Fitting a Time series Factor Model with fitTsfm in factorAnalytics

Sangeetha Srinivasan

November 21, 2014

Abstract

The purpose of this vignette is to demonstrate the use of `fitTsfm` 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)
```

1.2 Summary of related functions

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

- `fitTsfm(asset.names, factor.names, data, fit.method, variable.selection)`: Fits a time series (a.k.a. macroeconomic) factor model for one or more asset returns or excess returns using time series regression. Ordinary least squares-OLS, discounted least squares-DLS and robust regression fitting are possible. Variable selection methods include Step-wise, Subsets and Lars. An object of class `"tsfm"` containing the fitted objects, model coefficients, R-squared and residual volatility are returned.

- `coef(object)`: Extracts the coefficient matrix (intercept and factor betas) for all assets fit by the `tsfm` 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)`: 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 to calculate VaR. `invert` allows the VaR value to be expressed as a loss (vs. fund's return/profit).

- `fmEsDecomp(object)`: 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. `p` and `method` specify the confidence level and method to calculate VaR. `invert` allows the VaR value to be expressed as a loss (vs. fund's return/profit).

- `paFm(fit)`: Decompose total returns into returns attributed to factors and specific returns. An object of class `"pafm"` is returned, with methods for generic functions `plot`, `summary` and `print`.

- `plot(x)`: The `plot` method for class `"tsfm"` can be used for plotting factor model characteristics of an individual asset or a group of assets (default). The type of individual/group plot can be specified or chosen from a menu prompt (default if type not specified). Further the menu reappears (default) to enable multiple plots for the same asset(s) unless looping is disabled by setting `loop=FALSE`.

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

- `summary(object, se.type)`: The `summary` method for class `"tsfm"` returns an object of class `"summary.tsfm"` containing the summaries of the fitted `lm`, `lmRob` or `lars` objects and the chosen type (HC/HAC) of standard errors and t-statistics to display. Printing the factor model summary object outputs the call, coefficients (with standard errors and t-statistics), r-squared and residual volatility (under the homo-skedasticity assumption) for all assets.

1.3 Data

The following examples primarily use the managers dataset from the PerformanceAnalytics package. It's an xts data object with 132 observations on 10 variables; frequency is monthly.

```
data(managers)
colnames(managers)

##  [1] "HAM1"        "HAM2"        "HAM3"        "HAM4"        "HAM5"
##  [6] "HAM6"        "EDHEC LS EQ" "SP500 TR"    "US 10Y TR"   "US 3m TR"

range(index(managers))

## [1] "1996-01-31" "2006-12-31"
```

In the examples below, the monthly returns for the six hypothetical asset managers (HAM1 through HAM6) will be the explained asset returns. Columns 7 through 9, composed of the EDHEC Long-Short Equity hedge fund index, the S&P 500 total returns, and the total return series for the US Treasury 10-year bond will serve as explanatory factors. The last column (US 3-month T-bill) can be considered as the risk free rate. The series have unequal histories in this sample and fitTsfm removes asset-wise incomplete cases (asset's return data combined with respective factors' return data) before fitting a factor model.

```
asset.names <- colnames(managers[,1:6])
factor.names <- colnames(managers[,7:9])
mkt.name <- "SP500 TR"
rf.name <- "US 3m TR"
```

Typically, factor models are fit using excess returns. If the asset and factor returns are not already in excess return form, rf.name can be specified to convert returns into excess returns. Similarly, market returns can be specified to add market-timing factors to the factor model.

The CommonFactors dataset in the factorAnalytics package also provides a collection of common factors as both monthly (factors.M) and quarterly (factors.Q) time series.

```
data(CommonFactors)
names(factors.Q)

## [1] "SP500"         "GS10TR"        "USD.Index"     "Term.Spread"
## [5] "Credit.Spread" "dVIX"          "TED.Spread"    "OILPRICE"
## [9] "TB3MS"
```

```
range(index(factors.Q))

## [1] "1997-03-31" "2014-03-31"
```

2 Fit a time series factor model

In a time series or macroeconomic factor model, observable economic time series such as industrial production growth rate, interest rates, market returns and inflation are used as common factors that contribute to asset returns. For example, the famous single factor model by Sharpe and Sharpe (1970) uses the market excess return as the common factor (captures economy-wide or market risk) for all assets and the unexplained returns in the error term represents the non-market firm specific risk. Whereas, Chen et al. (1986) find that surprise inflation, the spread between long and short-term interest rates and between high and low grade bonds are significantly priced, while the market portfolio, aggregate consumption risk and oil price risk are not priced separately.

2.1 Excess returns & Market Timing factors

Let's take a look at the arguments for `fitTsfm`. The default regression fitting method is Ordinary Least Squares (OLS) and all factors are included in the model for all assets; no variable selection method is used. If `rf.name` is not specified by the user, perhaps because returns are already in excess return form, all returns are used as input by default.

```
args(fitTsfm)

## function (asset.names, factor.names, mkt.name = NULL, rf.name = NULL,
##     data = data, fit.method = c("OLS", "DLS", "Robust"), variable.selection = c("none",
##         "stepwise", "subsets", "lars"), mkt.timing = "both",
##     control = fitTsfm.control(...), ...)
## NULL
```

Here's an implementation of Sharpe's single index model for the 6 hypothetical assets described earlier. Since `rf.name` was included, excess returns were computed for all asset and factor returns before model fitting. The component `asset.fit` contains a list of fitted objects, one for each asset. Each object is of class `lm` if `fit.method="OLS" or "DLS"`, class `lmRob` if the `fit.method="Robust"`, or class `lars` if `variable.selection="lars"`. The different fit and variable selection methods are described in the next section.

