

Fundamental Factor Models Vignette

Doug Martin, Avinash Acharya, Lingjie Yi

August 29, 2016

Abstract

This report describes new functions in the R package `factorAnalytics` implemented during the Google Summer of Code (GSoC) 2016 *factorAnalytics* project. GSoC funding was awarded to UW AMath MS-CFRM student Avinash Acharya as project lead, with Doug Martin and Eric Zivot as mentors. MS-CFRM Graduate and AMath Professional Staff Member Lingjie (Chris) Yi is supported on the project by AMath faculty funds.

1 The Equity Fundamental Factor Model and Data Sets

The general mathematical form of equity fundamental factor model (FFM) implemented in `factorAnalytics` is

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

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 $\mathbf{D}_t = \text{diag}(\sigma_{\varepsilon t,1}^2, \sigma_{\varepsilon t,2}^2, \dots, \sigma_{\varepsilon t,N}^2)$, 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 30 stocks in the DJIA from January 2000 to March 2013, with 5 corresponding style factors (MKT CAP, ENTVAL, P2B, EV2S, SIZE) and a sector factor with 9 sectors.

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

stocks145scores6: Monthly returns of 145 stocks in the CRSP database from January 1990 to December 2014, with 6 corresponding style factors (ROE, BP, MOM121, SIZE, VOL121, EP) and a sector factor with 10 sectors.

wtsDjiaGmvLo: Weight vector of dimension 30 containing the weights of a long-only global minimum variance (GMV) portfolio for the 30 stocks in the `factorDataSetDjia5Yrs` data set. The weight vector was obtained using `PortfolioAnalytics` with the usual sample covariance matrix based on the 5 years of returns in `factorDataSetDjia5Yrs`.

wtsStocks145GmvLo: Weight vector of dimension 145 containing the weights of a long-only global minimum variance (GMV) portfolio for the 145 stocks in the `wtsStocks145GmvLo` data set for the five year time window from January 2008 to December 2012. The weight vector was obtained using `PortfolioAnalytics` with the usual sample covariance matrix based on that 5 years of returns.

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

```
help(stocks145scores6)
```

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

You can load `factorDataSetDjia5Yrs` and view the list components of the data for the first two stocks in January 2008 with:

```
data("factorDataSetDjia5Yrs")
dataDjia.5Yr = factorDataSetDjia5Yrs
head(dataDjia.5Yr, 2)
```

##		X	DATE	PERMNO	GVKEY	CUSIP	TICKER	NAME
##	2881	2881	Jan 2008	24643	1356	013817101	AA	ALCOA INC
##	2882	2882	Jan 2008	59176	1447	025816109	AXP	AMERICAN EXPRESS CO
##		RETURN	GSECTOR	SECTORNAMES	SECTOR	MKTCAP	ENTVAL	P2B
##	2881	-0.094665	15	Materials	MATRLS	29.38908	37.50708	1.865026
##	2882	-0.052095	40	Financials	FINS	50.62776	105.66676	4.398207
##		EV2S	SIZE					
##	2881	1.339921	10.28838					
##	2882	3.217232	10.83226					

You can load `stocks145scores6`, extract a five-year segment from 2008 to 2012, and view the list components of the data for the first two stocks in January 2008 with:

```
data("stocks145scores6")
dat = stocks145scores6
dat$DATE = as.yearmon(dat$DATE)
data145.5Yr = dat[dat$DATE >= as.yearmon("2008-01-01") &
  dat$DATE <= as.yearmon("2012-12-31"), ]
head(data145.5Yr, 2)
```

```
##          DATE compustat_id TICKER          NAME GSECTOR
## 31321 Jan 2008      197601    BHI BAKER HUGHES INC      10
## 31322 Jan 2008      854301    MO ALTRIA GROUP INC      30
##          SECTORNAMES SECTOR      ROE      BP      MOM121      SIZE
## 31321      Energy ENERGY 0.2531009 0.2954860 0.1748515 23.75465
## 31322 ConsumerStaples COSTAP 0.4537387 0.1080207 -0.1351413 25.79438
##          VOL121      EP RAWRETURN      RETURN      rf
## 31321 0.4290841 0.06935882 -0.198397 -0.198397 0.005008
## 31322 0.2275820 0.06201346 0.002779 0.002779 0.005008
```

2 Fitting a Fundamental Factor Model

The 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

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

We use the `factorDataSetDjia5Yrs` to illustrate the use of the function `fitFfm` to fit a FFM.

```
fitDjia5Yr = fitFfm(dataDjia.5Yr, addIntercept = T,
  asset.var = "TICKER", ret.var = "RETURN", date.var = "DATE",
  exposure.vars = c("SECTOR", "P2B", "MKTCAP"))
```

```
names(fitDjia5Yr)

## [1] "factor.fit"      "beta"           "factor.returns"
## [4] "residuals"       "r2"             "factor.cov"
## [7] "g.cov"           "resid.cov"      "return.cov"
## [10] "restriction.mat" "resid.var"       "call"
## [13] "data"            "date.var"        "ret.var"
## [16] "asset.var"       "exposure.vars"   "weight.var"
## [19] "fit.method"      "asset.names"     "factor.names"
## [22] "time.periods"
```

