Fitting Statistical Factor Models: factorAnalytics vignette

Sangeetha Srinivasan

May 28, 2015

Abstract

The purpose of this vignette is to demonstrate the use of `fitSfm` and related control, analysis and plot functions in the `factorAnalytics` package.

Contents

1 Overview

1.1 Load Package

The latest version of the `factorAnalytics` package can be downloaded from R-forge through the following command:

```
install.packages("factorAnalytics", repos="http://R-Forge.R-project.org")
```

Load the package and it's dependencies.

```
library(factorAnalytics)
options(digits=3)
```

1.2 Summary of related functions

Here's a list of the functions and methods demonstrated in this vignette:

- `fitSfm(data, k, max.k, refine, sig, check, corr, ...)`: Fits a statistical factor model for one or more asset returns using Principal Component Analysis (PCA). When the number of assets exceeds the number of time periods, Asymptotic Principal Component Analysis (APCA) is performed. Additionaly for APCA, user can specify a method (one of Connor and Korajczyk (1993) or Bai and Ng (2002)) to determine the number of factors and/or choose to use the Connor and Korajczyk (1988) refinement to the APCA procedure. The returned object is of class "sfm" and contains the fitted "lm" object, estimated factor realizations, factor loadings, R-squared, residual volatility, factor model return covariance and the factor mimicking portfolio weights.
- `coef(object, ...)`: Extracts the coefficient matrix (intercept and factor betas) for all assets fit by the "sfm" object.
- `fitted(object, ...)`: Returns an "xts" data object of fitted asset returns from the factor model for all assets.
- `residuals(object, ...)`: Returns an "xts" data object of residuals from the fitted factor model for all assets.
- `fmCov(object, use, ...)`: Returns the `N x N` symmetric covariance matrix for asset returns based on the fitted factor model. `"use"` specifies how missing values are to be handled.

- `fmSdDecomp(object, use, ...)`: Returns a list containing the standard deviation of asset returns based on the fitted factor model and the marginal, component and percentage component factor contributions estimated from the given sample. `"use"` specifies how missing values are to be handled.

- `fmVaRDecomp(object, p, method, invert, ...)`: Returns a list containing the value-at-risk for asset returns based on the fitted factor model and the marginal, component and percentage component factor contributions estimated from the given sample. `"p"` and `"method"` specify the confidence level and method (one of "modified","gaussian", "historical" or "kernel") to calculate VaR. VaR is by default a positive quantity and specifying `"invert=TRUE"` allows the VaR value to be expressed as a negative quantity. These 3 arguments, `"p"`, `"method"` and `"invert"` are passed on to the `VaR` function in the `PerformanceAnalytics` package to calculate VaR.

- `fmEsDecomp(object, p, method, invert, ...)`: Returns a list containing the expected shortfall for asset returns based on the fitted factor model and the marginal, component and percentage component factor contributions estimated from the given sample. Arguments `"p"`, `"method"` and `invert` are the same as above.

- `plot(x)`: The `plot` method for class "sfm" can be used for plotting factor model characteristics of a group of assets (default) or an individual asset. The user can select the type of plot either from the menu prompt or directly via argument `which`. In case multiple plots are needed, the menu is repeated after each plot (enter 0 to exit). User can also input a numeric vector of plot options via `which`.

- `predict(object, newdata, ...)`: The `predict` method for class "sfm" returns a vector or matrix of predicted values for a new data sample or simulated values.

- `summary(object, se.type, n.top, ...)`: The `summary` method for class "sfm" returns an object of class `"summary.sfm"` containing the summaries of the fitted "lm" objects, factor loadings, residual volatilities (under the homo-skedasticity assumption), R-squared values and factor mimicking portfolio weights. Printing the factor model summary object outputs the call, coefficients (with standard errors and t-statistics), r-squared and residual volatility for all assets, and top long and short positions for each factor mimicking portfolio.

1.3 Data

The examples in this vignette primarily use the `StockReturns` dataset available in the `factorAnalytics` package. It contains two "data.frame" objects, `r.M` and `r.W`.

```
# load the Rdata object
data(StockReturns)
# view class and dimensions
class(r.M)

## [1] "data.frame"

dim(r.M)

## [1] 120  15

# variable names
colnames(r.M)

##  [1] "CITCRP" "CONED"  "CONTIL" "DATGEN" "DEC"    "DELTA"  "GENMIL"
##  [8] "GERBER" "IBM"    "MOBIL"  "PANAM"  "PSNH"   "TANDY"  "TEXACO"
## [15] "WEYER"

# range of observations
range(rownames(r.M))

## [1] "1978-01-01" "1987-12-01"
```

The `r.M` data object, originally used in Berndt (1991), has 120 observations of 15 variables (U.S. stock returns) and the frequency is monthly.

```
class(r.W)

## [1] "data.frame"

dim(r.W)

## [1]  182 1618

range(rownames(r.W))

## [1] "1997-01-08" "2000-06-28"
```

Whereas, the `r.W` data.frame object has 182 observations of 1618 variables (U.S. stock returns) and the frequency is weekly.

The yield curve example in section 3, uses the `TreasuryYields` dataset in the `factorAnalytics` package. This containes an "xts" data object `tr.yields`. The data was obtained as a "txt" file from the companion website to Ruppert (2010).

```
data(TreasuryYields)
head(tr.yields)

##            X1mo X3mo X6mo X1yr X2yr X3yr X5yr X7yr X10yr X20yr X30yr
## 1990-01-02   NA 7.83 7.89 7.81 7.87 7.90 7.87 7.98  7.94    NA  8.00
## 1990-01-03   NA 7.89 7.94 7.85 7.94 7.96 7.92 8.04  7.99    NA  8.04
## 1990-01-04   NA 7.84 7.90 7.82 7.92 7.93 7.91 8.02  7.98    NA  8.04
## 1990-01-05   NA 7.79 7.85 7.79 7.90 7.94 7.92 8.03  7.99    NA  8.06
## 1990-01-08   NA 7.79 7.88 7.81 7.90 7.95 7.92 8.05  8.02    NA  8.09
## 1990-01-09   NA 7.80 7.82 7.78 7.91 7.94 7.92 8.05  8.02    NA  8.10

range(index(tr.yields))

## [1] "1990-01-02" "2008-10-31"
```

The "xts" data object, `tr.yields`, contains yields on Treasury bonds at 11 maturities, T = 1, 3, and 6 months and 1, 2, 3, 5, 7, 10, 20, and 30 years. Daily yields were taken from a U.S. Treasury website for the time period January 2, 1990, to October 31, 2008.

Daily yields are missing from some values of T. For example, the 20-year constant maturity series were discontinued at the end of calendar year 1986 and reinstated on October 1, 1993. Omitting the missing values of the differenced data, leaves 819 days of observations. Excluding the one-month and 20-year yields would leave us with a longer series.

2 Fit a statistical factor model

In statistical factor models, factor realizations are not directly observable (unlike times series factor models) and the factor loadings are not known (unlike some fundamental factor models). Both factors and betas must be extracted from the returns data using statistical methods such as factor analysis or Principal Component Analysis (PCA). PCA uses the eigen decomposition of the covariance (or correlation) matrix of asset returns to find the first `K` principal components that explain the largest portion of the sample covariance matrix of returns. Factor loadings are then estimated using time series regression. Factor analysis involves maximum likelihood optimization to estimate the factor loadings and the residual covariance matrix, constructing the factor realizations and choosing a rotation of the coordinate system for a more meaningful interpretation of the factors.

In `fitSfm`, PCA is applied to extract the factor realizations when the number of time series observations, T, is greater than the number of assets, N. When $N > T$, the sample covariance matrix for asset returns is singular and Aymptotic principal Component Analysis (APCA) due to Connor and Korajczyk (1988) is performed.

Let's take a look at the arguments for `fitSfm`.

```
args(fitSfm)

## function (data, k = 1, max.k = NULL, refine = TRUE, sig = 0.05,
##     check = FALSE, corr = FALSE, ...)
## NULL
```

A time series of asset returns is input via argument `data`. If `data` is not of class "xts", rownames must provide an "xts" compatible time index. Specifying `check=TRUE`, issues a warning if any asset is found to have identical observations. Note that before model fitting, incomplete cases in `data` are removed.

For both PCA and APCA, any number of factors less than `min(N,T)` can be chosen explicitly via argument `k`. Alternately for APCA, a method to determine the number of factors can be specified: `k="bn"` corresponds to Bai and Ng (2002) and `k="ck"` corresponds to Connor and Korajczyk (1993). User can specify the maximum number of factors, `max.k` to consider with these methods. If not, it is assumed to be either ten or $T - 1$, whichever is smaller.

For the "ck" method, `sig` specifies the desired level of significance. Argument `refine` specifies whether a refinement of the APCA procedure from Connor and Korajczyk (1988) that may improve efficiency is to be used.

When `corr=TRUE`, the correlation matrix of returns are used for finding the principal components instead of the covariance matrix. This is typically decided by practioners on a case-by-case

basis. The variable with the highest variance dominates the PCA when the covariance matrix is used. However, this may be justified if a volatile asset is more interesting for some reason and volatility information shouldn't be discarded. On the other hand, using the correlation matrix standardizes the variables and makes them comparable, avoiding penalizing variables with less dispersion.

Finally, if the median of the 1st principal component is negative, all it's factor realizations are automatically inverted to enable more meaningful interpretation.

2.1 Principal Component Analysis

Refer to chapter 15 of Zivot and Jia-hui (2006) for a description of the procedure used.

The following example fits a statistical factor model with two principal components for `r.M`, the monthly returns on fifteen U.S. stocks described in section 1. Since the number of observations is larger than the number of assets in this case, `fitSfm` would choose to perform PCA.

```
fit.pca <- fitSfm(r.M, k=2)
```

The resulting object, `fit.pca`, has the following attributes.

```
class(fit.pca)

## [1] "sfm"

names(fit.pca)

##  [1] "asset.fit"   "k"           "factors"     "loadings"    "alpha"
##  [6] "r2"          "resid.sd"    "residuals"   "Omega"       "eigen"
## [11] "mimic"       "call"        "data"        "asset.names"
```

The component `k` contains the number of factors, either as input or determined by "ck" or "bn" methods. The N (or, T for APCA) eigenvalues of the sample covariance matrix are in `eigen`. The $T \times K$ "xts" object of estimated factor realizations are in `factors`.

The component `asset.fit` contains an object of class "mlm" or "lm" from the time-series OLS regression of asset returns on estimated factors. The estimated factor loadings[1] are in `loadings` and regression alphas are in `alpha`. The $T \times N$ "xts" object of residuals from the OLS regression are in `residuals`. R-squared and residual standard deviations are in `r2` and `resid.sd` respectively.

[1] Refer to the summary method in section 2.3 for standard errors, degrees of freedom, t-statistics etc.

The $N \times N$ return covariance matrix estimated by the factor model is in `Omega`[2]. The $N \times K$ matrix of factor mimicking portfolio weights are given in `mimic`[3]. The remaining components contain the input choices and the data. The fitted factor model is printed below.

```
fit.pca # print the fitted "sfm" object

##
## Call:
## fitSfm(data = r.M, k = 2)
##
## Model dimensions:
## Factors  Assets Periods
##       2      15     120
##
## Factor Loadings:
##       F.1              F.2
##  Min.   :0.044   Min.   :-0.824
##  1st Qu.:0.139   1st Qu.:-0.067
##  Median :0.250   Median : 0.012
##  Mean   :0.231   Mean   :-0.002
##  3rd Qu.:0.308   3rd Qu.: 0.142
##  Max.   :0.417   Max.   : 0.365
##
## R-squared values:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.031   0.214   0.427   0.398   0.573   0.925
##
## Residual Volatilities:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0415  0.0538  0.0721  0.0693  0.0779  0.1090
```

[2]Section 4.1 on Factor model Covariance gives a detailed derivation.

[3]The summary method in section 2.3 helps to make this more tangible by summarizing the largest and smallest weights for each factor mimicking portfolio.