```
fit.Sharpe <- fitTsfm(asset.names=asset.names, factor.names="SP500 TR",
                      rf.name="US 3m TR", data=managers)
names(fit.Sharpe)

##  [1] "asset.fit"          "alpha"              "beta"
##  [4] "r2"                 "resid.sd"           "call"
##  [7] "data"               "asset.names"        "factor.names"
## [10] "fit.method"         "variable.selection"

fit.Sharpe

##
## Call:
## fitTsfm(asset.names = asset.names, factor.names = "SP500 TR",
##     rf.name = "US 3m TR", data = managers)
##
## Model dimensions:
## Factors  Assets Periods
##       1       6     132
##
## Regression Alphas:
##                HAM1    HAM2    HAM3    HAM4    HAM5    HAM6
## (Intercept) 0.00577 0.00909 0.00622 0.00403 0.00173 0.00784
##
## Factor Betas:
##      SP500.TR
## HAM1    0.390
## HAM2    0.338
## HAM3    0.552
## HAM4    0.691
## HAM5    0.321
## HAM6    0.324
##
## R-squared values:
##   HAM1   HAM2   HAM3   HAM4   HAM5   HAM6
## 0.4339 0.1673 0.4341 0.3148 0.0829 0.2601
```

```
##
## Residual Volatilities:
##   HAM1   HAM2   HAM3   HAM4   HAM5   HAM6
## 0.0193 0.0334 0.0274 0.0443 0.0441 0.0206
```

In the following example, market timing factors are included in addition to the 3 other factors available in the `managers` dataset. Market timing accounts for the price movement of the general stock market relative to fixed income securities. "HM" follows Henriksson and Merton (1981) and $up.market = max(0, R_m - R_f)$, is added as a factor in the regression. Similarly, "TM" follows Treynor and Mazuy (1966), to account for market timing with respect to volatility, and $market.sqd = (R_m - R_f)^2$ is added as a factor. Option "both" (default) adds both of these factors.

```
# adding up-market timing factor ("HM") to the model
fit1 <- fitTsfm(asset.names=asset.names, factor.names=factor.names,
                mkt.name="SP500 TR", mkt.timing="HM", data=managers)

## Warning:  Excess returns were not computed.  Returns data were used as input
##             for all factors and assets.

fit1$beta

##      EDHEC.LS.EQ SP500.TR  US.10Y.TR up.market
## HAM1      0.2701  0.30767 -0.2223710  -0.03959
## HAM2      1.6034 -0.50588  0.0001061   0.57451
## HAM3      1.2709  0.08229  0.1456135   0.08839
## HAM4      1.1495  0.52212 -0.1212071  -0.45362
## HAM5      1.6236 -0.20308  0.2707245   0.04736
## HAM6      1.3108 -0.33719 -0.1840786   0.29259

fit1$r2

##   HAM1   HAM2   HAM3   HAM4   HAM5   HAM6
## 0.5018 0.5675 0.6661 0.4202 0.2306 0.5873

fit1$resid.sd

##    HAM1    HAM2    HAM3    HAM4    HAM5    HAM6
## 0.01895 0.02415 0.02155 0.04253 0.04121 0.01581
```

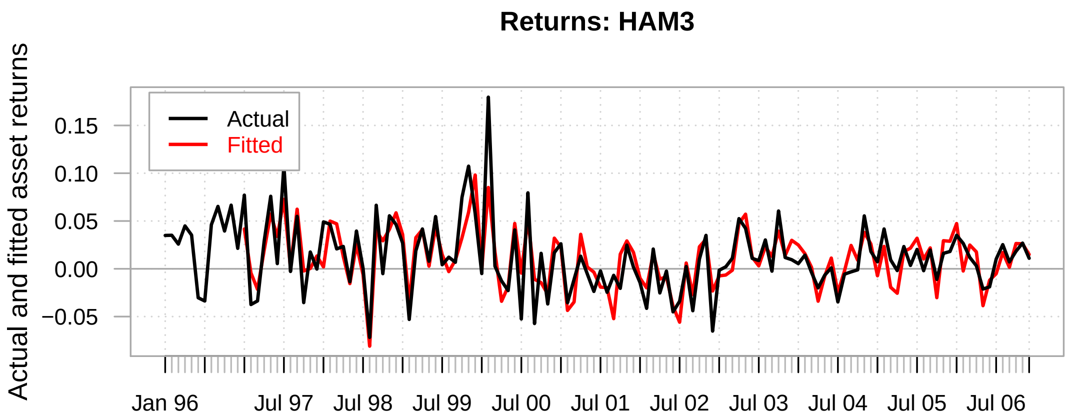
2.2 Fit methods & Variable Selection

Alternatives to "OLS" regression are `robust regression` (resistant to outliers and heteroskedasticity) and `exponentially discounted weights` (accounts for time variation in coefficients). These can be selected via the argument `fit.method` as shown below.

