(5, 1), stripText.cex = 0.8,
    axis.cex = 0.8)

## $SdFPCR
##            Alpha  SIZE  P2B EV2S Resid
## Portfolio 102.0 -18.3 -0.4 -1.3  18.0
## AA         28.9  45.2  4.0 -2.2  24.0
## BA         54.3  16.7  8.7  4.3  16.0
## BAC        15.1  -0.8  2.1 11.2  72.3
## CAT        37.0  12.3 -0.2 -1.4  52.4
## CVX        64.1 -12.8  4.4  4.9  39.4
## DD         53.1  28.0 -1.8 -1.6  22.3
## GE         34.6  -8.7  2.0 17.4  54.6
## HPQ          37.1  26.4  3.4 -0.2  33.3
## IBM          22.3  -0.6 26.0  0.1  52.3
```

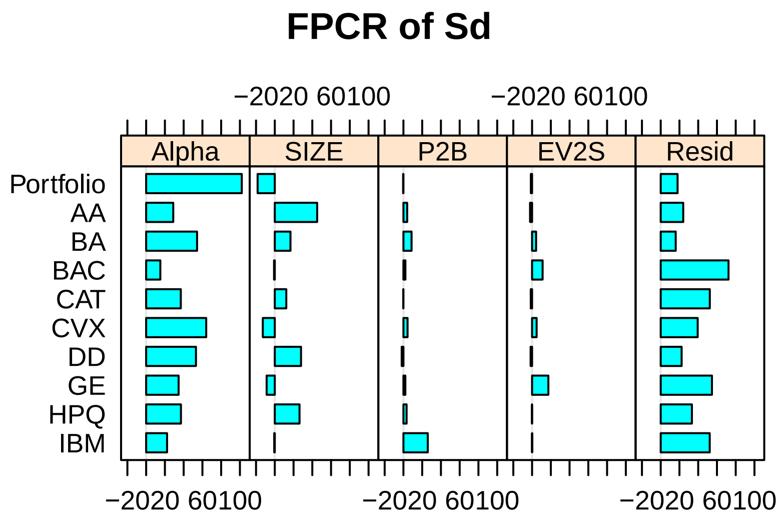

Figure 12: DJIA Stocks and GMV Long-Only Portfolio Standard Deviation FPCR Decompositions

Now we use expected shortfall (ES) to do an FPCR decomposition and provide only the Lattice display shown in Figure 13.

```r
repRisk(fitDjia5YrIntStyle, wtsDjia, risk = "ES", decomp = "FPCR",
    nrowPrint = 10, sliceby = "factor", isPrint = F,
    isPlot = T, layout = c(5, 1), stripText.cex = 0.8,
    axis.cex = 0.8)

## NULL
```

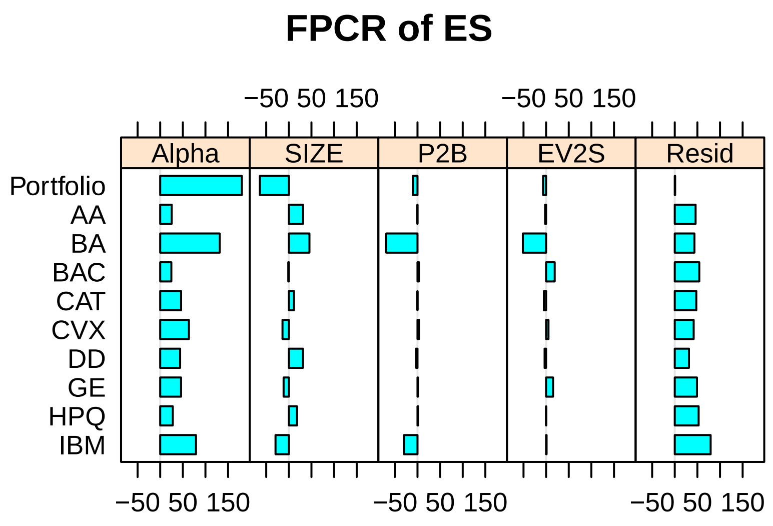

Figure 13: DJIA Stocks and GMV Long-Only Portfolio Expected Shortfall FPCR Decompositions

Now we use expected shortfall (ES) to do the FCR decomposition shown in Figure 14.