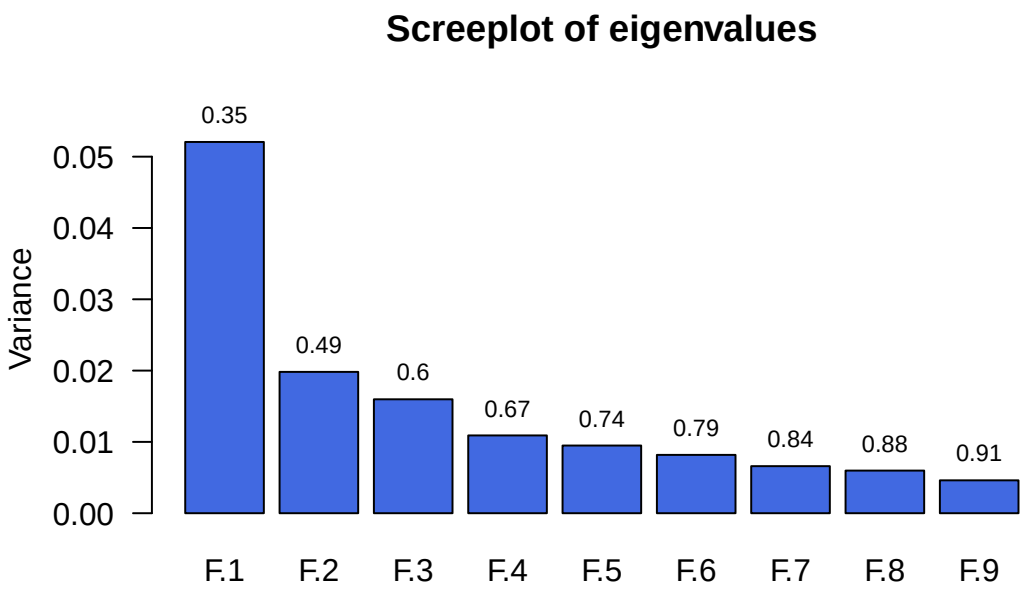

Figure 1: Screeplot of eigenvalues: fit.pca

The screeplot of eigenvalues is illustrated in Figure 1 (option 1 on the plot menu; refer to section 5 for a list of all the options). The first principal component explains about thirty five percent of the total variance, and the first two components explain about half of the total variance. Specifying `eig.max=0.9` displays the first set of components that explain at least ninety percent of the total variance.

```
plot(fit.pca, which=1, eig.max=0.9)
```

The time series of estimated factor returns are displayed in Figure 2.

```
plot(fit.pca, which=2)
```

The estimated factor loadings for all assets are shown in Figure 3. Note that the first factor has all positive loadings. The second factor has both positive and negative loadings.

```
plot(fit.pca, which=3, a.sub=1:15)
```

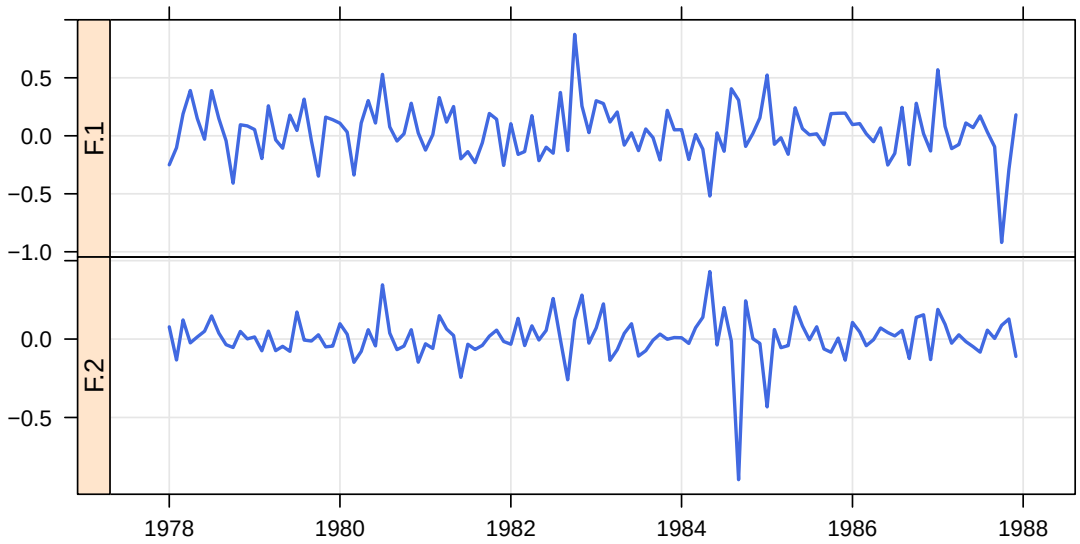

Figure 2: Time series of estimated factors: fit.pca

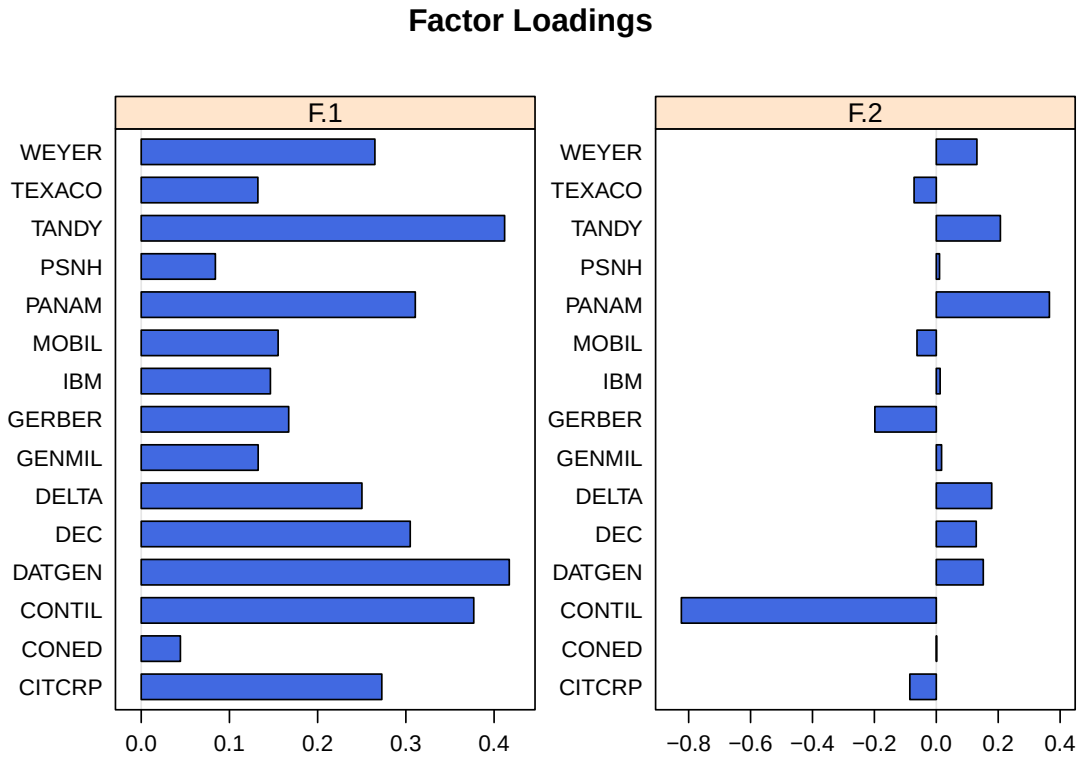

Figure 3: Estimated factor loadings: fit.pca

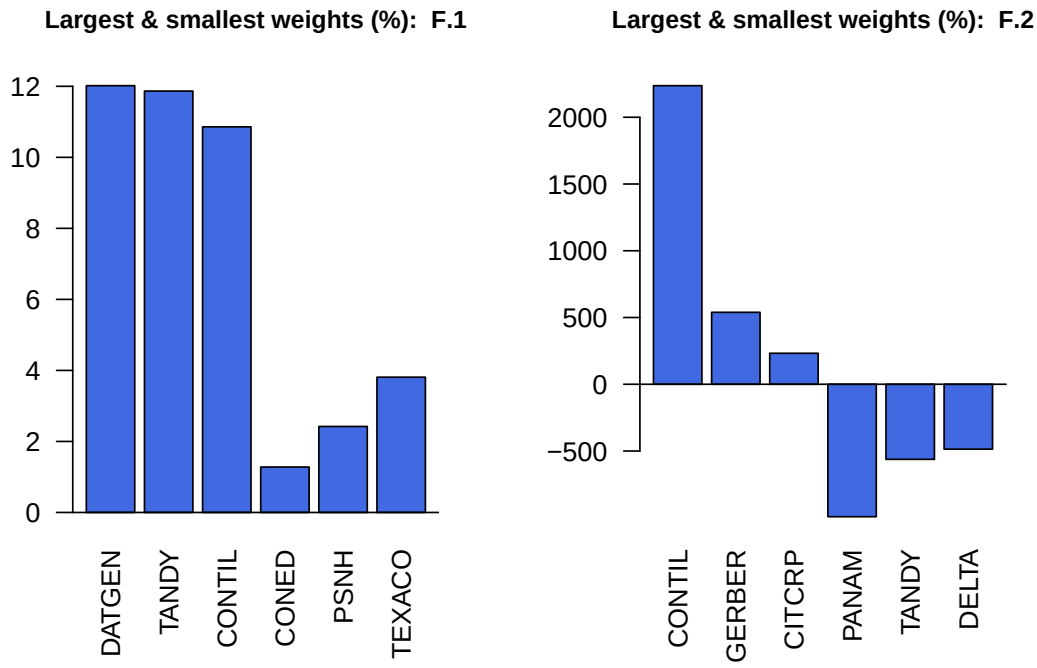

Figure 4: Top 3 largest and smallest weights in the factor mimicking portfolios

Figure 4 displays the top three (`n.top`) assets with the largest and smallest weights in each factor mimicking portfolio. For the first factor, assets DATGEN, TANDY and CONTIL have the highest weights and assets CONED, PSNH and TEXACO have the lowest weights. Since all the weights in the first portfolio are positive, this might be construed as a market-wide factor.

```
# Factor mimicking portfolio weights from PCA fit
t(fit.pca$mimic)

##     CITCRP   CONED CONTIL DATGEN     DEC   DELTA  GENMIL GERBER     IBM
## F.1 0.0786  0.0128  0.109   0.12  0.0878  0.0721  0.0382 0.0482  0.0422
## F.2 2.3217 -0.0324 22.365  -4.12 -3.5049 -4.8626 -0.4649 5.3906 -0.3367
##      MOBIL   PANAM    PSNH  TANDY TEXACO   WEYER
## F.1 0.0447  0.0895  0.0242  0.119 0.0381  0.0763
## F.2 1.6931 -9.9234 -0.2840 -5.624 1.9488 -3.5637

plot(fit.pca, which=12, n.top=3)
```

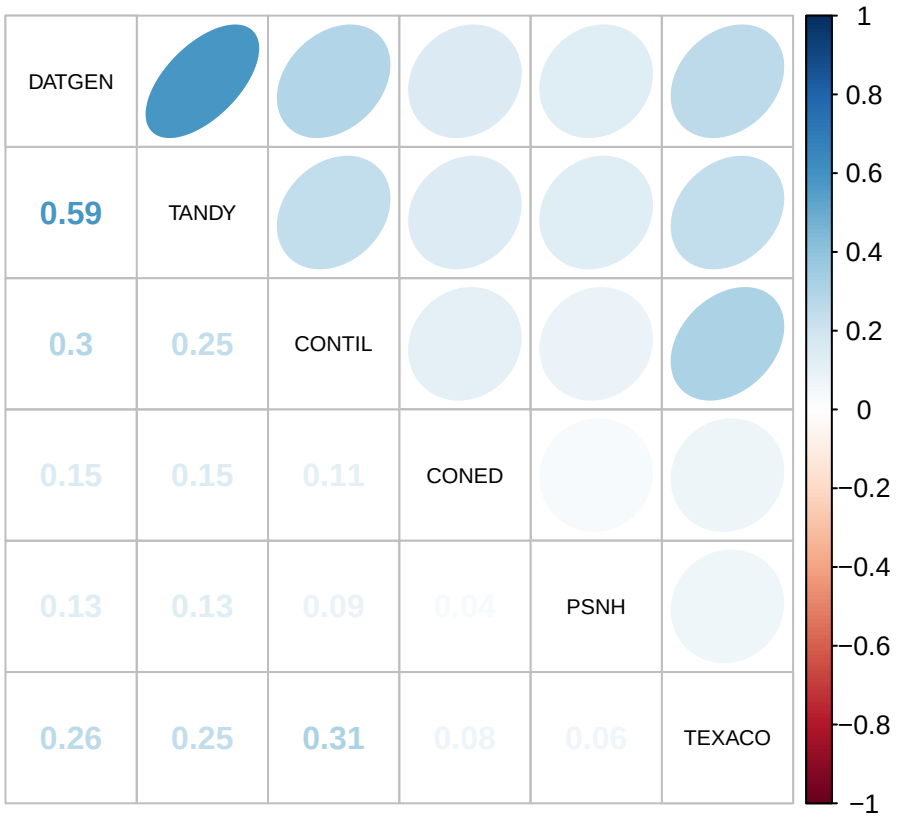

Figure 5: Correlations between assets with the top 3 largest and smallest positions in the F.1's factor mimicking portfolio

Figure 5 gives the correlations between assets with `n.top` largest and smallest weights in the factor mimicking portfolio for the first principal component.

```
plot(fit.pca, which=13, f.sub=1, n.top=3)
```

2.2 Asymptotic Principal Component Analysis

The following example fits a statistical factor model with two principal components for `r.W`, the weekly returns on 1618 U.S. stocks described in section 1. Since the number of observations is smaller than the number of assets in this case, `fitSfm` would choose to perform APCA. The primary difference is that the $T \times T$ covariance matrix is used instead. Refer to chapter 15 of Zivot and Jia-hui (2006) for a description of the procedure used.