```
fit2 <- fitTsfm(asset.names=asset.names, factor.names=factor.names,
                mkt.name="SP500 TR", data=managers, fit.method="Robust")

## Warning:  Excess returns were not computed.  Returns data were used as input
##              for all factors and assets.

fit2$beta

##      EDHEC.LS.EQ SP500.TR US.10Y.TR up.market market.sqd
## HAM1      0.2310  0.38937  -0.16199   -0.3660      4.100
## HAM2      1.2070 -0.19524  -0.07257    0.1155      1.034
## HAM3      0.7712  0.06450   0.08170    0.3443     -1.321
## HAM4      1.4412 -0.67132   0.05012    1.6818     -9.757
## HAM5      1.3957 -0.73305   0.21700    1.1453     -6.706
## HAM6      1.1520 -0.06414  -0.15837    0.1366      2.774

fit2$r2

##   HAM1   HAM2   HAM3   HAM4   HAM5   HAM6
## 0.3272 0.2461 0.4745 0.3424 0.2569 0.4819

fit2$resid.sd

##    HAM1    HAM2    HAM3    HAM4    HAM5    HAM6
## 0.01705 0.02052 0.01572 0.03534 0.02806 0.01374
```

Notice the lower R-squared values and smaller residual volatilities with robust regression. Figures 1 and 2 give a graphical comparison of the fitted returns for asset "HAM3" and residual volatilities from the factor model fits. Figure 1 depicts the smaller influence that the volatility of Jan 2000 has on the robust regression. Plot options are described in detail in section 4.

```
par(mfrow=c(2,1))
plot(fit1, plot.single=TRUE, which.plot.single=1, asset.name="HAM3", loop=FALSE)
plot(fit2, plot.single=TRUE, which.plot.single=1, asset.name="HAM3", loop=FALSE)
```



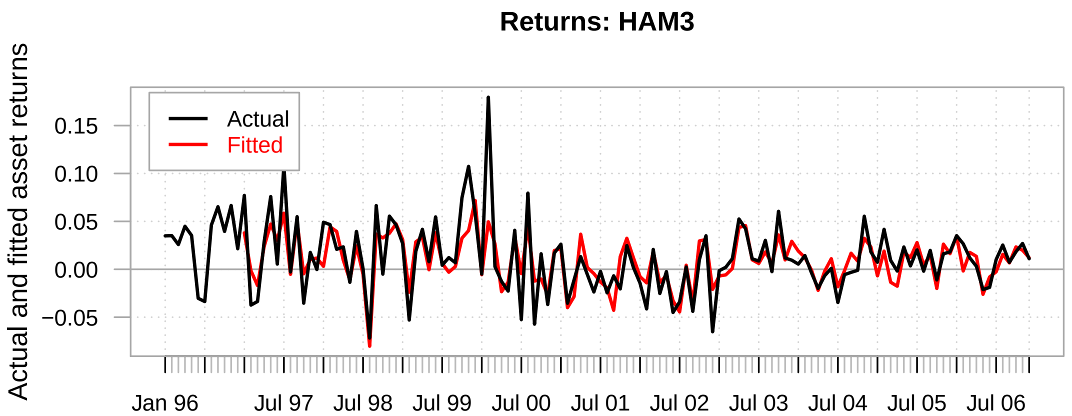


Figure 1: HAM3 Returns: fit1-OLS (top) vs fit2-Robust (bottom)

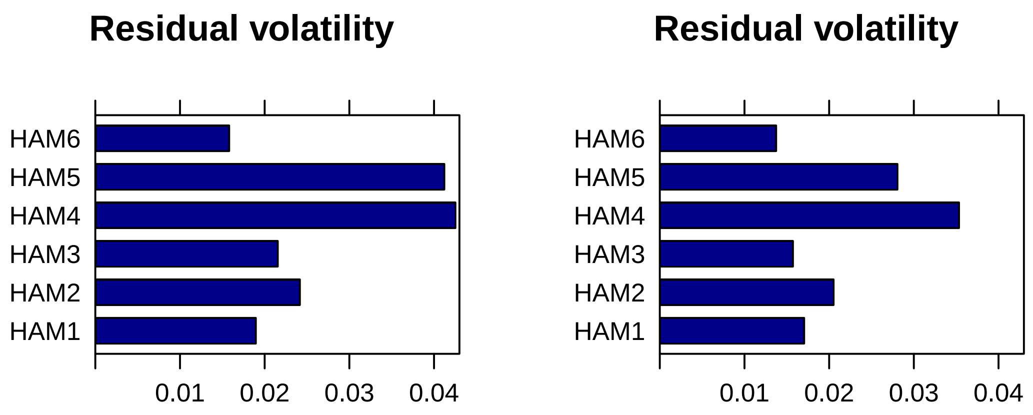


Figure 2: Residual vol: fit1-OLS (left) vs fit2-Robust (right)

```
par(mfrow=c(1,2))
plot(fit1, which.plot.group=5, loop=FALSE, xlim=c(0,0.043))
plot(fit2, which.plot.group=5, loop=FALSE, xlim=c(0,0.043))
```

By adding more factors in fit1 and fit2, though the R-squared values have improved (when compared to Sharpe's single index model), one might prefer to employ variable selection methods such as `stepwise`, `subsets` or `lars` to avoid over-fitting. The method can be selected via the `variable.selection` argument. The default `none`, uses all the factors and performs no variable selection. `stepwise` performs traditional forward or backward stepwise OLS regression, starting from an initial (given) set of factors and adds factors only if the regression fit, as measured by the Bayesian Information Criterion (BIC) or Akaike Information Criterion (AIC), improves. `subsets` enables subsets selection using `regsubsets`; chooses the best performing subset of any given size or within a range of subset sizes. `lars` corresponds to least angle regression using `lars` with variants "lasso", "lar", "forward.stagewise" or "stepwise".

Remarks:

- Variable selection methods `stepwise` and `subsets` can be combined with any of the fit methods, "OLS", "DLS" or "Robust".
- If variable selection method selected is `lars`, `fit.method` will be ignored.
- Refer to the section on `fitTsfm.control` for more details on the control arguments to the different variable selection methods.

The next example uses the `lars` variable selection method. The default type and criterion used are `lasso` and the `Cp` statistic. The `subsets` variable selection method is demonstrated next for comparison using the same set of factors. However, the best subset of size 4 for each asset is chosen. Figures 3 and 4 display the factor betas from the two fits.