```r
repRisk(fitDjia5YrIntStyle, wtsDjia, risk = "ES", decomp = "FCR",
    nrowPrint = 10, sliceby = "factor", isPrint = F,
    isPlot = T, layout = c(5, 1), stripText.cex = 0.8,
    axis.cex = 0.8)

## NULL
```

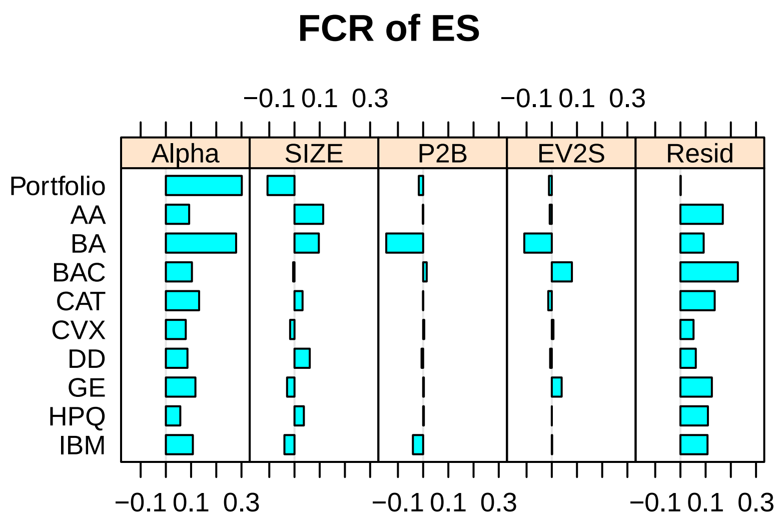

Figure 14: DJIA Stocks and GMV Long-Only Portfolio Expected Shortfall FCR Decompositions

A common use of repRisk is to compare the factor risk decompositions of a portfolio using all three risk measures, skipping the risk decomposition of the assets. This can be done as follows.

```
repRisk(fitDjia5YrIntStyle, wtsDjia, risk = c("Sd",
    "ES", "VaR"), decomp = "FPCR", sliceby = "factor",
    isPrint = T, isPlot = TRUE, layout = c(5, 1), portfolio.only = T,
    stripText.cex = 0.8, axis.cex = 0.8)

## $`Portfolio FPCR Non-Parametric`
##     Total Alpha  SIZE   P2B EV2S Resid
## Sd    100 102.0 -18.3  -0.4 -1.3  18.0
## ES    100 180.3 -64.2 -10.3 -6.7   0.9
## VaR   100 189.4 -76.6  -7.8 -3.0  -2.1
```

5 Factor Model Monte Carlo

The use of factor model Monte Carlo (FMMC) for a fundamental factor model results in a simulated set of asset returns based on a resampling of factor returns and a resampling or simulation of residual returns of the fitted model, using the exposures matrix from the last time period used in fitting the model. Our implementation of FMMC for fundamental factor model is both a special case and a generalization of FMMC for assets with unequal histories treated by Jiang and Martin (2015), "Better Risk and Performance Estimates with Factor Model Monte Carlo", Journal of Risk, June, 2015. The implementation is a special case in that we deal only with assets having equal histories, and it is a generalization in that the use of empirical residuals is generalized to include simulation of residuals from a fitted parametric distribution. The latter can be a normal distribution, a skewed-t distribution, or a Cornish-Fisher quantile based distribution. The method here and the method in Jiang and Martin (2015) both use resampled factor returns (no distribution or copula model for factor returns). A major advantage of FMMC for factor models is that the simulated asset returns can reflect the non-normality in the asset returns that is not captured by the factor model based covariance matrix. As such, the method can be advantageous for more accurate risk analysis, e.g., capturing tail risk more accurately, and for optimizing portfolios using downside risk measures such as expected shortfall.

Given a fitted fundamental factor model

$$\mathbf{r}_t = \mathbf{B}_{t-1}\hat{\mathbf{f}}_t + \hat{\boldsymbol{\varepsilon}}_t, \quad t = 1, 2, \cdots, T. \tag{3}$$

where a time-varying alpha or market factor, if any, is represented by a first column of ones in the $N \times K$ dimensional exposures matrix $\mathbf{B}_{t-1}$, the FMMC method uses the set of $K \times 1$ dimensional factor returns estimates $\hat{\mathbf{f}}_t$, $t = 1, 2, \cdots, T$, $N \times 1$ dimensional residuals $\hat{\boldsymbol{\varepsilon}}_t$, $t = 1, 2, \cdots, T$, and the end of period exposures matrix $\mathbf{B}_T$. [1]

These quantities are used to generate a simulated set of asset returns $\mathbf{r}^s_m m = 1, 2, \cdots, M$ as follows.

1. A set of M simulated factor returns $\mathbf{f}^s_m$ $m = 1, 2, \cdots, M$ are generated as M simple bootstrap samples of $\hat{\mathbf{f}}_t$, $t = 1, 2, \cdots, T$, i.e., by sampling the factor returns estimates with replacement.
2. A set of M simulated residual $\boldsymbol{\varepsilon}^s_m$ $m = 1, 2, \cdots, M$ are generated in one of the following two ways:
   (a) By forming M simple bootstrap samples of the $\hat{\boldsymbol{\varepsilon}}_t$, $\quad t = 1, 2, \cdots, T$, or
   (b) By estimating a parametric distribution for the $\hat{\boldsymbol{\varepsilon}}_t$, $\quad t = 1, 2, \cdots, T$ and making M Monte Carlo draws from the fitted distribution.
3. Using the results of the two steps above, compute the M simulated asset returns as

$$\mathbf{r}^s_m = \mathbf{B}_T\mathbf{f}^s_m + \boldsymbol{\varepsilon}^s_m \; m = 1, 2, \cdots, M. \tag{4}$$

[1]The discussion here applies equally well to the case where the time index of the exposures matrix is aligned with that of the factor returns rather than lagged by one time period.

It is to be noted that this method assumes that there is no serial correlation in the factor returns estimates or the residuals, and this assumption is usually a good approximation in practice. However, if one is concerned that there is such serial correlation the block bootstrap could be used in place of the combination of 1 and 2-(a) above.

The factorAnalytics function `fmmcSemiParametric` implements the above FMMC method based on function arguments that are the result of first fitting a fundamental factor model to the data with fitFfm, combined with function arguments based on user options concerning the type of FMMC. We will illustrate the use of `fmmcSemiParametric` on the DJIA five-year monthly data set `factorDataSetDjia5Yrs.` But first we take a look at the arguments of `fmmcSemiParam`:

```
args(fmmcSemiParam)

## function (B = 1000, factor.ret, beta, alpha, resid.par, resid.dist = c("normal",
##     "Cornish-Fisher", "skew-t", "empirical"), boot.method = c("random",
##     "block"), seed = 123)
## NULL
```

`B` is the number of bootstrap samples, `factor.ret` is the set of factor returns estimates returned by the use of `fitFfm`, `beta` is exposures matrix for the final period returned by `fitFfm`, and `alpha` is a fixed vector of intercept values that if omitted are assumed to be zero. Our example below uses the default values B = 1000, `boot.method = “random”` (means simple bootstrap), `seed = 123` (for reproducibility of the example). We use two choices of `resid.dist`, first we use `resid.dist = “empirical”` which corresponds to 2-(a) above and then we use `resid.dist = “normal”`. The user must provide appropriate values for `resid.par` that depend on the the choice for `resid.dist`. For the choice `resid.dist = “empirical”` the `resid.par` must be the $N \times T$ dimensional xts time series in the `$residuals` component of the model fit, and for the choice `resid.dist = “normal”` the `resid.par` must be an $N \times 2$ matrix with the first column being estimates of the means of the residuals for each of the N assets and the second column being estimates of the standard deviations of the residuals for each of the assets.

The result of using `fmmcSemiParametric` is a list with three components, each of which is a matrix containing the following simulated values:

- `sim.fund.return` (a $B \times N$ matrix of simulated asset returns)
- `boot.factor.ret` (a $B \times K$ matrix of simulated factor returns)
- `sim.resid` (a $B \times N$ matrix of simulated residuals)

In order to use `fmmcSemiParam` for the DJIA data we first fit a fundamental factor model (without alpha or market term) to the `factorDataSetDjia5Yrs` data:

```r
data("factorDataSetDjia5Yrs")
N = 22
exposure.vars <- c("P2B", "MKTCAP", "SECTOR")

spec1 <- specFfm(data = dataDjia5Yr, asset.var = asset.var,
    ret.var = ret.var, date.var = date.var, exposure.vars = exposure.vars,
    weight.var = NULL, addIntercept = FALSE, rob.stats = FALSE)
# lag the exposures
spec1 <- lagExposures(spec1)
# standardize the expsoures , you can also not call
# this
spec1 <- standardizeExposures(spec1, Std.Type = "None")
# fit the model
mdlFit <- fitFfmDT(spec1)
fit.ffm <- extractRegressionStats(spec1, fitResults = mdlFit)
```

Next we use `fmmcSemiParam` to create simulated values of the asset returns based on the use of bootstrapped factor returns and bootstrapped (empirical) residuals:

```r
resid.par = fit.ffm$residuals
fmmcDat = fmmcSemiParam(B = 1000, factor.ret = fit.ffm$factor.returns,
    beta = fit.ffm$beta, resid.par = resid.par, boot.method = "random",
    resid.dist = "empirical")
names(fmmcDat)

## [1] "sim.fund.ret"    "boot.factor.ret" "sim.resid"
```

Now let's verify that the that returns of the 22 DJIA stocks over the five-year period are well represented by the set of 500 simulated sets of 22 returns in `fmmcDat$sim.fund.return` with respect to returns means and standard deviations. In order to do this we first extract the multivariate time series of returns of those stocks from the data frame `factorDataSetDjia5Yrs.`

```r
data = factorDataSetDjia5Yrs
djiaDat = tapply(data$RETURN, list(data$DATE, data$TICKER),
    I)
djiaRet = xts(djiaDat, as.yearmon(rownames(djiaDat)))
```

Now we compare the simulated returns means with the observed returns means for the first 10 DJIA stocks, and see that the agreement is quite good.