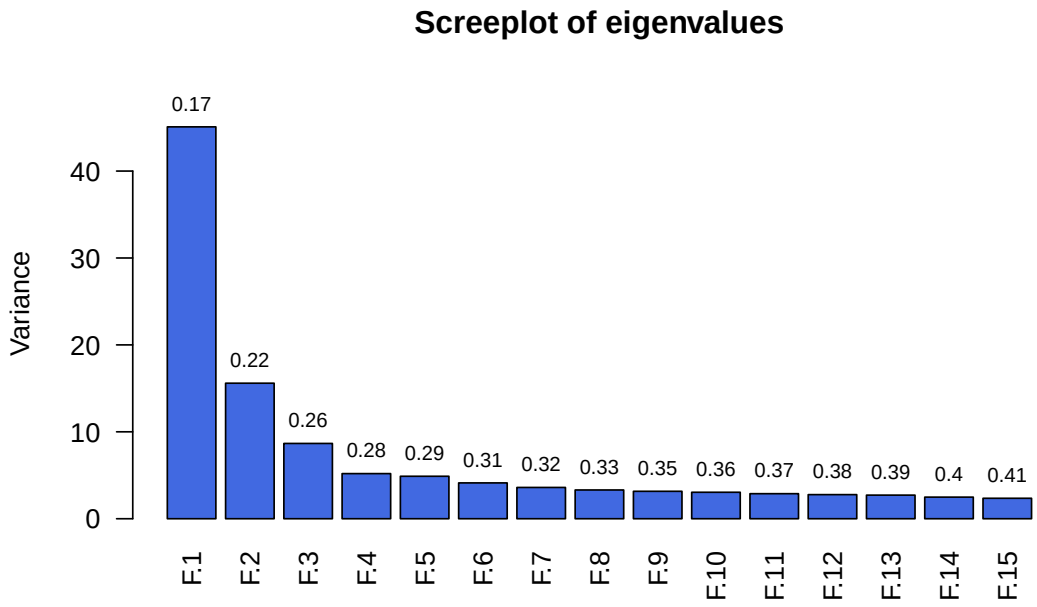

Figure 6: Screeplot of eigenvalues: fit.apca

```
fit.apca <- fitSfm(r.W, k=15)
```

Since the optional argument `refine=TRUE` by default, the APCA refinement will be used. This procedure involves recaling the returns using the residual variances obtained from one iteration of the APCA procedure, recomputing the $T \times T$ covariance matrix and performing a second iteration of the APCA procedure using this covariance matrix. This refinement may improve efficiency.

Figures 6 and 7 give the screeplot of eigenvalues and the estimated time series of the first 4 factor realizations respectively.

```
plot(fit.apca, which=1, eig.max=0.4, las=2)
```

```
plot(fit.apca, f.sub=1:4, which=2)
```

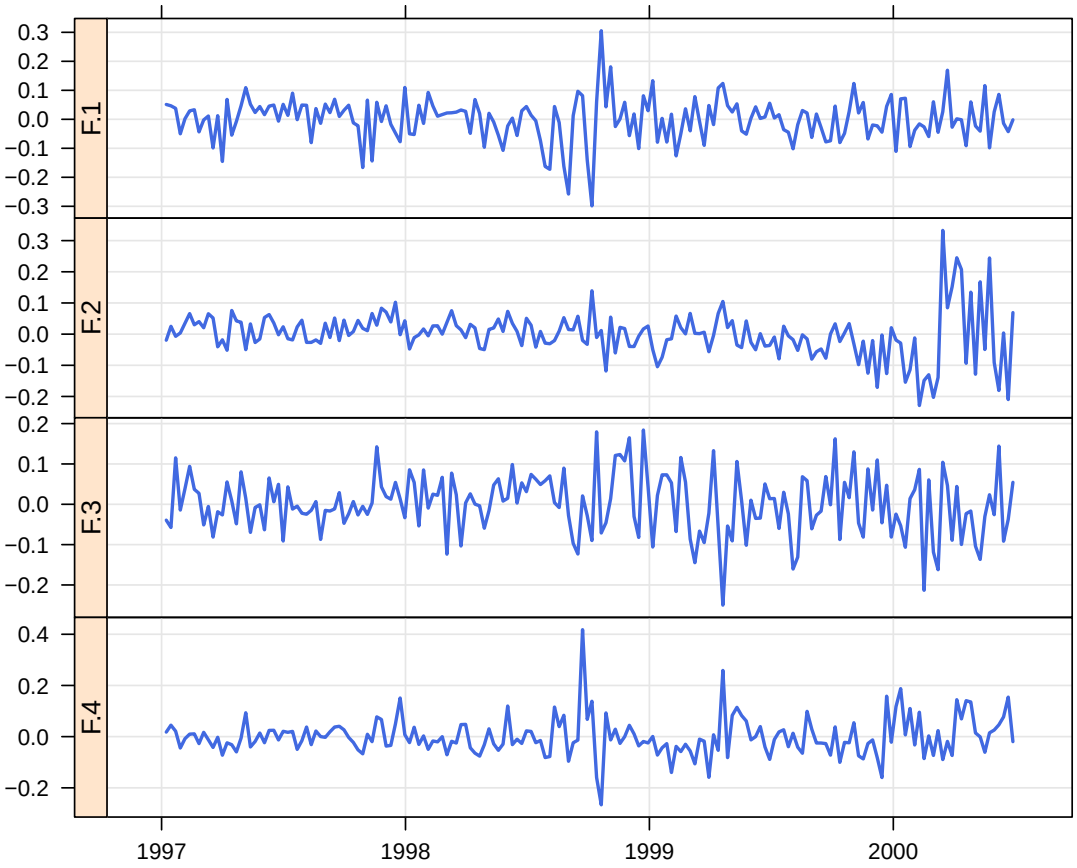

Figure 7: First four factor returns: fit.apca

Note that the number of factors was known or pre-specified in fit.apca above. In practice, the number of factors is unknown and must be determined from the data. Two such procedures are available via `fitSfm` via the argument `k`: `"bn"` corresponds to Bai and Ng (2002) and `"ck"` corresponds to Connor and Korajczyk (1993). The maximum number of factors to be considered with these methods is specified via `max.k`. By default, it is assumed to be either ten or $T - 1$, whichever is smaller. For the "ck" method, `sig` specifies the desired level of significance.

Here are some examples using the "ck" or "bn" method for performing APCA with the weekly return data for 1618 U.S. stocks. We find that both these methods select 2 factors and hence output the same factor model in this case.

```
# APCA with the Bai & Ng method
fit.apca.bn <- fitSfm(r.W, k="bn")
summary(fit.apca.bn$loadings)

##       F.1               F.2
##  Min.   :-0.177   Min.   :-1.119
##  1st Qu.: 0.218   1st Qu.:-0.181
##  Median : 0.317   Median : 0.005
##  Mean   : 0.332   Mean   :-0.078
##  3rd Qu.: 0.419   3rd Qu.: 0.096
##  Max.   : 0.950   Max.   : 0.411

# APCA with the Connor-Korajczyk method
fit.apca.ck <- fitSfm(r.W, k="ck", sig=0.05)
fit.apca.ck$k

## [1] 2
```

Finally, since the number of assets is large, it helps to look at the histograms of R-squared values and residual volatilities for all assets as shown in Figures 8 and 9 respectively.

```
plot(fit.apca, which=4, legend.loc="topright")
```

```
plot(fit.apca, which=5, legend.loc="topright")
```

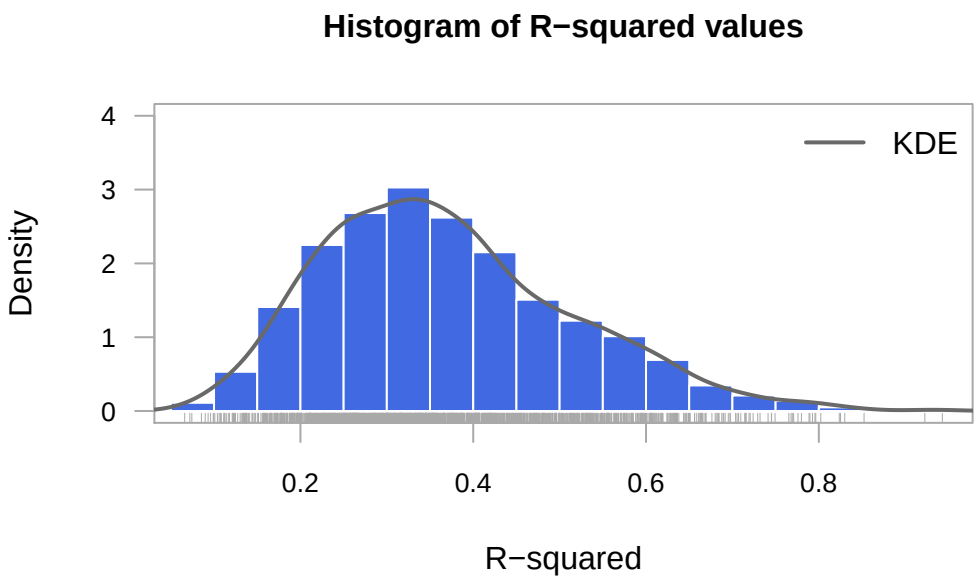

Figure 8: Histogram of R-squared values: fit.apca

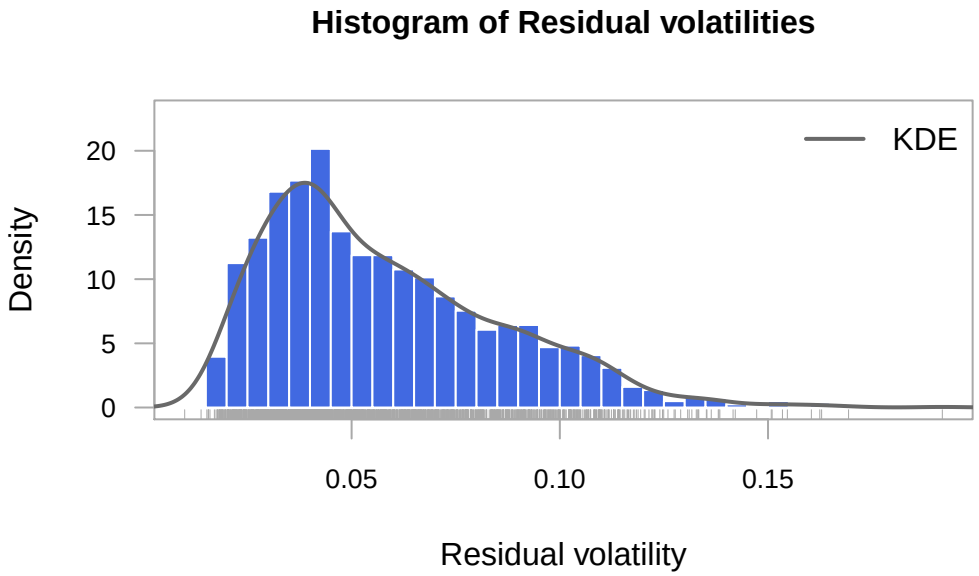

Figure 9: Histogram of Residual volatilities: fit.apca

2.3 S3 generic methods

```
methods(class="sfm")

##  [1] coef        fitted      fmCov       fmEsDecomp  fmSdDecomp
##  [6] fmVaRDecomp plot        predict     print       residuals
## [11] summary
## see '?methods' for accessing help and source code
```

Many useful generic accessor functions are available for "sfm" fit objects. `coef()` returns a matrix of estimated model coefficients including the intercept. `fitted()` returns an xts data object of the component of asset returns explained by the factor model. `residuals()` returns an "xts" data object with the component of asset returns not explained by the factor model. `predict()` uses the fitted factor model to estimate asset returns given a set of new or simulated factor return data.

```
## S3 method for class "sfm"
summary.sfm (object, se.type="Default", n.top=3, ...)
```

`summary()` prints the call, coefficients (with standard errors and t-statistics), r-squared and residual volatility for all assets, and `n.top` long and short positions for each factor mimicking portfolio. Argument `se.type`, one of "Default", "HC" or "HAC", allows for heteroskedasticity and autocorrelation consistent estimates and standard errors. The returned object of class `"summary.sfm"` contains a list of summaries of the fitted "lm" objects, factor loadings, residual volatilities (under the homo-skedasticity assumption), R-squared values and factor mimicking portfolio weights.