```
fit.lars <- fitTsfm(asset.names=colnames(managers[,(1:6)]),
                    factor.names=colnames(managers[,(7:9)]), data=managers,
                    rf.name="US 3m TR", mkt.name="SP500 TR")
fit.lars$beta

##      EDHEC.LS.EQ SP500.TR US.10Y.TR up.market market.sqd
## HAM1      0.2374  0.12387 -0.210615   0.31807    -1.9203
## HAM2      1.6063 -0.46221 -0.007166   0.49058     0.4648
## HAM3      1.2636  0.08757  0.131896   0.07926     0.1747
## HAM4      1.0852 -0.18880 -0.041934   0.91369    -7.1087
## HAM5      1.6257 -0.40348  0.277982   0.42752    -2.1413
## HAM6      1.3182 -0.68511 -0.185234   0.94164    -3.5562

fit.lars$r2

##   HAM1   HAM2   HAM3   HAM4   HAM5   HAM6
## 0.5061 0.5629 0.6588 0.4407 0.2335 0.5973

fit.sub <- fitTsfm(asset.names=colnames(managers[,(1:6)]),
                   factor.names=colnames(managers[,(7:9)]), data=managers,
                   rf.name="US 3m TR", mkt.name="SP500 TR",
                   variable.selection="subsets", nvmin=4, nvmax=4)
fit.sub$beta

##      EDHEC.LS.EQ SP500.TR US.10Y.TR up.market market.sqd
## HAM1      0.2419       NA   -0.2074    0.5372     -2.916
## HAM2      1.6062 -0.46319        NA    0.4936      0.442
## HAM3      1.2612  0.07006    0.1332    0.1131         NA
## HAM4      1.0844 -0.19453        NA    0.9316     -7.242
## HAM5      1.6259 -0.20873    0.2710    0.0526         NA
## HAM6      1.3001 -0.34145   -0.1912    0.3025         NA

fit.sub$r2

##   HAM1   HAM2   HAM3   HAM4   HAM5   HAM6
## 0.5045 0.5629 0.6587 0.4405 0.2325 0.5880
```

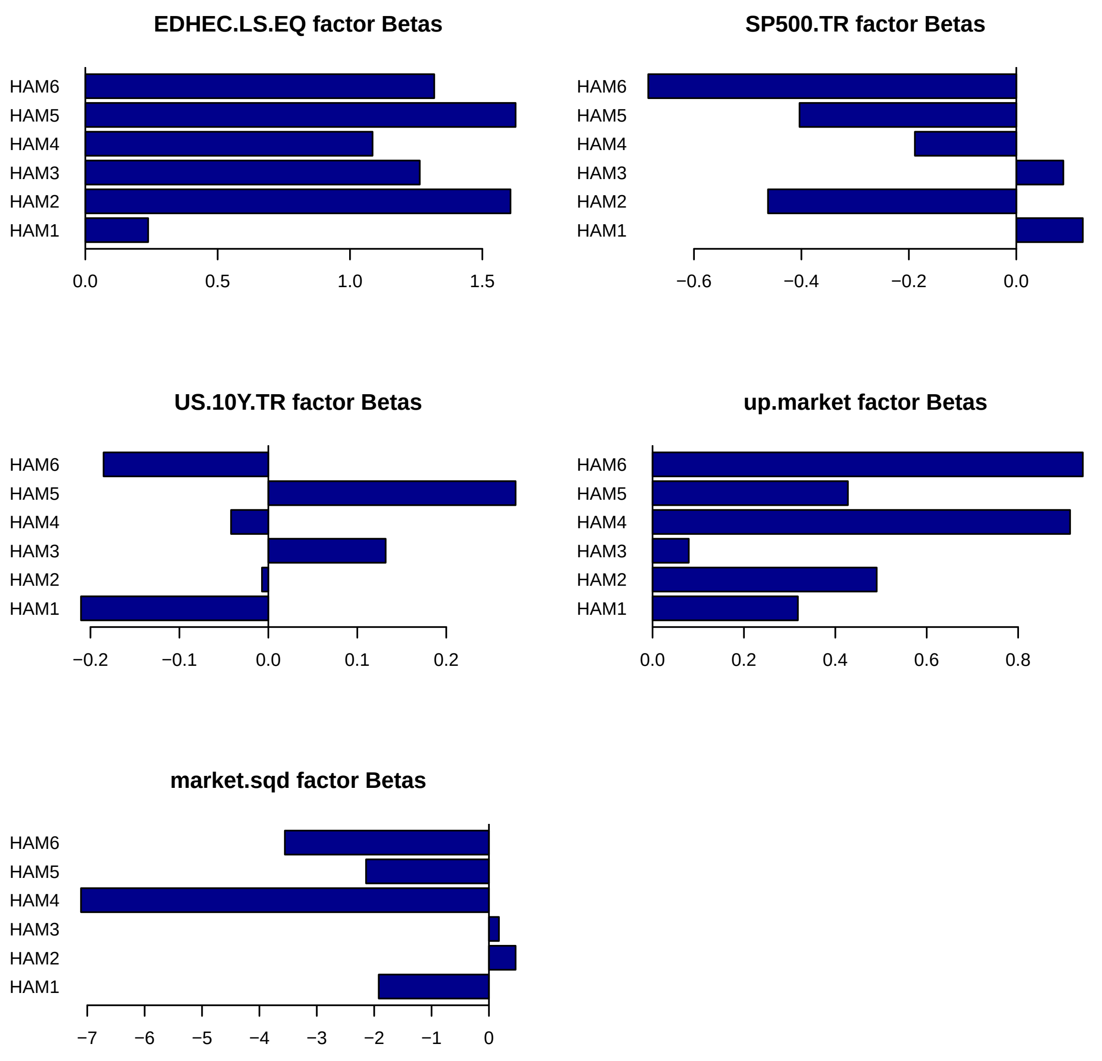


Figure 3: Factor betas: fit.lars