```r
round(apply(djiaRet, 2, mean)[1:10], 3)

##     AA     BA    BAC    CAT    CVX     DD     GE    HPQ    IBM   INTC 
## -0.012  0.004  0.000  0.014  0.007  0.008  0.000 -0.016  0.013  0.002

round(apply(fmmcDat$sim.fund.ret, 2, mean)[1:10], 3)

##     AA     BA    BAC    CAT    CVX     DD     GE    HPQ    IBM   INTC 
## -0.010  0.011 -0.004  0.012  0.006  0.004 -0.003 -0.011  0.012  0.003
```

Now we do the same thing for returns standard deviations, and see that the agreement is also quite good.

```r
round(apply(djiaRet, 2, sd)[1:10], 3)

##    AA    BA   BAC   CAT   CVX    DD    GE   HPQ   IBM  INTC 
## 0.137 0.089 0.197 0.126 0.064 0.093 0.109 0.090 0.056 0.081

round(apply(fmmcDat$sim.fund.ret, 2, sd)[1:10], 3)

##    AA    BA   BAC   CAT   CVX    DD    GE   HPQ   IBM  INTC 
## 0.141 0.080 0.187 0.125 0.057 0.093 0.105 0.094 0.056 0.079
```

The above use of `fmmcSemiParam` with bootstrapped residuals as well as bootstrapped factor returns is attractive because it is simple and because in addition to making no distributional assumptions about the factor returns it makes no assumptions about the distributions of the residuals.

By way of contrast let's see what happens if we assume the residuals associated with the 22 DJIA fitted fundamental factor model returns have normal distributions and fit them using the sample means and standard deviations of the residuals. First we use `fmmcSemiParam` with default choice `resid.dist = “normal”` and with `resid.par` a matrix with first column the sample mean of the residuals and second column the standard deviation of the residuals

```
resid.mean = apply(B = 1000, coredata(fit.ffm$residuals),
    2, mean, na.rm = T)
resid.sd = matrix(sqrt(fit.ffm$resid.var))
resid.par = cbind(resid.mean, resid.sd)
fmmcDatNormal = fmmcSemiParam(factor.ret = fit.ffm$factor.returns,
    beta = fit.ffm$beta, resid.par = resid.par, boot.method = "random")
```

Then we compare the means and standard deviations of the simulated asset returns with those of the actual returns.

```
round(apply(djiaRet, 2, mean)[1:10], 3)

##     AA     BA    BAC    CAT    CVX     DD     GE    HPQ    IBM   INTC
## -0.012  0.004  0.000  0.014  0.007  0.008  0.000 -0.016  0.013  0.002

round(apply(fmmcDatNormal$sim.fund.ret, 2, mean)[1:10],
    3)

##     AA     BA    BAC    CAT    CVX     DD     GE    HPQ    IBM   INTC
## -0.013  0.010 -0.001  0.010  0.006  0.006 -0.004 -0.014  0.013  0.001

round(apply(djiaRet, 2, sd)[1:10], 3)

##    AA    BA   BAC   CAT   CVX    DD    GE   HPQ   IBM  INTC
## 0.137 0.089 0.197 0.126 0.064 0.093 0.109 0.090 0.056 0.081

round(apply(fmmcDatNormal$sim.fund.ret, 2, sd)[1:10],
    3)