See the `fitFfm` 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 `ffmRsqr` to make a plot of the time series of R-squared values for each of the 60 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 `ffmRsqr` below as shown below results in both as shown in Figure 2.

```
ffmRsqr(fitDjia5Yr, rsqAdj = T, plt.type = 2, isPrint = F)

##      Mean R-Squared Mean Adj R-Squared
##              0.66              0.46
```

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

Model Fit Variance Inflation Factors

When your model includes continuous style factor variables the function `ffmRsqr` 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 2.

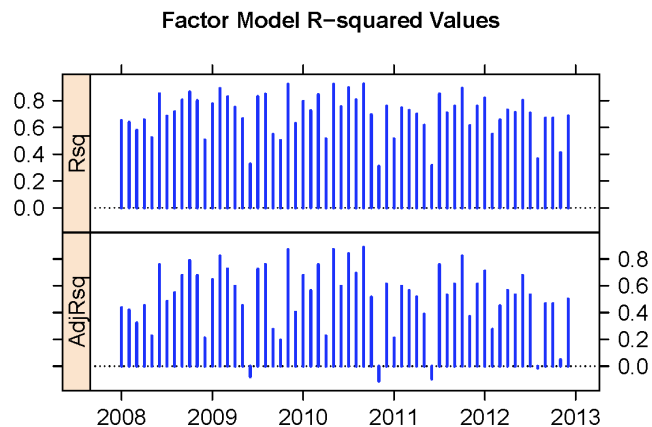


Figure 1: DJIA Data FFM R-Squared Values for Each of 60 Months

```
ffmRsq(fitDjia5Yr, VIF = T, rsq = F, rsqAdj = F, isPrint = F)
```

```
## $Mean.VIF
##      P2B MKTCAP
##      1.02   1.02
```

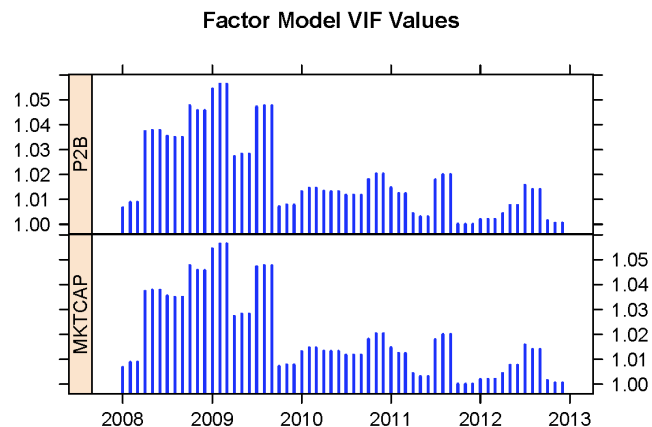


Figure 2: DJIA Data FFM Variance Inflation Factors for Each of 60 Months

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. The function `ffmTstats` also determines the number of significant t-statistics each month, as shown in Figure 4.

```
ffmTstats(fitDjia5Yr) # NOTE: Use las = 2
```