```
plot(fit.lars, which.plot.group=2, loop=FALSE)
```

```
plot(fit.sub, which.plot.group=2, loop=FALSE)
```

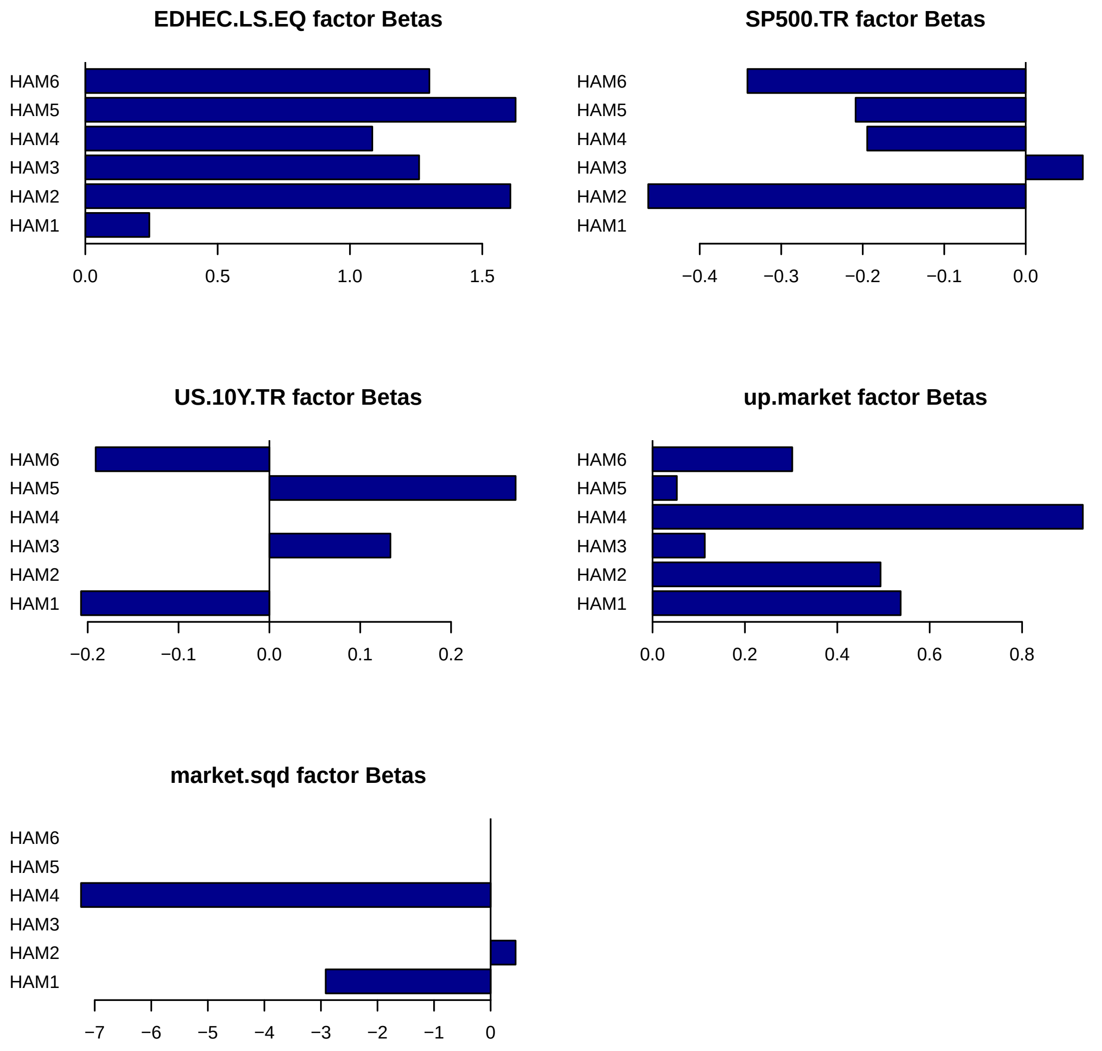


Figure 4: Factor betas: fit.sub

2.3 fitTsfm.control

Since `fitTsfm` calls many different regression fitting and variable selection methods, it made sense to collect all the optional controls for these functions and process them via `fitTsfm.control`. This function is meant to be used internally by `fitTsfm` when arguments are passed to it via the ellipsis. The use of control parameters was demonstrated with subset.size in the fit.sub example earlier.