Factor model covariance and risk decomposition functions are explained in section 4 and the `plot` method is discussed separately in Section 5.

Here are some examples using the statistical factor models fitted earlier.

```
# all estimated coefficients from PCA example
coef(fit.pca)

##        (Intercept)    F.1      F.2
## CITCRP    0.001761 0.2727 -0.08549
## CONED     0.016733 0.0444  0.00119
## CONTIL   -0.008949 0.3769 -0.82358
## DATGEN   -0.010411 0.4172  0.15182
```

```
## DEC       0.006514 0.3049  0.12907
## DELTA     0.000199 0.2502  0.17906
## GENMIL    0.011167 0.1326  0.01712
## GERBER    0.011475 0.1672 -0.19851
## IBM       0.003690 0.1464  0.01240
## MOBIL     0.010565 0.1552 -0.06235
## PANAM    -0.011994 0.3107  0.36542
## PSNH     -0.007648 0.0841  0.01046
## TANDY     0.006845 0.4119  0.20710
## TEXACO    0.007307 0.1323 -0.07177
## WEYER    -0.002030 0.2649  0.13123
```

```
# compare returns data with fitted and residual values for CITCRP: fit.pca
CITCRP.ts <- merge(fit.pca$data[,1], fitted(fit.pca)[,1],
                   residuals(fit.pca)[,1])
colnames(CITCRP.ts) <- c("CITCRP.return","CITCRP.fitted","CITCRP.residual")
tail(CITCRP.ts)
```

```
##            CITCRP.return CITCRP.fitted CITCRP.residual
## 1987-07-01         0.041       0.05562        -0.01462
## 1987-08-01         0.033       0.00565         0.02735
## 1987-09-01        -0.086      -0.02429        -0.06171
## 1987-10-01        -0.282      -0.25650        -0.02550
## 1987-11-01        -0.136      -0.08875        -0.04725
## 1987-12-01         0.064       0.06053         0.00347
```

```
# summary for fit.pca with HAC standard erros
sum.pca <- summary(fit.pca, se.type="HAC", n.top=3)
names(sum.pca)
```

```
## [1] "call"      "se.type"   "sum.list"  "mimic.sum"
```

```
# print the summary for the 1st asset
sum.pca$sum.list[[1]]
```

```
##
## Call:
## lm(formula = CITCRP ~ f)
```

```
##
## Residuals:
##          CITCRP
## Min    -0.13961
## 1Q     -0.03487
## Median  0.00822
## 3Q      0.03419
## Max     0.14293
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)        0.00176    0.00461    0.38    0.703
## object$factorsF.1  0.27271    0.02206   12.36   <2e-16 ***
## object$factorsF.2 -0.08549    0.04724   -1.81    0.073 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0505 on 117 degrees of freedom
## Multiple R-squared:  0.618,Adjusted R-squared:  0.611
## F-statistic: 94.6 on 2 and 117 DF,  p-value: <2e-16
```

```
# print the summary for the factor mimicking portfolio weights
sum.pca$mimic.sum
```

```
## $F.1
##   Top.Long.Name Top.Long.Weight Top.Short.Name Top.Short.Weight
## 1        DATGEN           0.120          CONED           0.0128
## 2         TANDY           0.119           PSNH           0.0242
## 3        CONTIL           0.109         TEXACO           0.0381
##
## $F.2
##   Top.Long.Name Top.Long.Weight Top.Short.Name Top.Short.Weight
## 1        CONTIL           22.36          PANAM            -9.92
## 2        GERBER            5.39          TANDY            -5.62
## 3        CITCRP            2.32          DELTA            -4.86
```

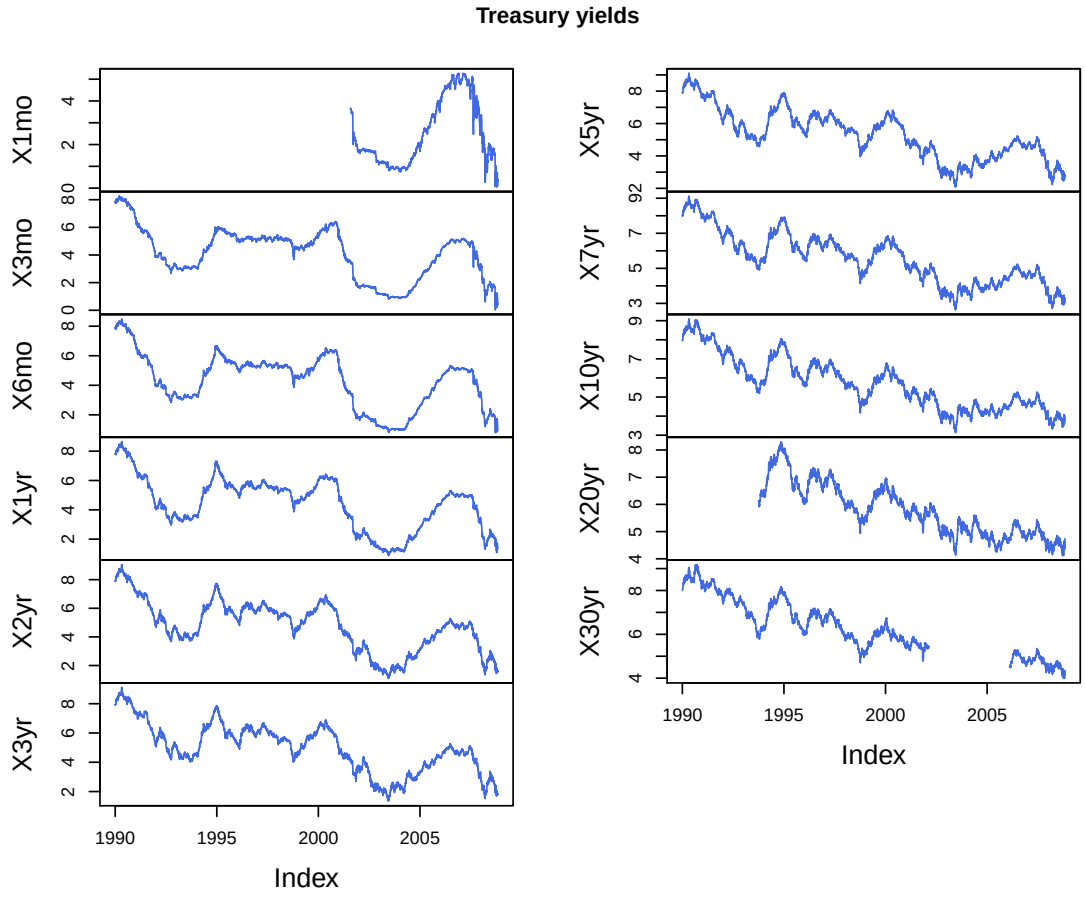

Figure 10: Treasury yields data for 11 different maturities

3 Treasury yield curve example

The following example uses PCA to model yield curve variations similar to Example 17.2 in Ruppert (2010). The Treasury yields data used was described in section 1 earlier. Figure 10 plots the time series of the raw data and Figure 11 shows the treasury yield curve through time.

```
plot.zoo(tr.yields, main="Treasury yields", col="royalblue")
```

```
dat <- na.omit(tr.yields) # remove NAs
time = c(1/12,.25,.5,1, 2, 3, 5, 7, 10, 20, 30)
plot(time, as.vector(dat[1,]), ylim=c(0,6), type="b", col="royalblue", lwd=2,
     pch=19, ylab="Yield", xlab="T")
```

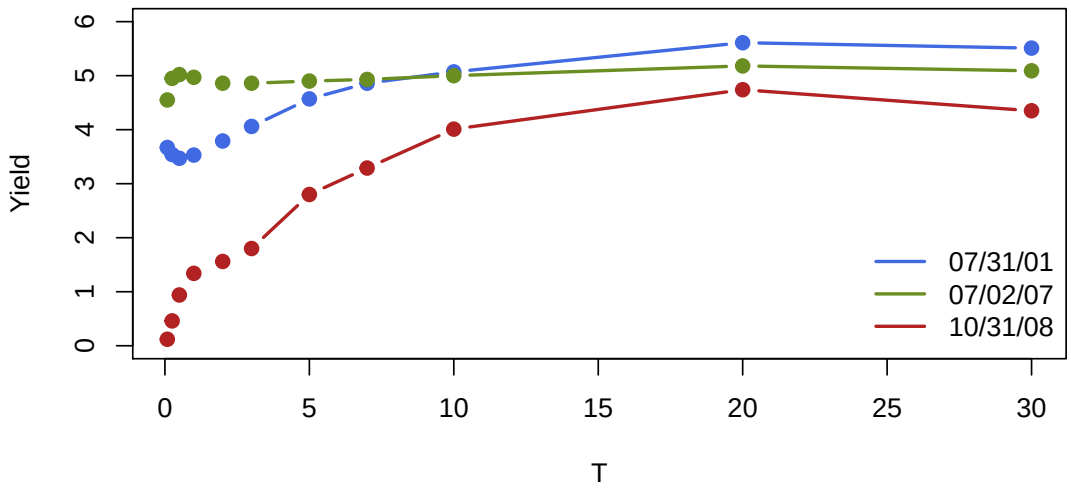

Figure 11: Treasury yield curve at 3 different dates

```
lines(time, as.vector(dat[486,]), type="b", lwd=2, col="olivedrab", pch=19)
lines(time, as.vector(dat[821,]), type="b", lwd=2, col="firebrick", pch=19)
legend("bottomright", c("07/31/01","07/02/07","10/31/08"),
       col=c("royalblue","olivedrab","firebrick"), lwd=2, bty="n")
```

Next, we fit a statistical factor model to the differenced data, with missing values removed. Since all 11 series have the same units and a comparable scale, PCA is performed on the sample correlation matrix.

```
diff.yield <- na.omit(diff(tr.yields))
dim(diff.yield)

## [1] 819  11

yield.pca <- fitSfm(diff.yield, k=3, corr=TRUE)
```

Figure 12 shows a screeplot of all the eigenvalues. Approximately 94 percent of the variation is explained by the first 3 principal components and 99 percent explained by the first five. So, the choice of $k = 3$ when fitting the model is not inappropriate.

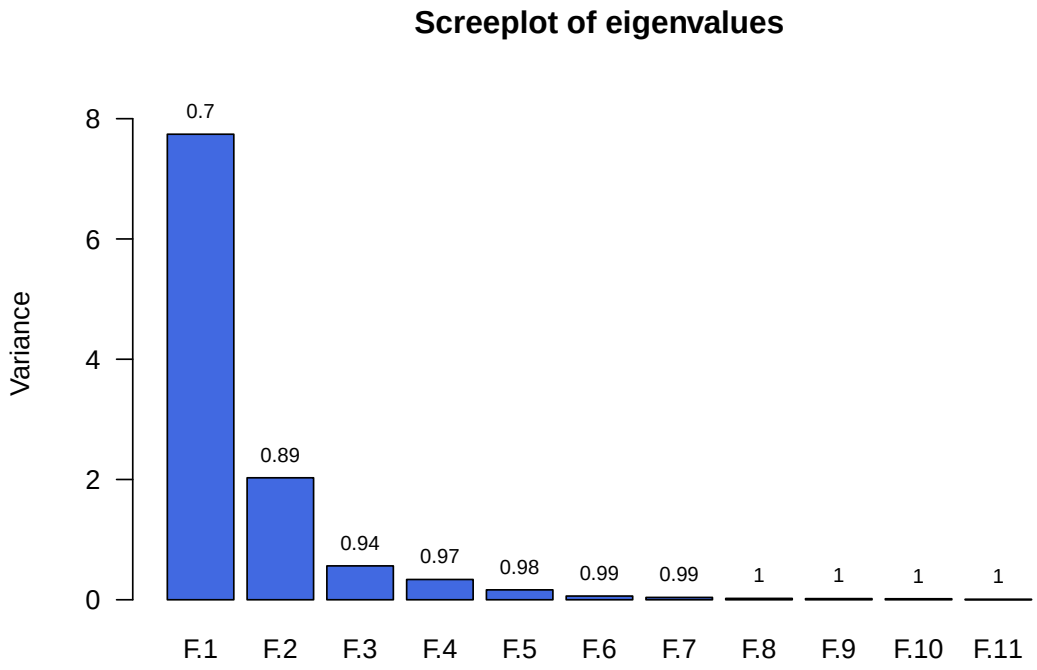

Figure 12: Screeplot of eigenvalues for the changes in Treasury yields

```
plot(yield.pca, which=1, f.sub=1:3, eig.max=1)
```