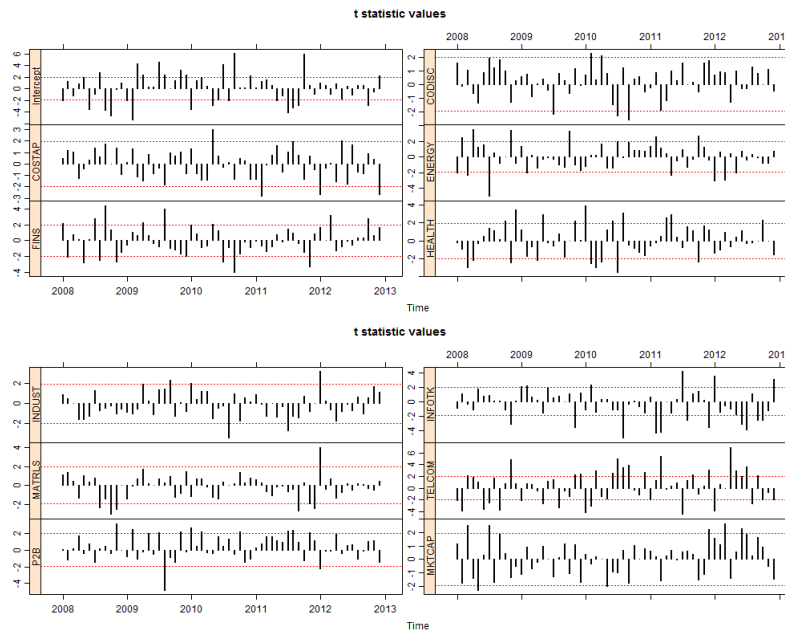


Figure 3: DJIA Data FFM t-Statistics for 9 Sectors and 3 Style Factors

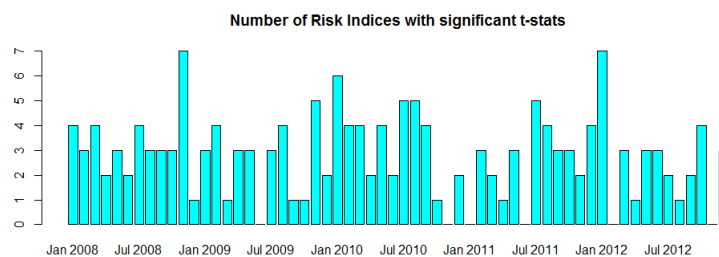


Figure 4: DJIA FFM Number of Significant t-Statistics Each Month

3 Risk and Performance Reports

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

- `repExposures`

- `repReturn`
- `fmXXDecomp` (with `XX = Sd, Es, VaR`)
- `portXXDecomp` (with `XX = Sd, Es, VaR`)
- `repRisk`

We illustrate the use of each of these in turn for the FFM using either the `dat145.5Yr` data (created in Section 1 from the 5 year segment of `stocks145scores6`) and the corresponding global minimum variance portfolio weights `wtsStocks145GmvLo` for that data, or the `factorDataSetDjia5Yrs` and corresponding global minimum variance portfolio weights `wtsDjiaGmvLo`. We first load these two GMV weight vectors.

```
data(wtsStocks145GmvLo)
wts145 = wtsStocks145GmvLo
data(wtsDjiaGmvLo)
wtsDjia = wtsDjiaGmvLo
```

The data set `dat145.5Yr` has 10 sectors and 6 style factors (called “scores” by Capital IQ). We fit a FFM to this data set, naming the result `fit145`.

```
fit145 = fitFfm(data = data145.5Yr, exposure.vars = c("SECTOR",
  "ROE", "BP", "MOM121", "SIZE", "VOL121", "EP"),
  date.var = "DATE", ret.var = "RETURN", asset.var = "TICKER",
  fit.method = "WLS", z.score = TRUE)
```

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 volatilities of the style factors and sector factors, and then to make barplots of those exposures. See Figure 5 for the latter.

```
repExposures(fit145, wts145, isPlot = FALSE, digits = 1)
```

##		Mean	Volatility
##	ROE	-0.1	15.8
##	BP	9.1	25.0
##	MOM121	-1.0	23.5
##	SIZE	-12.0	28.2
##	VOL121	-5.3	24.0
##	EP	3.4	15.5
##	CODISC	2.8	0.0
##	COSTAP	19.5	0.0
##	ENERGY	8.8	0.0
##	FINS	0.0	0.0
##	HEALTH	14.2	0.0
##	INDUST	2.7	0.0
##	INFOTK	5.2	0.0
##	MATRLS	5.4	0.0
##	TELCOM	0.0	0.0
##	UTILS	41.4	0.0

```
repExposures(fit145, wts145, isPrint = F, isPlot = T,
  which = 3, add.grid = F, zeroLine = F, color = "Blue",
  layout = c(1, 3))
```

Next we plot the time series of the six style factor exposures in Figure 6, and display boxplots of those exposures in Figure 7.

```
repExposures(fit145, wts145, isPrint = F, isPlot = T,
  which = 1, add.grid = F, zeroLine = T, color = "Blue")
```

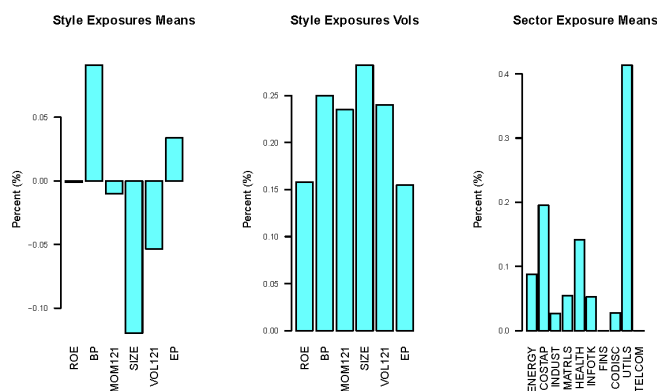



Figure 5: Dat145 GMV Long-Only Portfolio Style and Sector Exposures Means and Volatilities