```
args(fitTsfm.control)

## function (decay = 0.95, weights, model = TRUE, x = FALSE, y = FALSE,
##     qr = TRUE, nrep = NULL, scope, scale, direction, trace = FALSE,
##     steps = 1000, k = 2, nvmin = 1, nvmax = 8, force.in = NULL,
##     force.out = NULL, method, really.big = FALSE, type, normalize = TRUE,
##     eps = .Machine$double.eps, max.steps, lars.criterion = "Cp",
##     K = 10)
## NULL
```

Here's an ordered list of control parameters passed by `fitTsfm` matched with their respective functions for easy reference. See the corresponding help files for more details on each parameter.

- `lm`: "weights","model","x","y","qr"
- `lmRob`: "weights","model","x","y","nrep"
- `step`: "scope","scale","direction","trace","steps","k"
- `regsubsets`: "weights","nvmax","force.in","force.out","method","really.big"
- `lars`: "type","normalize","eps","max.steps","trace"
- `cv.lars`: "K","type","normalize","eps","max.steps","trace"

There are 4 other arguments passed to `fitTsfm.control` that determine the type of factor model fit chosen.

- `decay`: Determines the decay factor for `fit.method="DLS"`, which performs exponentially weighted least squares, with weights adding to unity.
- `nvmin`: The lower limit for the range of subset sizes from which the best model (BIC) is found when performing `subsets` selection. Note that the upper limit was already passed to `regsubsets` function. By specifying `nvmin=nvmax`, users can obtain the best model of a

particular size (meaningful to those who want a parsimonious model, or to compare with a different model of the same size, or perhaps to avoid over-fitting/ data dredging etc.).

- `lars.criterion`: An option (one of "Cp" or "cv") to assess model selection for the `"lars"` variable selection method. "Cp" is Mallow's Cp statistic and "cv" is K-fold cross-validated mean squared prediction error.

2.4 Summary, Predict, Coefficients, Fitted values and Residuals

```
methods(class="tsfm")

##  [1] coef.tsfm*        fitted.tsfm*      fmCov.tsfm*
##  [4] fmEsDecomp.tsfm*  fmSdDecomp.tsfm*  fmVaRDecomp.tsfm*
##  [7] plot.tsfm*        predict.tsfm*     print.tsfm*
## [10] residuals.tsfm*   summary.tsfm*
##
##    Non-visible functions are asterisked
```

Many useful generic accessor functions are available for `tsfm` fit objects. Here are some examples using the time series factor model fit by `fit.sub` earlier. `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 of the part of asset returns not explained by the factor model.

`summary()` prints standard errors and t-statistics for all estimated coefficients in addition to R-squared values and residual volatilities. Heteroskedasticity and auto-correlation consistent estimates and standard errors are available via the `se.type` argument. The returned "summary.tsfm" object also contains the summary objects returned by `lm`, `lm.Rob` or `lars`, which usually give more detailed statistics from the regression fit. `predict` uses the fitted factor model to estimate asset returns given a set of new or simulated factor return data.

```
coef(fit.sub)

##      (Intercept) EDHEC.LS.EQ SP500.TR US.10Y.TR up.market market.sqd
## HAM1   0.0010271      0.2419       NA   -0.2074    0.5372     -2.916
## HAM2  -0.0101414      1.6062 -0.46319        NA    0.4936      0.442
## HAM3  -0.0034598      1.2612  0.07006    0.1332    0.1131         NA
## HAM4  -0.0043613      1.0844 -0.19453        NA    0.9316     -7.242
## HAM5  -0.0053476      1.6259 -0.20873    0.2710    0.0526          NA
## HAM6  -0.0007332      1.3001 -0.34145   -0.1912    0.3025          NA

tail(fitted(fit.sub))

##                 HAM1      HAM2      HAM3      HAM4      HAM5      HAM6
## 2006-07-31 0.0000729 -0.019750 -0.010324 -0.007196 -0.014214 -0.011273
## 2006-08-31 0.0102240  0.004103  0.011736  0.017516  0.007962  0.005588
## 2006-09-30 0.0104476 -0.014113 -0.003768  0.005884 -0.013822 -0.007288
## 2006-10-31 0.0187903  0.018126  0.022177  0.029616  0.016255  0.019178
## 2006-11-30 0.0119278  0.017805  0.020838  0.024890  0.020790  0.018515
## 2006-12-31 0.0147528  0.009907  0.009883  0.017221  0.005692  0.018192

tail(residuals(fit.sub))

##                 HAM1      HAM2      HAM3      HAM4      HAM5      HAM6
## 2006-07-31 -0.018703  0.002420  0.016294 -0.009034 -0.006416 -0.015457
## 2006-08-31  0.001466 -0.019813  0.009154 -0.040226  0.004528  0.009302
## 2006-09-30 -0.008208 -0.013547  0.006408  0.009256  0.022462 -0.014972
## 2006-10-31  0.020100 -0.005236 -0.007687  0.018374  0.006535 -0.004088
## 2006-11-30 -0.004528 -0.001505  0.001762  0.008110 -0.021290  0.007185
## 2006-12-31 -0.007663 -0.020517 -0.003293 -0.001031  0.021598 -0.001102

# comparing data, fitted and residual values for HAM1
tail(merge(fit.sub$data[,1], fitted(fit.sub)[,1], residuals(fit.sub)[,1]))

##                HAM1    HAM1.1    HAM1.2
## 2006-07-31 -0.01863 0.0000729 -0.018703
## 2006-08-31  0.01169 0.0102240  0.001466
## 2006-09-30  0.00224 0.0104476 -0.008208
## 2006-10-31  0.03889 0.0187903  0.020100
## 2006-11-30  0.00740 0.0119278 -0.004528
## 2006-12-31  0.00709 0.0147528 -0.007663