A summary of the distribution of the factor loadings and the top three assets with the largest and smallest weights in each factor mimicking portfolio are shown below.

```
beta <- yield.pca$loadings
summary(beta)
```

```
##       F.1              F.2               F.3
##  Min.   :0.176   Min.   :-0.588   Min.   :-0.375
##  1st Qu.:0.231   1st Qu.:-0.328   1st Qu.:-0.173
##  Median :0.272   Median : 0.053   Median :-0.025
##  Mean   :0.294   Mean   :-0.052   Mean   : 0.031
##  3rd Qu.:0.373   3rd Qu.: 0.204   3rd Qu.: 0.162
##  Max.   :0.397   Max.   : 0.251   Max.   : 0.870
```

```
summary(yield.pca)$mimic.sum
```

```
## $F.1
##   Top.Long.Name Top.Long.Weight Top.Short.Name Top.Short.Weight
## 1          X5yr           0.107           X1mo           0.0488
## 2          X7yr           0.106           X3mo           0.0595
## 3          X3yr           0.106           X6mo           0.0781
##
## $F.2
##   Top.Long.Name Top.Long.Weight Top.Short.Name Top.Short.Weight
## 1          X3mo           0.942          X30yr           -0.487
## 2          X1mo           0.853          X20yr           -0.477
## 3          X6mo           0.741          X10yr           -0.386
##
## $F.3
##   Top.Long.Name Top.Long.Weight Top.Short.Name Top.Short.Weight
## 1          X1mo           1.754           X6mo           -0.805
## 2         X30yr           0.970           X1yr           -0.803
## 3         X20yr           0.801           X2yr           -0.738
```

Figure 13 and 14 show the factor loadings as barplots and as line plots respectively.

```
plot(yield.pca, which=3, f.sub=1:3, a.sub=1:11)
```

```
plot(time, beta[,1], ylim=c(-.8,.8), type="b", col="royalblue", lwd=2, pch=19,
     ylab="Factor loading", xlab="T")
lines(time, beta[,2], type="b", lwd=2, col="olivedrab", pch=19)
lines(time, beta[,3], type="b", lwd=2, col="firebrick", pch=19)
legend("bottomright", c("F.1","F.2","F.3"),
       col=c("royalblue","olivedrab","firebrick"), lwd=2, bty="n")
```

All the weights in the first portfolio are positive and roughly the same and any change in the first factor affects all the variables by similar amounts, causing somewhat paralell shifts. So, this might be interpreted as a level factor. The factor loadings for the second principal component are decreasing and any change in this factor affects the slope of the yield curve. Finally, the factor loadings for the third principal component are decreasing and then increasing and any change in this factor affects the curvature of the yield curve. This is illustrated next in Figure 15.

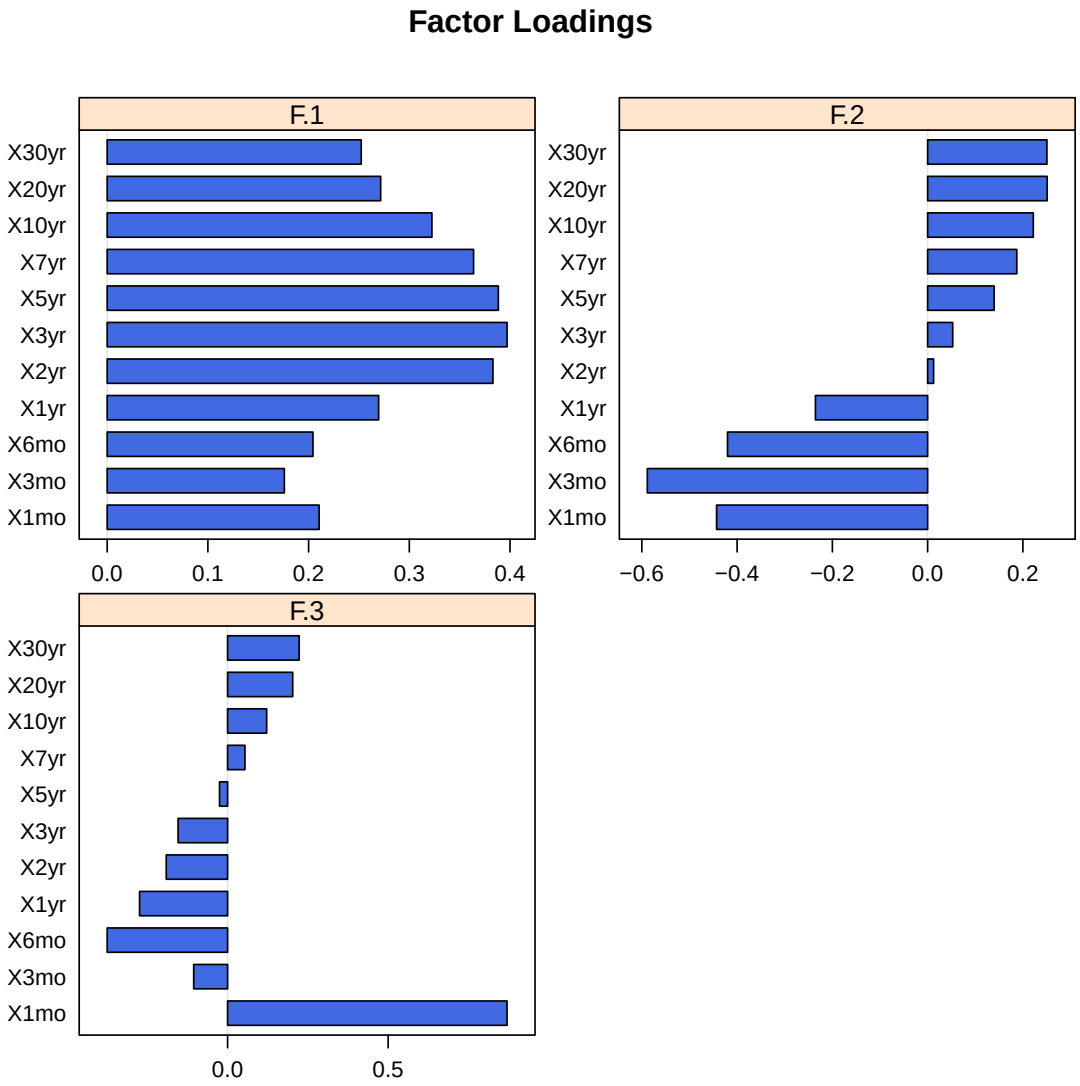

Figure 13: Factor loadings on the 1st three Principal components

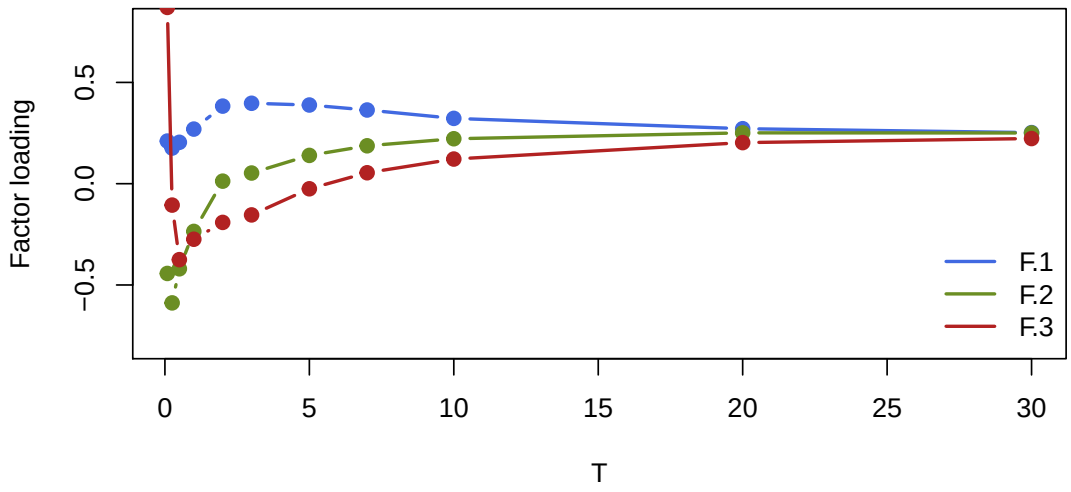

Figure 14: The loadings on the 1st three factors across maturities

```
mu <- colMeans(dat)
par(mfrow=c(3,1))
for (i in 1:3) {
  plot(time, mu, ylim=c(2,5.3), type="b", col="royalblue", lwd=2, pch=19,
       ylab="Yield", xlab="T")
  lines(time, mu+beta[,i], type="b", lwd=2, lty=2, col="olivedrab", pch=19)
  lines(time, mu-beta[,i], type="b", lwd=2, lty=2, col="firebrick", pch=19)
  legend("bottomright", bty="n",
         c("mean", paste("mean+F.",i,sep=""), paste("mean-F.",i,sep="")),
         col=c("royalblue","olivedrab","firebrick"), lwd=2, lty=c(1,2,2))
}
```

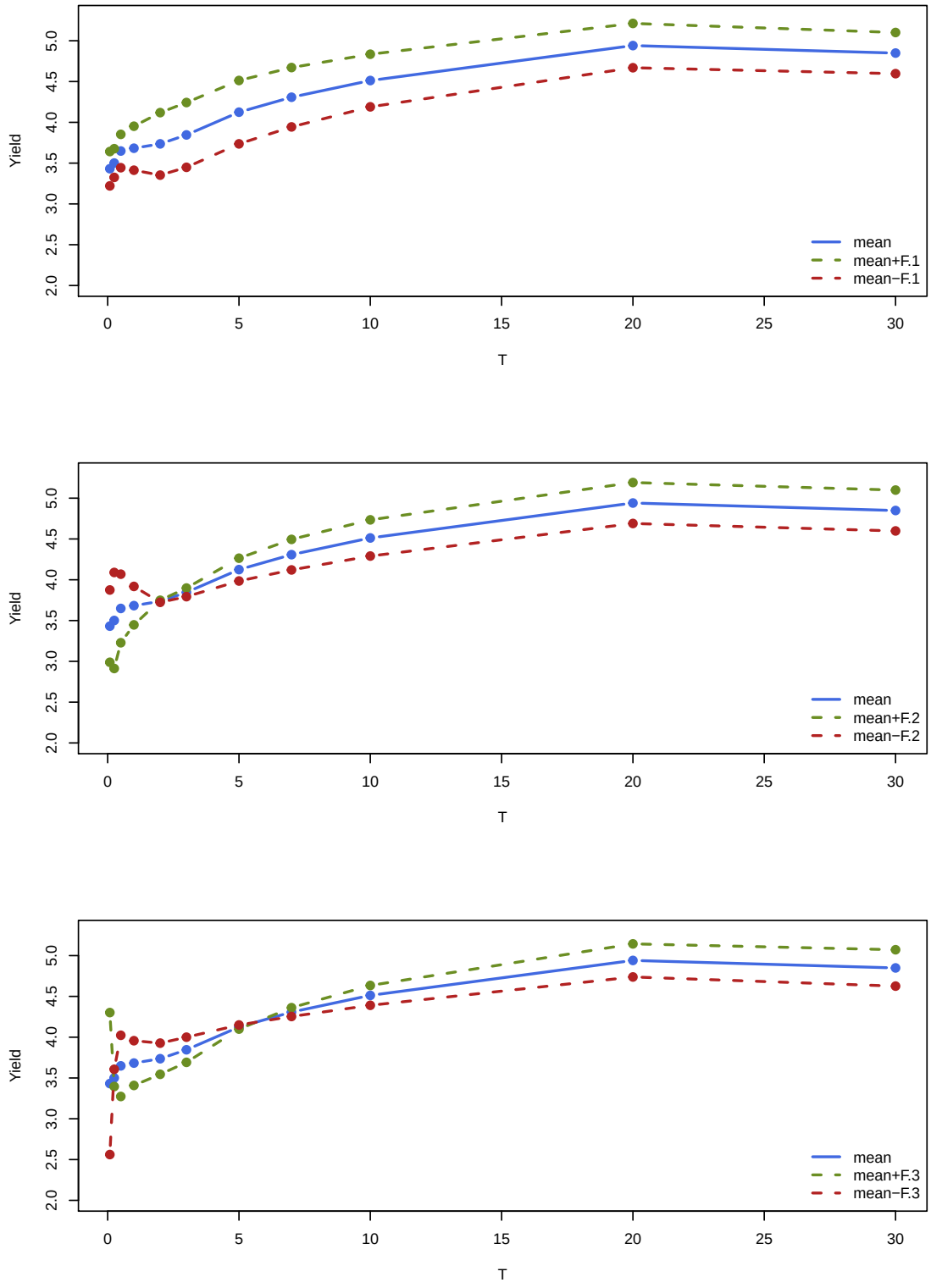

Figure 15: Effect of a unit change in the first 3 factors on the yield curve: level (shift), slope (tilt) and curvature (bend)

4 Factor Model Covariance & Risk Decomposition

4.1 Factor model covariance

Following Zivot and Jia-hui (2006), $R_{i,t}$, the return on asset i $(i = 1, ..., N)$ at time t $(t = 1, ..., T)$, is fitted with a factor model of the form,

$$R_{i,t} = \alpha_i + \boldsymbol{\beta}_i \, \mathbf{f_t} + \epsilon_{i,t} \tag{1}$$

where, α_i is the intercept, $\mathbf{f_t}$ is a $K \times 1$ vector of factor returns at time t, $\boldsymbol{\beta}_i$ is a $1 \times K$ vector of factor exposures for asset i and the error terms $\epsilon_{i,t}$ are serially uncorrelated across time and contemporaneously uncorrelated across assets so that $\epsilon_{i,t} \sim iid(0, \sigma_i^2)$. Thus, the variance of asset i's return is given by

$$var(R_{i,t}) = \boldsymbol{\beta}_i \, var(\mathbf{f_t}) \, \boldsymbol{\beta}_i' + \sigma_i^2 \tag{2}$$

And, the $N \times N$ covariance matrix of asset returns is

$$var(\mathbf{R}) = \boldsymbol{\Omega} = \mathbf{B} \, var(\mathbf{F}) \, \mathbf{B}' + \mathbf{D} \tag{3}$$

where, R is the $N \times T$ matrix of asset returns, B is the $N \times K$ matrix of factor betas, $\mathbf{F}$ is a $K \times T$ matrix of factor returns and D is a diagonal matrix with σ_i^2 along the diagonal.