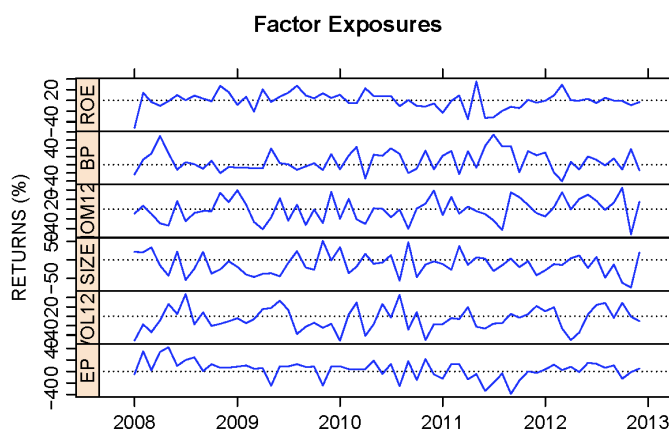


Figure 6: Dat145 GMV Long-Only Portfolio Style Exposures Time Series

```
repExposures(fit145, wts145, isPrint = FALSE, isPlot = TRUE,
  which = 2, add.grid = TRUE, scaleType = "same",
  layout = c(3, 3))
```

repReturn

The function `repReturn` 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

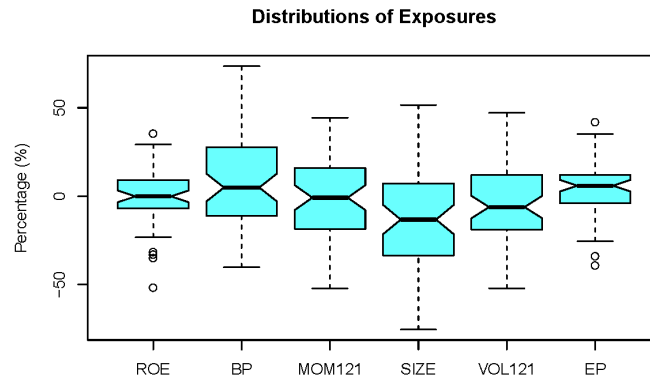


Figure 7: Dat145 GMV Long-Only Portfolio Style Exposures Boxplots

3. Time Series plot of portfolio sector returns

4. Boxplot of Portfolio Returns Components.

We illustrate their use on the FFM fitted object `fit145` with `wt145`. 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(fit145, wt145, isPlot = FALSE, digits = 2)
```

##		Mean	Volatility
##	PortRet	1.06	6.19
##	ResidRet	0.22	1.75
##	FacRet	0.84	5.58
##	ROE	0.01	0.07
##	BP	-0.10	0.55
##	MOM121	0.07	0.47
##	SIZE	0.03	0.41
##	VOL121	-0.02	0.61
##	EP	-0.10	0.36
##	CODISC	0.04	0.21
##	COSTAP	0.12	1.18
##	ENERGY	0.11	0.62
##	FINS	0.00	0.00
##	HEALTH	0.08	0.86

## INDUST	0.01	0.20
## INFOTK	0.04	0.40
## MATRLS	0.04	0.38
## TELCOM	0.00	0.00
## UTILS	0.51	2.13

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(fit145, wts145, isPrint = FALSE, isPlot = TRUE,
          which = 1, add.grid = TRUE, scaleType = "same")
```

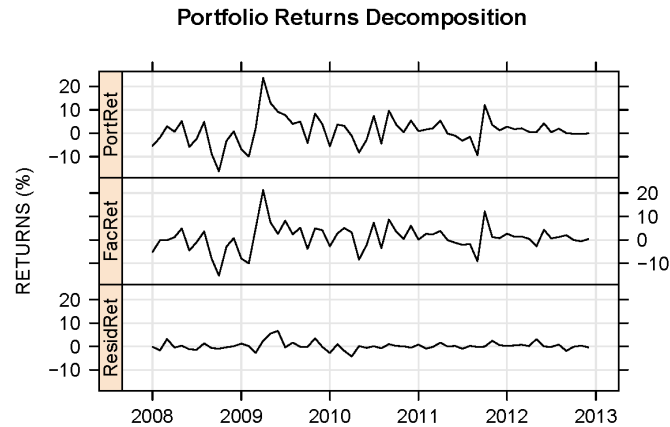


Figure 8: Fit145 Portfolio Factor Return and Specific Return Decomposition

```
repReturn(fit145, wts145, isPrint = FALSE, isPlot = TRUE,
          which = 2, add.grid = TRUE, zeroLine = T, color = "Blue",
          scaleType = "same")
```

```
repReturn(fit145, wts145, isPrint = FALSE, isPlot = TRUE,
          which = 3, add.grid = TRUE, zeroLine = T, color = "Blue",
          scaleType = "same")
```