# printed summary for the time series factor model
summary(fit.sub, se.type="HAC")

##
```

```
## Call:
## fitTsfm(asset.names = colnames(managers[, (1:6)]), factor.names = colnames(managers[,
##     (7:9)]), mkt.name = "SP500 TR", rf.name = "US 3m TR", data = managers,
##     variable.selection = "subsets", nvmin = 4, nvmax = 4)
##
## Factor Model Coefficients:
##
## Asset1: HAM1
## (HAC Standard Errors & T-stats)
##
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.00103    0.00236   0.435 6.64e-01
## EDHEC.LS.EQ  0.24192    0.17591   1.375 1.72e-01
## US.10Y.TR   -0.20737    0.07459  -2.780 6.35e-03
## up.market    0.53716    0.12952   4.147 6.47e-05
## market.sqd  -2.91591    0.86662  -3.365 1.04e-03
##
## R-squared: 0.504, Residual Volatility: 0.0189
##
## Asset2: HAM2
## (HAC Standard Errors & T-stats)
##
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.0101    0.00454  -2.233 2.75e-02
## EDHEC.LS.EQ   1.6062    0.24344   6.598 1.34e-09
## SP500.TR     -0.4632    0.29554  -1.567 1.20e-01
## up.market     0.4936    0.48289   1.022 3.09e-01
## market.sqd    0.4420    2.42722   0.182 8.56e-01
##
## R-squared: 0.563, Residual Volatility: 0.0242
##
## Asset3: HAM3
## (HAC Standard Errors & T-stats)
##
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.00346    0.00325  -1.066 0.288727
## EDHEC.LS.EQ  1.26124    0.33071   3.814 0.000222
## SP500.TR     0.07006    0.14278   0.491 0.624579
## US.10Y.TR    0.13323    0.08813   1.512 0.133362
## up.market    0.11309    0.14033   0.806 0.421975
##
## R-squared: 0.659, Residual Volatility: 0.0217
##
## Asset4: HAM4
## (HAC Standard Errors & T-stats)
##
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.00436    0.00959  -0.455  0.65006
## EDHEC.LS.EQ  1.08443    0.33490   3.238  0.00157
## SP500.TR    -0.19453    0.46066  -0.422  0.67361
## up.market    0.93155    0.80924   1.151  0.25206
## market.sqd  -7.24224    3.48194  -2.080  0.03975
##
## R-squared: 0.44, Residual Volatility: 0.0419
##
## Asset5: HAM5
## (HAC Standard Errors & T-stats)
##
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.00535    0.00845  -0.633 5.29e-01
## EDHEC.LS.EQ  1.62587    0.36569   4.446 3.11e-05
## SP500.TR    -0.20873    0.29891  -0.698 4.87e-01
## US.10Y.TR    0.27104    0.24358   1.113 2.70e-01
## up.market    0.05260    0.40361   0.130 8.97e-01
##
## R-squared: 0.233, Residual Volatility: 0.0412
##
## Asset6: HAM6
## (HAC Standard Errors & T-stats)
##
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.000733    0.00386   -0.19 8.50e-01
## EDHEC.LS.EQ  1.300116    0.20395    6.37 3.06e-08
## SP500.TR    -0.341448    0.08714   -3.92 2.34e-04
## US.10Y.TR   -0.191247    0.08990   -2.13 3.76e-02
## up.market    0.302487    0.20456    1.48 1.45e-01
##
## R-squared: 0.588, Residual Volatility: 0.0158
```

3 Factor Model Covariance & Risk Decomposition

3.1 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 and uses "pairwise.complete.obs" (default) to handle NAs. Other options for handling missing observations include "everything", "all.obs", "complete.obs" and "na.or.complete".

```
fmCov(fit.sub)

##          HAM1     HAM2     HAM3     HAM4     HAM5     HAM6
## HAM1 0.000693 0.000334 0.000469 0.000622 0.000336 0.000295
## HAM2 0.000334 0.001300 0.000725 0.000761 0.000698 0.000571
## HAM3 0.000469 0.000725 0.001344 0.001010 0.000761 0.000584
## HAM4 0.000622 0.000761 0.001010 0.003064 0.000837 0.000653
## HAM5 0.000336 0.000698 0.000761 0.000837 0.002446 0.000555
## HAM6 0.000295 0.000571 0.000584 0.000653 0.000555 0.000728