##    AA    BA   BAC   CAT   CVX    DD    GE   HPQ   IBM  INTC
## 0.121 0.081 0.155 0.099 0.060 0.114 0.089 0.089 0.061 0.082
```

Once again the mean values agree quite well, but we see that the simulated returns based on the assumption of normally distributed returns have volatilities that under-estimate the actual returns volatilities for six of the first 10 stocks, sometimes substantially so. However, comparison of the volatilities for the simulated returns reveals that the volatilities for the simulated returns are closer to those of the actual returns. There is a detailed reason for this, which leave for the reader to ponder, and just point out that it has to do with the fact that some of the stocks have substantially non-normal returns distributions.

Main message: It is not safe to use normal distributions in modeling stock returns with a fundamental factor model (or otherwise). It is for this reason that `fmmcSemiParam` allows you to use skewed t-distributions and Cornish-Fisher quantile representation of non-normal distributions for the residuals.

6 Market plus Industry plus Country Models

In this discussion we treat the terms "Industry" and "Sector" interchangeably, noting that for some models, e.g., a U.S. equity model, one may prefer to just use sector factors but may also wish to use industry factors, and a global model with countries may also contain industry factors. Our current examples use sector factors but we refer to them in our mathematical models loosely as industry factors. For background on implementing the model fitting constraints required by the models in this section, see the Appendix in Menchero, J. (2010) "Characteristics of Factor Portfolios", *The Journal of Portfolio Management*.

Market + Industry Model

The market plus industry sector model has the form

$$\begin{aligned} \mathbf{r}_t &= \left(\mathbf{1} \;\vdots\; \mathbf{B}_i \right)\mathbf{f}_{mi,t} + \boldsymbol{\varepsilon}_t, \quad t = 1, 2, \cdots, T \\ &= \mathbf{B}_{mi}\mathbf{f}_{mi,t} + \boldsymbol{\varepsilon}_t \end{aligned} \tag{5}$$

where $\mathbf{B}_{mi}$ is an $N \times (K+1)$ matrix with $\mathbf{B}_i$ an $N \times K$ matrix of $1's$ and $0's$ representing K industry sectors, with each stock belonging to one and only one sector over the given time interval, and

$$\mathbf{f}_{mi,t} = (f_{0,t}, f_{1,t}, , f_{2,t}, \cdots, f_{K,t})'. \tag{6}$$

It follow that the sum of the K column vectors of $\mathbf{B}_i$ is a vector of $1's$, and $\mathbf{B}_{mi}$ is rank deficient with rank K instead of $K+1$. Consequently the use of least squares to fit the above model for each time period does not lead to a unique solution.

In order to obtain a unique LS solution one can reparameterize the model in such a way as to impose a constraint that leads to a unique solution. The most natural way to do this is to treat the factor return $f_{0,t}$ as a market component of return and treat the factor returns , $f_{1,t}$, , $f_{2,t}, \cdots, f_{K,t}$ as deviations from the market return. Thus we impose a constraint:

$$f_{1,t} + f_{2,t} + \cdots + f_{K,t} = 0. \tag{7}$$

This can be accomplished with the reparameterization[2]

$$\mathbf{f}_{mi,t} = \mathbf{R}_{mi}\mathbf{g}_{mi,t} \tag{8}$$

$$\mathbf{r}_t = \mathbf{B}_{mi}\mathbf{R}_{mi}\mathbf{g}_{mi,t} + \boldsymbol{\varepsilon}_t \tag{9}$$

where

$$\mathbf{R}_{mi} = \begin{pmatrix} \mathbf{I}_K \\ \boldsymbol{a}' \end{pmatrix} \sim (K+1) \times K \tag{10}$$

$$\boldsymbol{a}' = (0, -1, -1, \cdots, -1) \ \sim K \times 1 \tag{11}$$

$$\mathbf{g}_{mi,t} = (g_{1,t}, g_{2,t}, \cdots, g_{K,t})'. \tag{12}$$

The model (9) now has a unique least-squares solution $\hat{\mathbf{g}}_t$, and it is easy to check that (8) insures that the constraint (7) is satisfied.

We will illustrate use of `fitFfmDT` to fit a market plus sector model to the DJIA stock returns and sector data. But first we fit a pure sector model without a market component and examine the factor return coefficients for the first month of the five-year fitting window as a reference point.

[2]See the Appendix of Menchero, J. (2010). "The Characteristics of Factor Portfolios", *The Journal of Performance Measurement*, Fall 2010.

```
dat = factorDataSetDjia5Yrs

spec1 <- specFfm(data = dat, asset.var = "TICKER",
    ret.var = "RETURN", date.var = "DATE", exposure.vars = "SECTOR",
    weight.var = NULL, addIntercept = F, rob.stats = FALSE)
# lag the exposures
spec1 <- lagExposures(spec1)

# fit the model
mdlFit <- fitFfmDT(spec1)

# extract regression results
results <- extractRegressionStats(spec1, fitResults = mdlFit)
# retrofit object
fitSec <- convert(SpecObj = spec1, FitObj = mdlFit,
    RegStatsObj = results)



round(coef(summary(fitSec)$sum.list[[1]])[, 1], 3)

## SECTORCOSTAP SECTORENERGY   SECTORFINS SECTORHEALTH SECTORINDUST SECTORINFOTK
##       -0.007        0.034       -0.121       -0.028       -0.016        0.037
## SECTORMATRLS SECTORTELCOM
##        0.082       -0.080

round(fitSec$factor.returns[1, ], 3)

##            COSTAP ENERGY   FINS HEALTH INDUST INFOTK MATRLS TELCOM
## 2008-02-01 -0.007  0.034 -0.121 -0.028 -0.016  0.037  0.082  -0.08
```

Note that the last two lines of code produce identical results. This is because without any constraints such as those discussed above, the coefficients of the cross-section regression at each time period are extracted to form the time series of factor returns in the `factor.returns` component of the `ffm` object.

Now we fit a market plus sector model by adding the `fitFfm` argument `addIntercept = T`, and examine the coefficients $\hat{\mathbf{g}}_{mi,1}$ and the resulting factor returns $\hat{\mathbf{f}}_{mi,1}$ for the first month of the five-year fitting window.