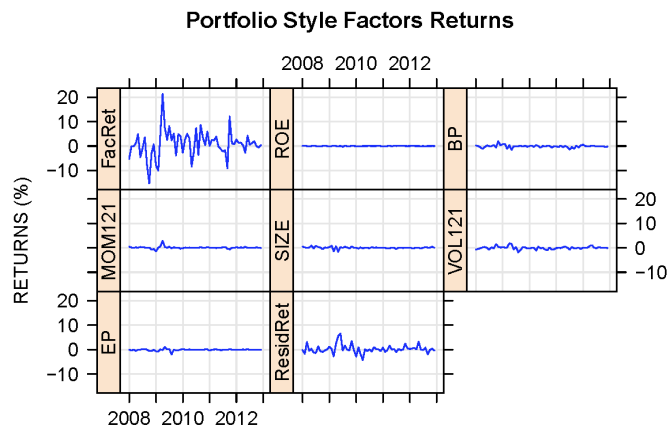


Figure 9: Fit145 Portfolio Style Factor Returns

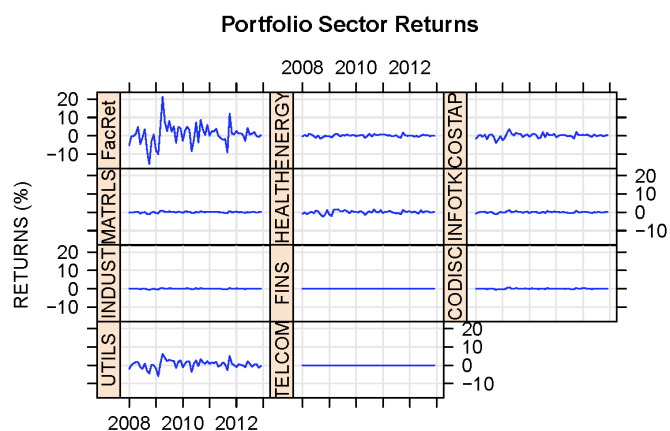


Figure 10: Fit145 Portfolio Sector Factor Returns

```
repReturn(fit145, wts145, isPrint = FALSE, isPlot = TRUE,
          which = 4)
```

fmXxDecomp

The functions `fmSdDecomp`, `fmEsDecomp` and `fmVaRDecomp` compute factor decompositions of risk for each of the assets used to fit a fundamental factor model with `fitFfm`, for the case of standard deviation/volatility (SD), value-at-risk (VaR) and expected shortfall (ES) risk measures, respectively. There are three decomposition types: factor percent contribution to risk (FPCR), factor contribution to risk (FCR) and factor marginal contribution to risk (FMCR). FPCR is the most commonly used by practitioners. The VaR and ES estimators can be computed using a

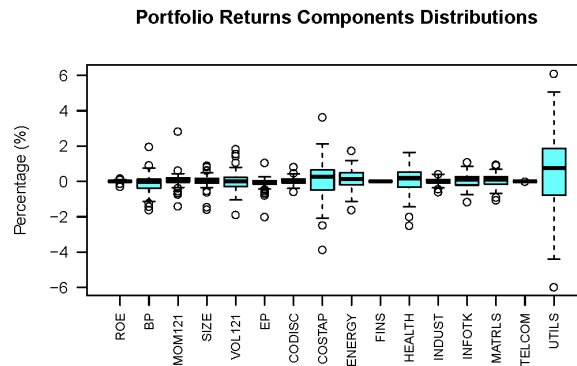


Figure 11: Fit145 Portfolio Return Components Boxplots

parametric normal distribution risk measure estimator or a non-parametric risk measure estimator, with the latter as default and with a tail probability default of 5% obtained by setting $p = .95$. We illustrate using the fitted model `fitDjia5YrStyle` with and intercept (alpha) and style factors only for the DJIA returns data set `dataDjia.5Yr`.

```
fitDjia5YrIntStyle = fitFfm(data = dataDjia.5Yr, exposure.vars = c("MKTCAP",
  "ENTVAL", "P2B", "EV2S", "SIZE"), date.var = "DATE",
  ret.var = "RETURN", asset.var = "TICKER", fit.method = "WLS",
  z.score = TRUE, addIntercept = T)
# Asset FPCR volatility decomposition
round(fmSdDecomp(fitDjia5YrIntStyle)$pcSd[1:3, ], digits = 1)
```

##	Alpha	MKTCAP	ENTVAL	P2B	EV2S	SIZE	Residuals
## AA	29.8	-2.0	-13.9	0.8	1.0	60.0	24.3
## AXP	23.7	0.9	-4.6	0.0	1.1	7.4	71.5
## BA	62.0	3.6	-21.1	18.8	8.3	20.9	7.4

```
p = 0.9
# Asset FPCR VaR decomposition: parametric normal
round(fmVaRDecomp(fitDjia5YrIntStyle, type = "normal",
  p = p)$pcVaR[1:3, ], digits = 1)
```