`fmCov()` computes the factor model covariance from a fitted factor model. Options for handling missing observations include "pairwise.complete.obs" (default), "everything", "all.obs", "complete.obs" and "na.or.complete".

```
Omega <- fmCov(fit.pca)
# return correlation plot for all 15 assets
plot(fit.pca, which=8, a.sub=1:15, tl.cex=0.7)
```

4.2 Standard deviation decomposition

Given the factor model in equation 1, the standard deviation of the asset i's return can be decomposed as follows (based on Meucci (2007)):

$$R_{i,t} = \alpha_i + \boldsymbol{\beta}_i \, \mathbf{f_t} + \epsilon_{i,t} \tag{4}$$

$$= \boldsymbol{\beta}_i^* \, \mathbf{f_t^*} \tag{5}$$

where, $\boldsymbol{\beta}_i^* = (\boldsymbol{\beta}_i \, \sigma_i)$ and $\mathbf{f_t^*} = [\mathbf{f_t'} \, z_t]'$, with $z_t \sim iid(0, 1)$.

By Euler's theorem, the standard deviation of asset i's return is:

$$Sd.fm_i = \sum_{k=1}^{K+1} cSd_{i,k} = \sum_{k=1}^{K+1} \beta_{i,k}^* \, mSd_{i,k} \tag{6}$$

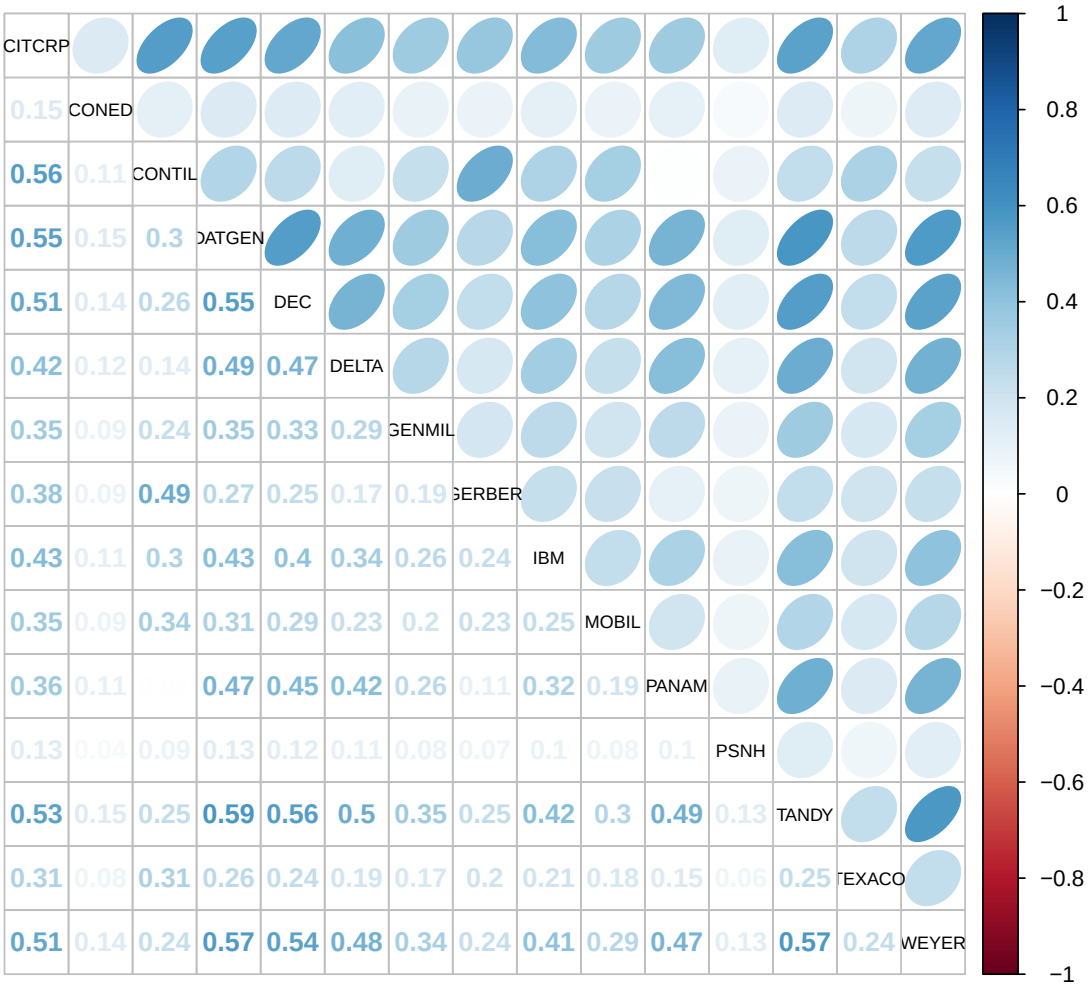

Figure 16: Factor model return correlation

where, summation is across the K factors and the residual, $\mathbf{cSd_i}$ and $\mathbf{mSd_i}$ are the component and marginal contributions to $Sd.fm_i$ respectively. Computing $Sd.fm_i$ and $\mathbf{mSd_i}$ is very straight forward. The formulas are given below and details are in Meucci (2007). The covariance term is approximated by the sample covariance.

$$Sd.fm_i = \sqrt{\boldsymbol{\beta}_i^* \, cov(\mathbf{F}^*) \, \boldsymbol{\beta}_i^{*'}} \tag{7}$$

$$\mathbf{mSd_i} = \frac{cov(\mathbf{F}^*) \, \boldsymbol{\beta}_i^*}{Sd.fm_i} \tag{8}$$

$$\mathbf{cSd_i} = \boldsymbol{\beta}_i^* \, \mathbf{mSd_i} \tag{9}$$

`fmSdDecomp` performs this decomposition for all assets in the given factor model fit object as shown below. The total standard deviation and component, marginal and percentage component contributions for each asset are returned.

```
decomp <- fmSdDecomp(fit.pca)
names(decomp)

## [1] "Sd.fm" "mSd"   "cSd"   "pcSd"

# get the factor model standard deviation for all assets
decomp$Sd.fm

## CITCRP  CONED CONTIL DATGEN    DEC  DELTA GENMIL GERBER    IBM  MOBIL
## 0.0812 0.0507 0.1508 0.1280 0.0995 0.0964 0.0655 0.0883 0.0594 0.0808
##  PANAM   PSNH  TANDY TEXACO  WEYER
## 0.1324 0.1104 0.1280 0.0803 0.0854

# get the component contributions to Sd; print first 6 assets
head(decomp$cSd)

##            F.1      F.2 residuals
## CITCRP 0.04807 1.80e-03    0.0314
## CONED  0.00204 5.61e-07    0.0486
## CONTIL 0.04947 8.99e-02    0.0114
## DATGEN 0.07140 3.60e-03    0.0530
## DEC    0.04904 3.34e-03    0.0472
## DELTA  0.03409 6.65e-03    0.0557

# plot the percentage component contributions to Sd for all 15 assets
plot(fit.pca, which=9, f.sub=1:2, a.sub=1:15)
```

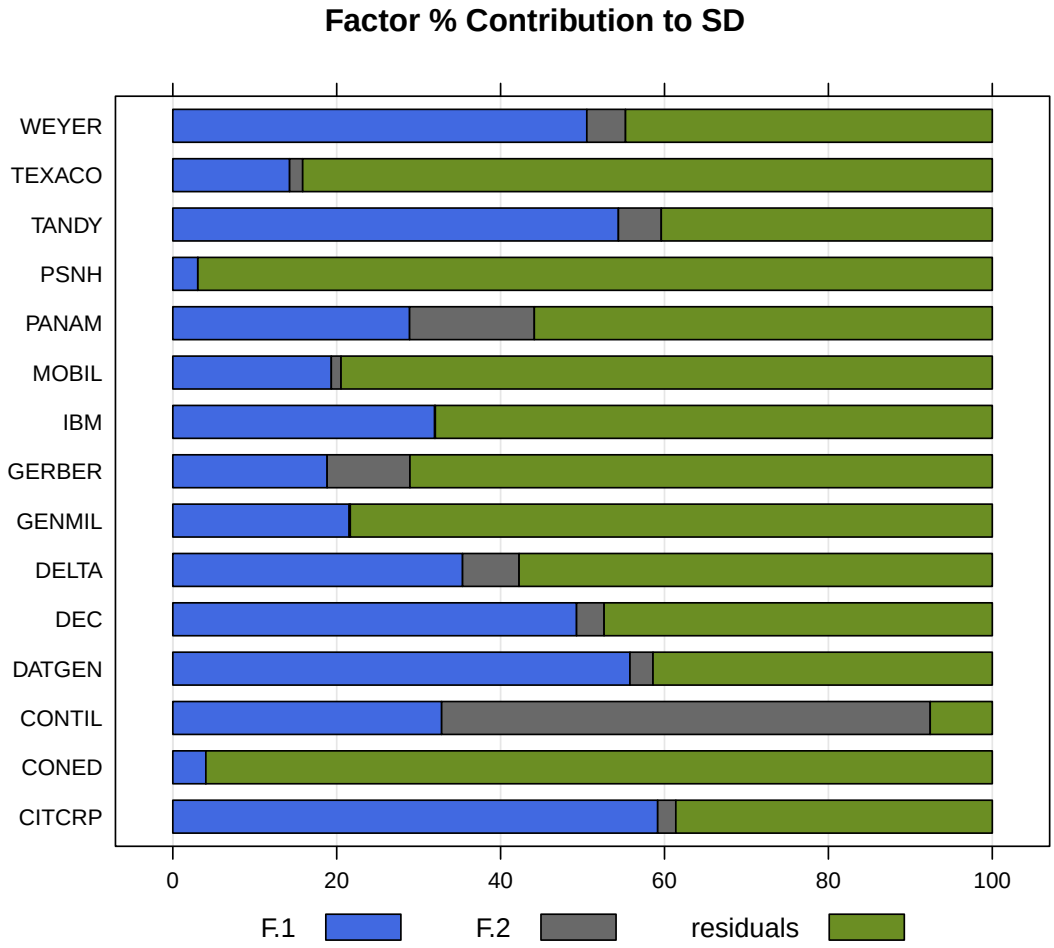

Figure 17: Percentage factor contribution to SD

4.3 Value-at-Risk decomposition

The VaR version of equation 6 is given below. By Euler's theorem, the value-at-risk of asset i's return is:

$$VaR.fm_i = \sum_{k=1}^{K+1} cVaR_{i,k} = \sum_{k=1}^{K+1} \beta^*_{i,k}\, mVaR_{i,k} \tag{10}$$

The marginal contribution to $VaR.fm$ is defined as the expectation of $F.star$, conditional on the loss being equal to $VaR.fm$. This is approximated as described in Epperlein and Smillie (2006) using a triangular smoothing kernel. $VaR.fm$ calculation is performed using the function `VaR` from the `PerformanceAnalytics` package. Refer to their help file for details and more options.