```r
spec1 <- specFfm(data = dat, asset.var = "TICKER",
    ret.var = "RETURN", date.var = "DATE", exposure.vars = "SECTOR",
    weight.var = NULL, addIntercept = T, rob.stats = FALSE)
# lag the exposures
spec1 <- lagExposures(spec1)

# fit the model
mdlFit <- fitFfmDT(spec1)

# extract regression results
results <- extractRegressionStats(spec1, fitResults = mdlFit)

fitSecInt <- convert(SpecObj = spec1, FitObj = mdlFit,
    RegStatsObj = results)


round(coef(summary(fitSecInt)$sum.list[[1]])[, 1],
    2)

##    V1    V2    V3    V4    V5    V6    V7    V8
## -0.01  0.01  0.05 -0.11 -0.02  0.00  0.05  0.09

round(fitSecInt$factor.returns[1, ], 2)

##            Market COSTAP ENERGY  FINS HEALTH INDUST INFOTK MATRLS TELCOM
## 2008-02-01  -0.01   0.01   0.05 -0.11  -0.02      0   0.05   0.09  -0.07

round(sum(fitSecInt$factor.returns[1, -1]), 2)

## [1] 0
```

Note that the next to last line of code above prints the unique least squares model coefficients vector $\hat{\mathbf{g}}_{mi,1}$ for month 1 (8 of them since there are 8 sectors). On the other hand the last line of code above prints the 9 factor returns coefficients $\hat{\mathbf{f}}_{mi,1} = (f_{0,1}, f_{1,1}, \cdots, f_{8,1})$ for month 1, the first one for the market and the other 8 for the sectors. This is because the `factor.returns` component of the fitted `ffm` object contain the results of using (8) to compute the factor returns $\hat{\mathbf{f}}_{mi,t}$ from the fitted model coefficients $\hat{\mathbf{g}}_{mi,t}$. We see that the last line of code results in zero as expected since the sum of the factor returns is constrained to be zero by the transformation (8).

N.B. Note that the factor returns covariance matrix estimate returned by `the factor.cov` component of the fitted `ffm` object is computed from the time series $\hat{\mathbf{g}}_{mi,t}$, not from the time series $\hat{\mathbf{f}}_{mi,t}$. Computing a factor returns sample covariance matrix from the latter will result in a singular covariance matrix due to the constraint (7).

There is no problem in extending the model (5) to include style factors. Dropping the time subscript for simplicity, and adding a style factors component the model would be

$$\mathbf{r} = \mathbf{B}_s\mathbf{f}_s + \mathbf{B}_{mi}\mathbf{f}_{mi} + \boldsymbol{\varepsilon} \tag{13}$$

where $\mathbf{B}_s$ is an $N \times K_s$ matrix of style exposures and $\mathbf{f}_s$ is a $K_s \times 1$ vector of style factor returns.

Style + Market + Industry + Country Model

The general form of the style-market-industry-country model is:

$$\mathbf{r} = \mathbf{B}_s\mathbf{f}_s + \mathbf{B}_{mi}\mathbf{f}_{mi} + \mathbf{B}_c\mathbf{f}_c + \boldsymbol{\varepsilon} \tag{14}$$

with style exposures matrix $\mathbf{B}_s \sim N \times K_1$, market and industry exposures matrix $\mathbf{B}_{mi} \sim N \times K_2$, and country exposures matrix $\mathbf{B}_c \sim N \times K_3$, with corresponding style factor returns $\mathbf{f}_s \sim K_1 \times 1$, and the following market-industry factor returns and country factor returns, respectively:

$$\mathbf{f}_{mi} = (f_{mi,0}, f_{mi,1}, \cdots, f_{mi,K_2})' \tag{15}$$

$$\mathbf{f}_c = (f_{c,1}, f_{c,2}, \cdots, f_{c,K_3})'\,. \tag{16}$$

As before the market-industry exposures matrix $\mathbf{B}_{mi}$ is rank deficient and this deficiency is removed by sum of industry factor returns constraint

$$f_{mi,1} + f_{mi,2}, + \cdots + f_{mi,K_2} = 0 \tag{17}$$

which can be implemented just as in equations (8) through (11) with $K = K_2$ and

$$\mathbf{g}_{mi} = (g_{mi,1}, g_{mi,2}, \cdots, g_{mi,K_2})'. \tag{18}$$

But since the columns of the country exposures matrix add to a vector of one's, the country factor part of the model results in an additional rank deficiency. This rank deficiency can be removed by using the country factor returns constraint

$$f_{c,1} + f_{c,2} + \cdots, +f_{c,K_3} = 0. \tag{19}$$

which can be implemented by setting

$$\mathbf{f}_c = \mathbf{R}_c\mathbf{g}_c \tag{20}$$

$$\mathbf{g}_c = (g_{c,1}, g_{c,2}, \cdots, g_{c,K_3})' \tag{21}$$

with the country restriction matrix

$$\mathbf{R}_c = \begin{pmatrix} \mathbf{I}_{K_3-1} \\ \boldsymbol{b}' \end{pmatrix} \sim K_3 \times (K_3 - 1) \tag{22}$$

$$\boldsymbol{b}' = (-1, -1, \cdots, -1) \sim 1 \times (K_3 - 1). \tag{23}$$

A Simulated Data Example

We have created an artificial example of a market+sector+country model (where sector plays the role of industry) consisting of random returns of 22 stocks with three sectors for the sector factor and two countries for the countries factor, for each of five months. The normally distributed returns for the three sectors alone have means of 1, 2, 3, with standard deviations .2. The two countries contribute additional normally distributed returns having means 4 and 5 with standard deviations .2. So returns associated with the first country have means 5, 6, 7 and means associated with the second country have means 6, 7, 8. Thus the overall mean of 6.5.

The code for creating the returns is as follows.