##	Alpha	MKTCAP	ENTVAL	P2B	EV2S	SIZE	Residuals
----	-------	--------	--------	-----	------	------	-----------

```
## AA    33.6    0.6  -13.7  0.4  0.0 53.8    25.3
## AXP   26.1    2.9   -3.9  0.1  2.2  4.5    68.1
## BA    63.7    7.0  -18.3 20.6  5.3 14.7     6.9

# Asset FPCR VaR decomposition: non-parametric
round(fmVaRDecomp(fitDjia5YrIntStyle, p = p)$pcVaR[1:3,
], digits = 1)

##      Alpha MKTCAP ENTVAL  P2B EV2S SIZE Residuals
## AA    35.1    3.6  -14.7  0.5  1.6 50.6    23.4
## AXP   52.2    7.2   -6.2  0.1 -0.8  4.6    42.8
## BA    69.2    8.9  -22.1 19.6  2.1 13.9     8.4

# Asset FPCR ES decomposition: parametric normal
round(fmEsDecomp(fitDjia5YrIntStyle, type = "normal",
p = p)$pcES[1:3, ], digits = 1)

##      Alpha MKTCAP ENTVAL  P2B EV2S SIZE Residuals
## AA    62.2   20.0  -11.7 -3.2 -7.4  6.8    33.4
## AXP   36.1   11.5  -0.9  0.3  6.8 -7.5    53.7
## BA    70.1   20.2  -7.5 27.7 -6.2 -9.4     5.1

# Asset FPCR ES decomposition: non-parametric
round(fmEsDecomp(fitDjia5YrIntStyle, p = p)$pcES[1:3,
], digits = 1)

##      Alpha MKTCAP ENTVAL  P2B EV2S SIZE Residuals
## AA     7.1   -9.2  -17.2  3.8  4.9 81.1    29.4
## AXP   34.5  -12.7  -20.1 -0.7 -11.5 45.3    65.1
## BA    32.3  -29.0  -64.3 -18.9 27.4 90.8    61.7
```

It is a useful exercise to fit the fundamental factor model with a decrease in tail probability from 10% to 5% by setting $p = .95$.

portXxDecomp

The functions `portSdDecomp`, `portVaRDecomp` and `portEsDecomp` compute FPCR, FCR and FMCR factor decompositions of portfolio risk in a manner similar to the way the `fmXxDecomp` functions do that for the assets in a fundamental factor model. The main difference is that the decomposition now depends on the portfolio weights. We use the fitted model `fitDjia5YrIntStyle` used for the preceding `fmXxDecomp` examples, along with the DJIA GMV long-only portfolio weights `wtsDjia`, to illustrate the portfolio factor risk decompositions.

```
# Portfolio FPCR volatility decomposition
```

```
wts = wtsDjia
```

```
round(portSdDecomp(fitDjia5YrIntStyle, wts)$pcSd, digits = 1)
```

##	Alpha	MKTCAP	ENTVAL	P2B	EV2S	SIZE	Residuals
##	95.3	-3.3	18.2	4.4	-0.1	-27.3	12.9

```
p = 0.9
```

```
# Portfolio FPCR VaR decomposition: parametric
```

```
# normal
```

```
round(portVaRDecomp(fitDjia5YrIntStyle, wts, type = "normal",  
  p = p)$pcVaR, digits = 1)
```

##	Alpha	MKTCAP	ENTVAL	P2B	EV2S	SIZE	Residuals
##	92.5	-10.3	13.2	5.7	0.2	-12.1	10.7

```
# Portfolio FPCR VaR decomposition: non-parametric
```

```
round(portVaRDecomp(fitDjia5YrIntStyle, wts, p = p)$pcVaR,  
  digits = 1)
```

##	Alpha	MKTCAP	ENTVAL	P2B	EV2S	SIZE	Residuals
##	94.6	-17.2	1.9	7.2	0.8	11.7	1.0

```
# Asset FPCR ES decomposition: parametric normal
```

```
round(portEsDecomp(fitDjia5YrIntStyle, wts, type = "normal",  
  p = p)$pcES, digits = 1)
```

##	Alpha	MKTCAP	ENTVAL	P2B	EV2S	SIZE	Residuals
##	85.9	-26.8	1.6	9.0	0.8	24.0	5.5


```
# Asset FPCR ES decomposition: non-parametric
round(portEsDecomp(fitDjia5YrIntStyle, wts, p = p)$pcES,
      digits = 1)
```

##	Alpha	MKTCAP	ENTVAL	P2B	EV2S	SIZE	Residuals
##	38.0	-38.1	-17.7	16.2	1.6	72.3	27.8

repRisk

The function `repRisk` combines the computation results of `fmXxDecomp` and `portXxDecomp` into a single report. We illustrate for using the `fitDjia5YrIntStyle` fit and `wts = wtsDjia` for the choice of standard deviation risk and the choice of non-parametric ES risk with tail probability 10%.

```
repVaR = repRisk(fitDjia5YrIntStyle, wts, risk = "VaR",
  decomp = "FPCR", type = "normal", p = p)
repEs = repRisk(fitDjia5YrIntStyle, wts, risk = "ES",
  decomp = "FPCR", type = "normal", p = p)
repVaR$FPCR[1:4, ]

## NULL

repEs$FPCR[1:4, ]

## NULL
```

Comparison with the corresponding results obtained by `fmSdDecomp`, `fmEsDecomp`, `portSdDecomp` and `portEsDecomp` in the preceding two subsection confirm that `repRisk` combines results from the `fmXxDecomp` and `portXxDecomp` functions.