`fmVaRDecomp` performs this decomposition for all assets in the given factor model fit object as shown below. The total VaR and component, marginal and percentage component contributions for each asset are returned.

```
decomp1 <- fmVaRDecomp(fit.apca, method="historical")
names(decomp1)

## [1] "VaR.fm"     "n.exceed"   "idx.exceed" "mVaR"       "cVaR"
## [6] "pcVaR"

# factor model Value-at-Risk; print first 6 assets
head(decomp1$VaR.fm)

##   IATV   ADCT   ADEX    ABM   ACTM    AFL
## 0.2269 0.1192 0.1379 0.0755 0.1590 0.0729

# marginal factor contributions to VaR from 1st 4 factors; print first 6 assets
head(decomp1$mVaR[,1:4])

##          F.1    F.2    F.3      F.4
## IATV  0.1371  0.344  0.416  0.30219
## ADCT  0.1606  0.846  0.791  3.42492
## ADEX -0.0857  0.127  0.286 -0.13810
## ABM  -0.0501  0.161 -0.286  0.00649
## ACTM  0.5374  0.132 -0.180 -0.06618
## AFL   0.0511 -0.070 -0.879  0.04256

# plot the 1st 4 factors % component contributions to VaR for the 1st 6 assets
plot(fit.apca, which=11, f.sub=1:4, a.sub=1:6)
```

4.4 Expected Shortfall decomposition

The Expected Shortfall (ES) version of equation 6 is given below. By Euler's theorem, the expected shortfall of asset i's return is:

$$ES.fm_i = \sum_{k=1}^{K+1} cES_{i,k} = \sum_{k=1}^{K+1} \beta^*_{i,k}\, mES_{i,k} \tag{11}$$

The marginal contribution to $ES.fm$ is defined as the expectation of $F.star$, conditional on the loss being less than or equal to $VaR.fm$. This is estimated as a sample average of the observations in that data window. Once again, $VaR.fm$ calculation is performed using the function `VaR` from the `PerformanceAnalytics` package. Refer to their help file for details and more options.

`fmEsDecomp` performs this decomposition for all assets in the given factor model fit object as shown below. In this example, `method` to calculate VaR is "historical" instead of the default

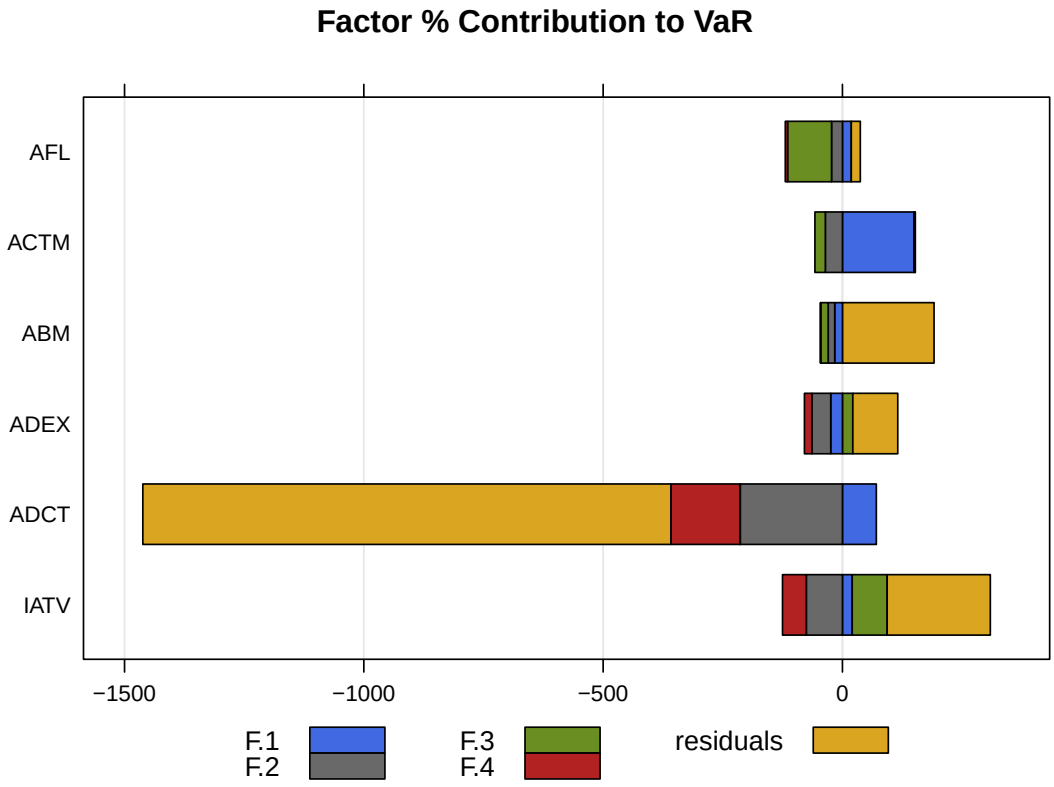

Figure 18: Percentage factor contribution to VaR

"modified". The total ES and component, marginal and percentage component contributions for each asset are returned.

```
decomp2 <- fmEsDecomp(fit.apca, method="historical")
names(decomp2)

## [1] "ES.fm"      "n.exceed"   "idx.exceed" "mES"        "cES"
## [6] "pcES"

# factor model Expected Shortfall; print first 6 assets
head(decomp2$ES.fm)

##   IATV   ADCT   ADEX    ABM   ACTM    AFL
## 0.2611 0.1998 0.1810 0.0947 0.2607 0.0885

# percentage component contributions to ES from 1st 4 factors; show 1st 6 assets
head(decomp2$pcES[,1:4])

##        F.1     F.2       F.3     F.4
## IATV  97.7   87.70 -41.62639 146.572
## ADCT  14.3    6.19  -0.00263   0.233
## ADEX 408.4 -393.83  14.83194  52.966
## ABM  -37.5  -15.71  48.36808  -3.896
## ACTM  47.8  -36.57 -24.46627   3.054
## AFL   58.4  -11.71  -5.16039  -7.984

# plot the 1st 4 factors % component contributions to ES for the 1st 6 assets
plot(fit.apca, which=10, f.sub=1:4, a.sub=1:6)
```

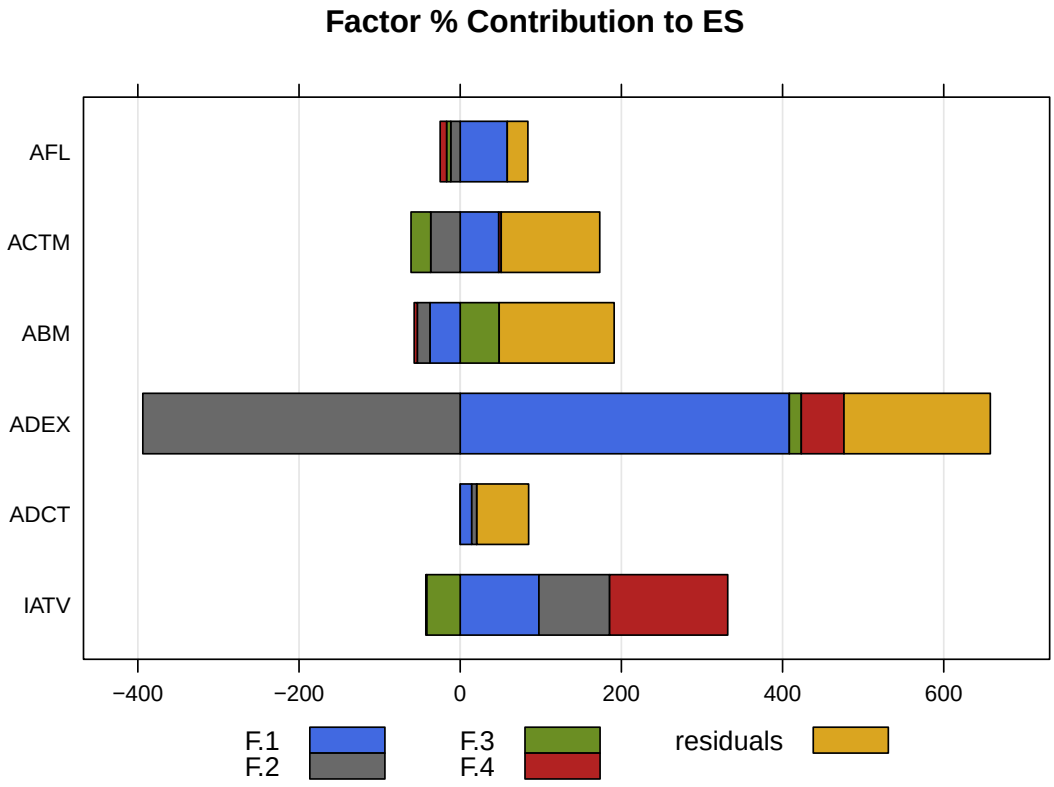

Figure 19: Percentage factor contribution to ES

5 Plot

Many plots for "sfm" objects have already been demonstrated. Let's take a look at all the available plot options and arguments.

```
## S3 method for class "sfm"
plot (x, which=NULL, f.sub=1:2, a.sub=1:6, n.top=3,
      plot.single=FALSE, asset.name,
      colorset=c("royalblue","firebrick","olivedrab","firebrick","goldenrod",
                 "mediumorchid","deepskyblue","chocolate","darkslategray"),
      legend.loc="topleft", las=1, lwd=2, maxlag=15,
      VaR.method="historical", eig.max=0.9, cum.var=TRUE, ...)
```

5.1 Group plots

This is the default option for plotting. Simply running `plot(fit)`, where `fit` is any "sfm" object, will bring up a menu (shown below) for group plots.

```
plot(fit.pca)


# Make a plot selection (or 0 to exit):
#
#  1: Screeplot of eigenvalues
#  2: Time series plot of estimated factors
#  3: Estimated factor loadings
#  4: Histogram of R-squared
#  5: Histogram of residual volatility
#  6: Scatterplot matrix of residuals, with histograms, density overlays,
#     correlations and significance stars
#  7: Factor model residual correlation
#  8: Factor model return correlation
#  9: Factor contribution to SD
# 10: Factor contribution to ES
# 11: Factor contribution to VaR
# 12: Factor mimicking portfolio weights - top long and short positions in each
#     factor
```

```
# 13: Asset correlations - top long and short positions in each factor
#
# Selection:
```

Remarks:

- Only a subset of assets and factors selected by `a.sub` and `f.sub` are plotted. The first 2 factors and first 6 assets are shown by default.
- `cum.var` applies to group plot 1, and specifies whether the cumulative fraction of the variance is printed above each bar in the screeplot of eigenvalues.
- `eig.max` also applies to group plot 1, and diplays the largest eigenvalues that cumulatively explain at least a given percent of the total variance.
- `n.top` applies to group plots 12 and 13, which involve summarizing the factor mimicking portfolios, and specifies the number of top positions to display.
- `VaR.method` applies to group plots 10 and 11, which are factor model risk ES and VaR decompositions respectively.

5.2 Menu and looping

If the plot type argument `which` is not specified, a menu prompts for user input. In case multiple plots are needed, the menu is repeated after each plot (enter 0 to exit). User can also input a numeric vector of plot options via `which`.

Note that Figures 1-9, 12, 13, 16-19 in the vignette above are all group plots.

5.3 Individual plots

Setting `plot.single=TRUE` enables individual asset plots. If there is more than one asset fit by the fitted object `x`, `asset.name` is also necessary. In case the `tsfm` object `x` contains only a single asset's fit, plot.tsfm can infer `asset.name` without user input.

Here's the individual plot menu.

```
plot(fit.pca, plot.single=TRUE, asset.name="DATGEN")


# Make a plot selection (or 0 to exit):
#
#  1: Actual and fitted asset returns
#  2: Actual vs fitted asset returns
#  3: Residuals vs fitted asset returns
#  4: Sqrt. of modified residuals vs fitted
#  5: Residuals with standard error bands
#  6: Time series of squared residuals
#  7: Time series of absolute residuals
#  8: SACF and PACF of residuals
#  9: SACF and PACF of squared residuals
# 10: SACF and PACF of absolute residuals
# 11: Non-parametric density of residuals with normal overlaid
# 12: Non-parametric density of residuals with skew-t overlaid
# 13: Histogram of residuals with non-parametric density and normal overlaid
# 14: QQ-plot of residuals
# 15: CUSUM test-Recursive residuals
# 16: CUSUM test-LS residuals
# 17: Recursive estimates (RE) test of LS regression coefficients
# 18: Rolling estimates over a 24-period observation window
#
# Selection:
```

Here are some examples which don't need interactive user input. These are individual plots for the DATGEN asset in the PCA fit illustrated earlier.