```r
# Country Incremental Components of Asset Returns
set.seed(10000)
Bind = cbind(rep(1, 30), c(rep(1, 10), rep(0, 20)),
    c(rep(0, 10), rep(1, 10), rep(0, 10)), c(rep(0,
        20), rep(1, 10)))
cty1 = matrix(rep(c(0, 1), 15))
cty2 = matrix(rep(c(1, 0), 15))
Bmic = cbind(Bind, cty1, cty2)
dimnames(Bmic)[[2]] = c("mkt", "sec1", "sec2", "sec3",
    "cty1", "cty2")
r.add = rnorm(30, 4, 0.2)
r.cty1 = rep(0, 30)
r.cty2 = rep(0, 30)
for (i in 1:30) {
    if (Bmic[i, "cty1"] == 1) {
        r.cty1[i] = r.add[i]
        r.cty2[i] = 0
    } else {
        r.cty1[i] = 0
        r.cty2[i] = r.add[i] + 1
    }
}

# Asset Returns for Market+Industry+Country Model
mu = c(1, 2, 3)
sd = c(0.2, 0.2, 0.2)
r = list()
r.mic = list()
fitMic = list()
fitMic1 = list()
for (i in 1:5) {
    set.seed(1099923 + (i - 1))
    r[[i]] = c(rnorm(10, mu[1], sd[1]), rnorm(10, mu[2],
        sd[2]), rnorm(10, mu[3], sd[3]))
    r.mic[[i]] = r[[i]] + r.cty1 + r.cty2
}
```

A normal qq-plot of the 30 asset returns for the first of the 5 time periods is shown in Figure 15.

```
qqnorm(r.mic[[1]], main = "MIC Model Equity Returns for First Period",
    xlab = "NORMAL QQ-PLOT", ylab = "RETURNS")
```

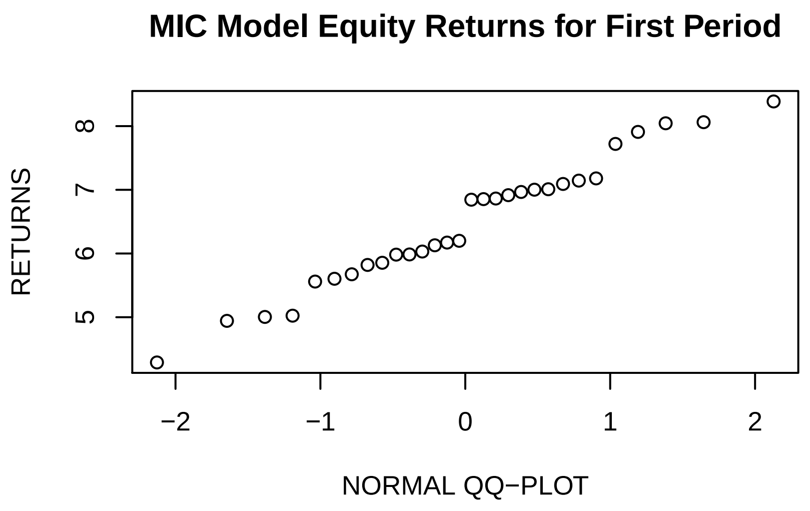

Figure 15: Normal QQ-Plot MIC Model Asset Returns First Period

Now we build the data frame required by `fitFfm`, fit the MIC model and display the factor returns for each of the five months. What we have been calling the Industry factor is called Sector for this example.