4 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

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

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 $\hat{\mathbf{f}}_t$, $t = 1, 2, \dots, T$, $N \times 1$ dimensional residuals $\hat{\boldsymbol{\varepsilon}}_t$, $t = 1, 2, \dots, T$, 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}_m^s , $m = 1, 2, \dots, M$ as follows.

1. A set of M simulated factor returns \mathbf{f}_m^s , $m = 1, 2, \dots, M$ are generated as M simple bootstrap samples of $\hat{\mathbf{f}}_t$, $t = 1, 2, \dots, T$, i.e., by sampling the factor returns estimates with replacement.
2. A set of M simulated residual $\boldsymbol{\varepsilon}_m^s$, $m = 1, 2, \dots, M$ are generated in one of the following two ways:

¹The 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.

- (a) By forming M simple bootstrap samples of the $\hat{\epsilon}_t$, $t = 1, 2, \dots, T$, or
- (b) By estimating a parameteric distribution for the $\hat{\epsilon}_t$, $t = 1, 2, \dots, T$ 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

$$\mathbf{r}_m^s = \mathbf{B}_T \mathbf{f}_m^s + \boldsymbol{\epsilon}_m^s \quad m = 1, 2, \dots, M. \quad (4)$$

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:

```
library(factorAnalytics)
data("factorDataSetDjia5Yrs")
N = 30
exposure.vars <- c("P2B", "MKTCAP", "SECTOR")
fit.ffm = fitFfm(data = factorDataSetDjia5Yrs, asset.var = "TICKER",
  ret.var = "RETURN", date.var = "DATE", exposure.vars = exposure.vars)
```

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

```
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 30 DJIA stocks over the five-year period are well represented by the set of 500 simulated sets of 30 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`.

```
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.

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

##      AA      AXP      BA      BAC      CAT      CSCO      CVX      DD      DIS      GE
## -0.012  0.012  0.004  0.000  0.014 -0.001  0.007  0.008  0.011  0.000

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

##      AA      AXP      BA      BAC      CAT      CSCO      CVX      DD      DIS      GE
## -0.015  0.013  0.003  0.006  0.013 -0.004  0.006  0.010  0.011 -0.002
```

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

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

##      AA      AXP      BA      BAC      CAT      CSCO      CVX      DD      DIS      GE
## 0.137 0.149 0.089 0.196 0.126 0.091 0.064 0.093 0.078 0.109

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

##      AA      AXP      BA      BAC      CAT      CSCO      CVX      DD      DIS      GE
## 0.144 0.144 0.089 0.193 0.129 0.092 0.062 0.093 0.075 0.106
```

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 30 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.ffmpeg$residuals),
  2, mean, na.rm = T)
resid.sd = matrix(sqrt(fit.ffmpeg$resid.var))
resid.par = cbind(resid.mean, resid.sd)
fmmcDatNormal = fmmcSemiParam(factor.ret = fit.ffmpeg$factor.returns,
  beta = fit.ffmpeg$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	AXP	BA	BAC	CAT	CSCO	CVX	DD	DIS	GE
##	-0.012	0.012	0.004	0.000	0.014	-0.001	0.007	0.008	0.011	0.000

```

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

```

	AA	AXP	BA	BAC	CAT	CSCO	CVX	DD	DIS	GE
##	-0.012	0.015	0.004	0.001	0.012	-0.001	0.005	0.008	0.011	-0.001

```

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

```

	AA	AXP	BA	BAC	CAT	CSCO	CVX	DD	DIS	GE
##	0.137	0.149	0.089	0.196	0.126	0.091	0.064	0.093	0.078	0.109

```

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

```

	AA	AXP	BA	BAC	CAT	CSCO	CVX	DD	DIS	GE
##	0.126	0.131	0.088	0.147	0.100	0.087	0.066	0.115	0.065	0.092

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 eight of the first 10 stocks, sometimes substantially so. However, comparison of the volatilities for all 30 stocks reveals that there are only 13 stocks for which the volatilities for the

simulated returns are smaller than those of the actual returns, and that when the volatilities of the simulated returns are larger than those of the actual returns they are much larger, for example .093 versus .065 for CAT and .132 versus .068 for HD. 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.

5 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.