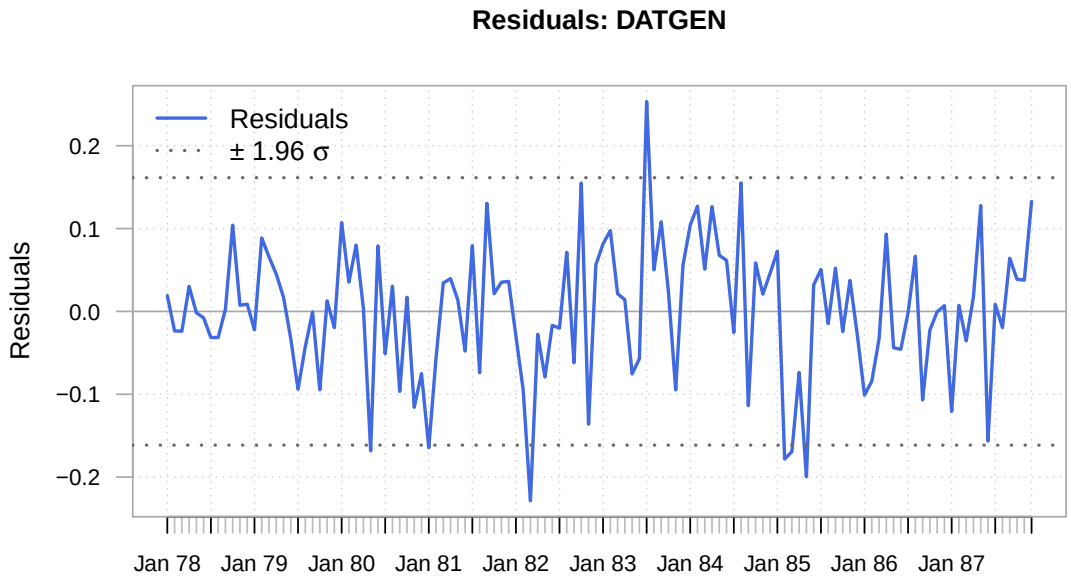

Figure 20: Time series plot of residuals with standard error bands: DATGEN

```
plot(fit.pca, plot.single=TRUE, asset.name="DATGEN", which=5)
```

```
plot(fit.pca, plot.single=TRUE, asset.name="DATGEN", which=10)
```

```
plot(fit.pca, plot.single=TRUE, asset.name="DATGEN", which=14)
grid()
```

```
plot(fit.pca, plot.single=TRUE, asset.name="DATGEN", which=11)
```

```
plot(fit.pca, plot.single=TRUE, asset.name="DATGEN", which=12)
```

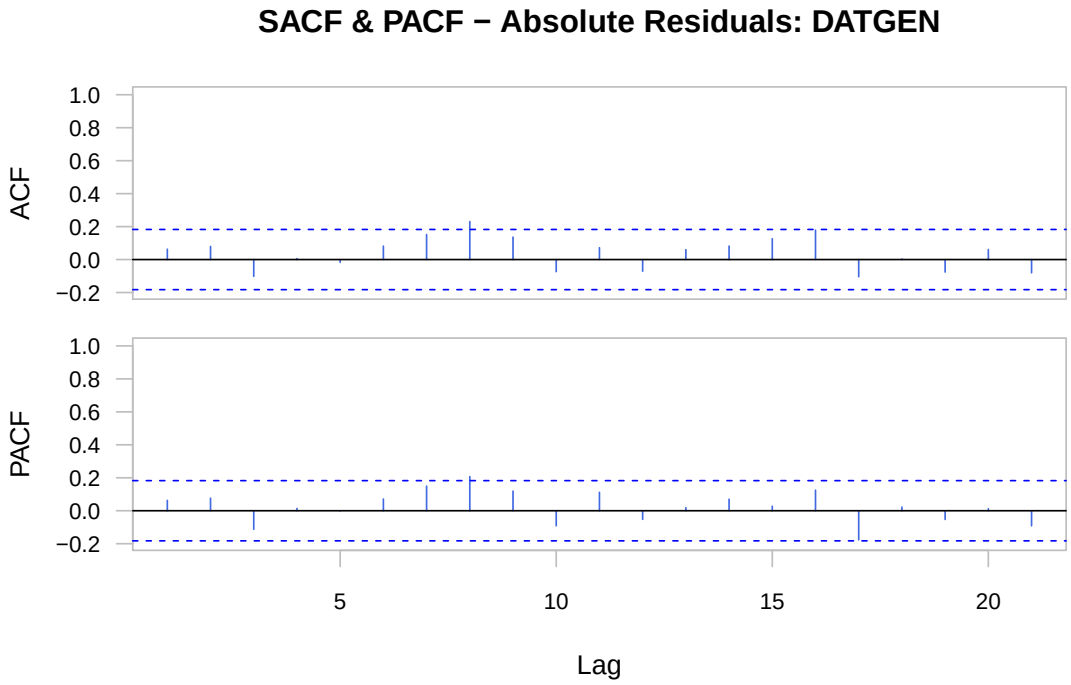

Figure 21: SACF and PACF of absolute residuals: DATGEN

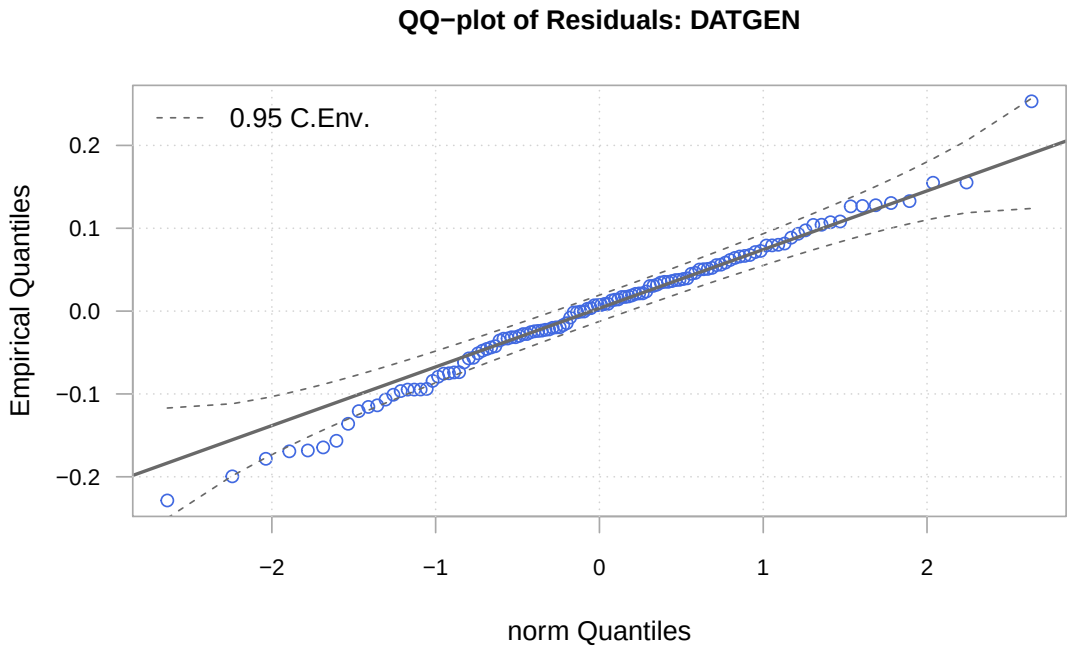

Figure 22: QQ-plot of residuals: DATGEN

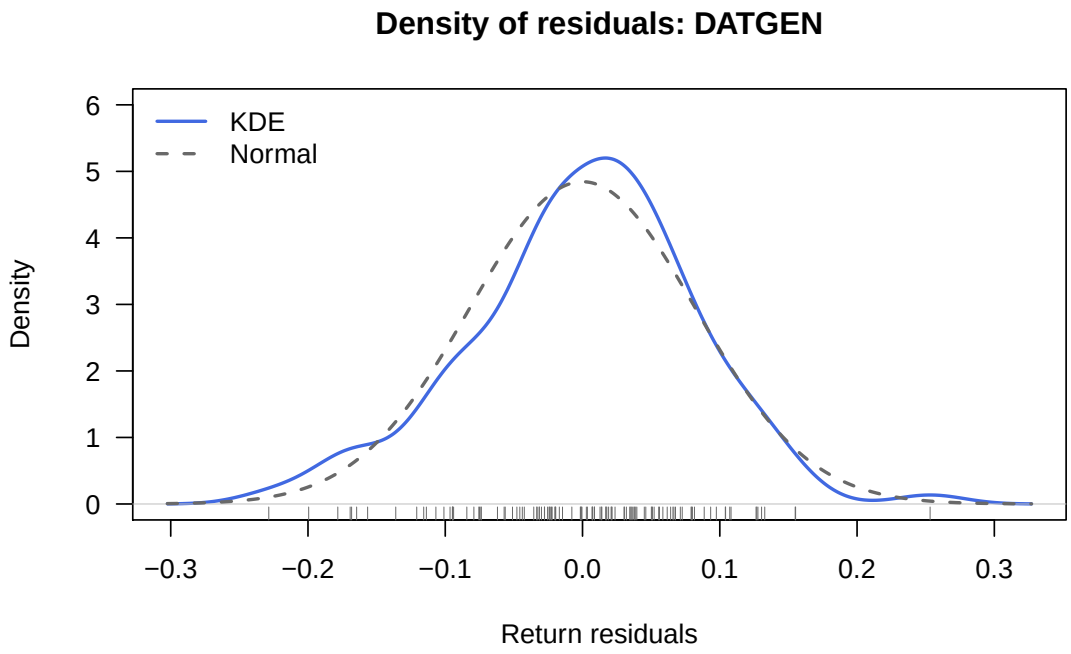

Figure 23: Non-parametric density of residuals with normal overlaid for DATGEN

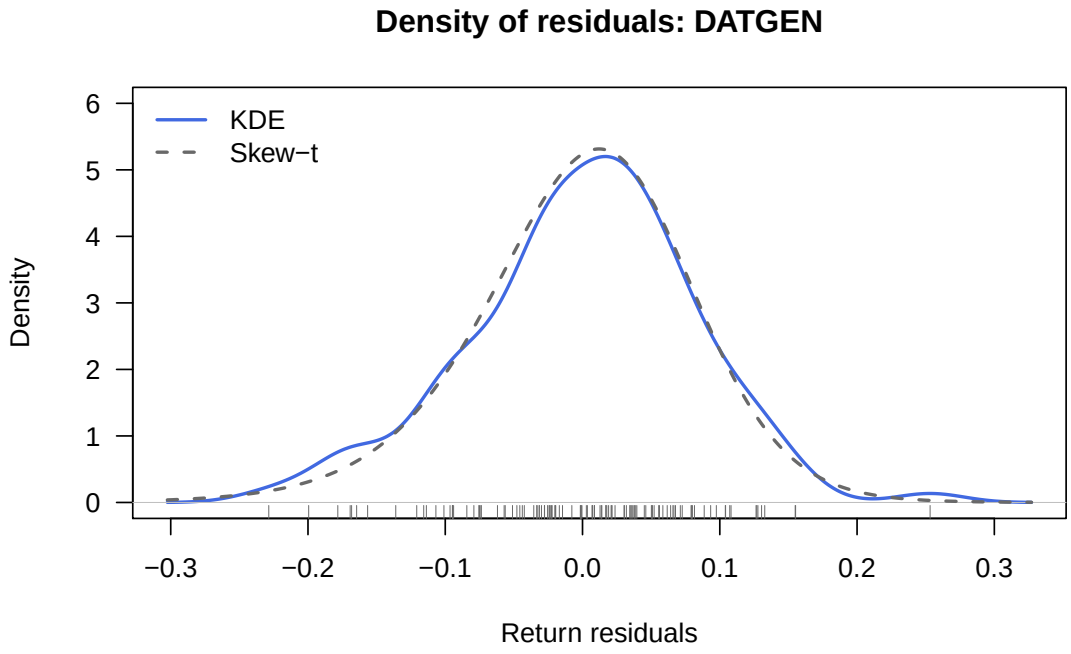

Figure 24: Non-parametric density of residuals with skew-t overlaid for DATGEN

References

J. Bai and S. Ng. Determining the number of factors in approximate factor models. *Econometrica*, 70(1):191–221, 2002.

E. R. Berndt. *The practice of econometrics: classic and contemporary*. Addison-Wesley Reading, MA, 1991.

G. Connor and R. A. Korajczyk. Risk and return in an equilibrium APT: Application of a new test methodology. *Journal of Financial Economics*, 21(2):255–289, 1988.

G. Connor and R. A. Korajczyk. A test for the number of factors in an approximate factor model. *The Journal of Finance*, 48(4):1263–1291, 1993.

E. Epperlein and A. Smillie. Portfolio risk analysis Cracking VAR with kernels. *RISK-LONDON-RISK MAGAZINE LIMITED-*, 19(8):70, 2006.

A. Meucci. Risk contributions from generic user-defined factors. *RISK-LONDON-RISK MAGAZINE LIMITED-*, 20(6):84, 2007.

D. Ruppert. *Statistics and data analysis for financial engineering*. Springer, 2010.

E. Zivot and W. Jia-hui. Modeling Financial Time Series with S-Plus Springer-Verlag. 2006.