```
Returns = unlist(r.mic)
COUNTRY = rep(rep(c("US", "India"), 15), 5)
SECTOR = rep(rep(c("SEC1", "SEC2", "SEC3"), each = 10),
    5)
TICKER = rep(c(LETTERS[1:26], paste0("A", LETTERS[1:4])),
    5)
DATE = rep(seq(as.Date("2000/1/1"), by = "month", length.out = 5),
    each = 30)
data.mic = data.frame(DATE = as.character(DATE), TICKER,
    Returns, SECTOR, COUNTRY)
```

```
exposure.vars = c("SECTOR", "COUNTRY")
fit = fitFfm(data = data.mic, asset.var = "TICKER",
    ret.var = "Returns", date.var = "DATE", exposure.vars = exposure.vars,
    addIntercept = T)
fit$factor.returns

##              Market       SEC1         SEC2      SEC3      India        US
## 2000-02-01 6.502875 -0.9644445 -0.005385534 0.9698300 -0.5012401 0.5012401
## 2000-03-01 6.531066 -0.9654840 -0.027179390 0.9926634 -0.5125961 0.5125961
## 2000-04-01 6.518297 -0.9564166 -0.009542802 0.9659594 -0.5127447 0.5127447
## 2000-05-01 6.583097 -1.0082975 -0.027957774 1.0362552 -0.5156385 0.5156385
```

We see that the Market values of the factor have values clustering around 6.5 as expected. We can also see that the three sector factor returns sum to zero and the two country factor returns sum to zero, as expected due to the constraints that they sum to zero.

Additional Mathematical Model Details

The style-market-industry-country model (14) may be written as

$$\mathbf{r} = \mathbf{B}_s\mathbf{f}_s + \tilde{\mathbf{B}}_{mi}\mathbf{g}_{mi} + \tilde{\mathbf{B}}_c\mathbf{g}_c + \boldsymbol{\varepsilon} \tag{24}$$

where the modified exposures matrices

$$\tilde{\mathbf{B}}_{mi} = \mathbf{B}_{mi}\mathbf{R}_{mi} \sim N \times K_2, \qquad \tilde{\mathbf{B}}_c = \mathbf{B}_c\mathbf{R}_c \sim N \times (K_3 - 1) \tag{25}$$

are full rank.

With $\mathbf{B}_{simc} = \left(\mathbf{B}_s \,\vdots\, \tilde{\mathbf{B}}_{mi} \,\vdots\, \tilde{\mathbf{B}}_c \right) \sim N \times (K_1 + K_2 + K_3 - 1)$ and $\mathbf{g}_{smic} = \left(\mathbf{f}_s', \mathbf{g}_{mi}', \mathbf{g}_c'\right)' \sim (K_1 + K_2 + K_3 - 1) \times 1$, the model (24) may be written in the form

$$\mathbf{r} = \mathbf{B}_{smic}\mathbf{g}_{smic} + \boldsymbol{\varepsilon}. \tag{26}$$

Under the usual condition that $\mathbf{B}_s$ is full rank the matrix $\mathbf{B}_{simc}$ will have full rank $K_1 + K_2 + K_3 - 1$, and the model (26)have a unique LS (or LS) solution

$$\hat{\mathbf{g}}_{smic} = \left(\hat{\mathbf{f}}'_s, \hat{\mathbf{g}}'_{mi}, \hat{\mathbf{g}}'_c\right)' . \tag{27}$$

As usual the fundamental factor model will be fit at times $t = 1, 2, \cdots, T$, thereby generating the time series of factor returns:

$$\hat{\mathbf{g}}_{smic,t} = \left(\hat{\mathbf{f}}'_{s,t}, \hat{\mathbf{g}}'_{mi,t}, \hat{\mathbf{g}}'_{c,t}\right)' . \tag{28}$$

The `factor.cob` component of the fitted `ff` object is the covariance matrix estimated from the above time series. On the other hand, the `factor.returns` component of the fitted `ff` object is:

$$\hat{\mathbf{f}}_{smic,t} = \left(\hat{\mathbf{f}}'_{s,t}, \hat{\mathbf{f}}'_{mi,t}, \hat{\mathbf{f}}'_{c,t}\right)' = \left(\hat{\mathbf{f}}'_{s,t}, \mathbf{R}_{mi}\hat{\mathbf{g}}'_{mi,t}, \mathbf{R}_c\hat{\mathbf{g}}'_{c,t}\right)' . \tag{29}$$

N.B. The t-statistics for $\hat{\mathbf{f}}'_{s,t}$ are computed using the diagonal elements of that part of covariance matrix in `factor.cob` associated with that factor return. But the t-statistics for $\hat{\mathbf{f}}'_{mi,t}$ and $\hat{\mathbf{f}}'_{c,t}$ are computed using the diagonal elements of $\mathbf{R}_{mi}\text{cov}(\hat{\mathbf{g}}'_{mi,t})\mathbf{R}'_{mi}$ and $\mathbf{R}_c\text{cov}(\hat{\mathbf{g}}'_{c,t})\mathbf{R}'_c$, respectively.