Market + Industry Model

The market plus industry sector model has the form

$$\begin{aligned} \mathbf{r}_t &= \begin{pmatrix} \mathbf{1} & \mathbf{B}_i \end{pmatrix} \mathbf{f}_{mi,t} + \boldsymbol{\varepsilon}_t, \quad t = 1, 2, \dots, T \\ &= \mathbf{B}_{mi} \mathbf{f}_{mi,t} + \boldsymbol{\varepsilon}_t \end{aligned} \quad (5)$$

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

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

It follows 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 $f_{1,t}, f_{2,t}, \dots, f_{K,t}$ as deviations from the market return. Thus we impose a constraint:

$$f_{1,t} + f_{2,t} + \dots + f_{K,t} = 0. \quad (7)$$

This can be accomplished with the reparameterization

$$\mathbf{f}_{mi,t} = \mathbf{R}_{mi} \mathbf{g}_{mi,t} \quad (8)$$

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

where

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

$$\mathbf{a}' = (0, -1, -1, \dots, -1) \sim K \times 1 \quad (11)$$

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

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 `fitFfm` 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.

```
dat = factorDataSetDjia5Yrs
fitSec = fitFfm(dat, asset.var = "TICKER", ret.var = "RETURN",
  date.var = "DATE", exposure.vars = "SECTOR")
round(coef(summary(fitSec)$sum.list[[1]])[, 1], 3)

## CODISC COSTAP ENERGY   FINS HEALTH INDUST INFOTK MATRLS TELCOM
## -0.010 -0.025 -0.093   0.002 -0.090 -0.042 -0.106 -0.035 -0.083
```

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

##          CODISC COSTAP ENERGY  FINS HEALTH INDUST INFOTK MATRLS TELCOM
## 2008-01-01 -0.01 -0.025 -0.093 0.002 -0.09 -0.042 -0.106 -0.035 -0.083
```

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 `fitF` 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.

```
fitSecInt = fitFfm(dat, asset.var = "TICKER", ret.var = "RETURN",
  date.var = "DATE", exposure.vars = "SECTOR", addIntercept = T)
round(coef(summary(fitSecInt)$sum.list[[1]]), 1,
  2)

##      g1      g2      g3      g4      g5      g6      g7      g8      g9
## -0.05  0.04  0.02 -0.04  0.07 -0.03  0.01 -0.05  0.02

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

##          Market CODISC COSTAP ENERGY FINS HEALTH INDUST INFOTK MATRLS
## 2008-01-01 -0.05  0.04  0.02 -0.04 0.07 -0.03  0.01 -0.05  0.02
##          TELCOM
## 2008-01-01 -0.03

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 (9 of them since there are 9 sectors). On the other hand the last line of code above prints the 10 factor returns coefficients $\hat{\mathbf{f}}_{mi,1} = (f_{0,1}, f_{1,1}, \dots, f_{9,1})$ for month 1, the first one for the

market and the other 9 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

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

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:

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

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:

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

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

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

$$f_{mi,1} + f_{mi,2} + \dots + f_{mi,K_2} = 0 \quad (17)$$

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

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

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

$$f_{c,1} + f_{c,2} + \dots + f_{c,K_3} = 0. \quad (19)$$

which can be implemented by setting

$$\mathbf{f}_c = \mathbf{R}_c \mathbf{g}_c \quad (20)$$

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

with the country restriction matrix

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

$$\mathbf{b}' = (-1, -1, \dots, -1) \sim 1 \times (K_3 - 1). \quad (23)$$

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 30 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.

```

# 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 12.

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

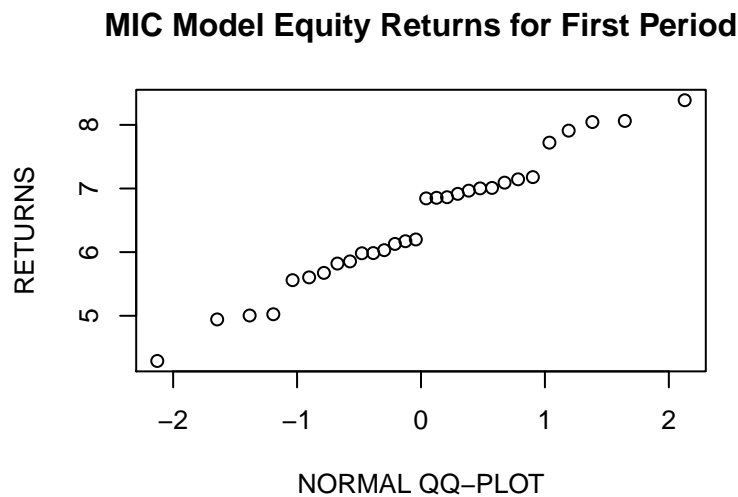


Figure 12: Normal QQ-Plot MIC Model Asset Returns First Period

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)
fit$factor.returns

##           Market      SEC1      SEC2      SEC3      India      US
## 2000-01-01 6.475849 -1.0046757 -0.061937902 1.0666136 -0.4967897 0.4967897
## 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

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

where the modified exposures matrices

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

are full rank.

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

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

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

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

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

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

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

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

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.cov` 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.