# return correlation plot; Angular Order of the Eigenvectors
plot(fit.sub, which.plot.group=7, loop=FALSE, order="AOE", method="ellipse",
     tl.pos = "d")
```

3.2 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)):

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

$$= \boldsymbol{\beta}_i^* \, \mathbf{f}_\mathbf{t}^* \tag{5}$$

where, $\boldsymbol{\beta}_i^* = (\boldsymbol{\beta}_i \; \sigma_i)$ and $\mathbf{f}_\mathbf{t}^* = [\mathbf{f}_\mathbf{t}' \; z_t]'$, with $z_t \sim iid(0,1)$.

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

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

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.

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

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

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

`fmSdDecomp` performs this decomposition for all assets in the given factor model fit object as shown below.

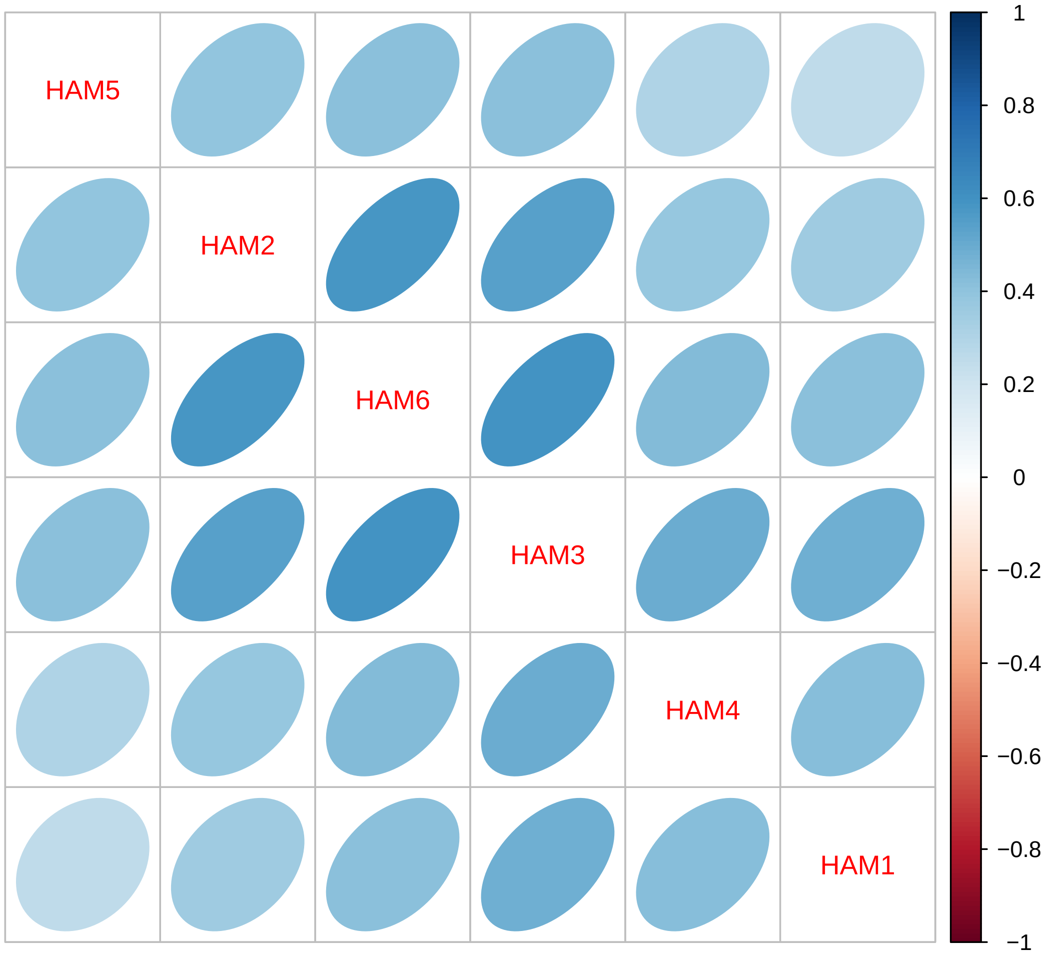


Figure 5: Factor model return correlation (pairwise complete obs)

```
decomp <- fmSdDecomp(fit.sub)
# get the factor model standard deviation for all assets
decomp$Sd.fm

##   HAM1   HAM2   HAM3   HAM4   HAM5   HAM6
## 0.0263 0.0360 0.0367 0.0554 0.0495 0.0270

# get the component contributions to Sd
decomp$cSd

##      EDHEC.LS.EQ SP500.TR US.10Y.TR up.market market.sqd residuals
## HAM1      0.0029  0.00000  0.001228  0.007201   1.45e-03   0.01355
## HAM2      0.0227 -0.00834  0.000000  0.005458   6.11e-05   0.01619
## HAM3      0.0204  0.00200 -0.000180  0.001631   0.00e+00   0.01279
## HAM4      0.0136 -0.00465  0.000000  0.009414   5.32e-03   0.03166
## HAM5      0.0175 -0.00289  0.000133  0.000354   0.00e+00   0.03434
## HAM6      0.0203 -0.00669  0.000925  0.003193   0.00e+00   0.00922

# get the marginal factor contributions to Sd
decomp$mSd

##      EDHEC.LS.EQ SP500.TR US.10Y.TR up.market market.sqd residuals
## HAM1      0.0120   0.0280 -0.005922   0.01341  -0.000496     0.717
## HAM2      0.0141   0.0180 -0.001527   0.01106   0.000138     0.670
## HAM3      0.0162   0.0285 -0.001350   0.01442  -0.000291     0.591
## HAM4      0.0126   0.0239 -0.003061   0.01011  -0.000734     0.756
## HAM5      0.0108   0.0138  0.000489   0.00672  -0.000229     0.833
## HAM6      0.0156   0.0196 -0.004834   0.01055  -0.000169     0.585

# get the percentage component contributions to Sd
decomp$pcSd

##      EDHEC.LS.EQ SP500.TR US.10Y.TR up.market market.sqd residuals
## HAM1        11.0     0.00     4.665    27.352       5.49      51.5
## HAM2        62.9   -23.14     0.000    15.139       0.17      44.9
## HAM3        55.7     5.45    -0.491     4.448       0.00      34.9
## HAM4        24.6    -8.40     0.000    17.006       9.61      57.2
## HAM5        35.4    -5.84     0.268     0.715       0.00      69.4
## HAM6        75.4   -24.81     3.427    11.834       0.00      34.2
```

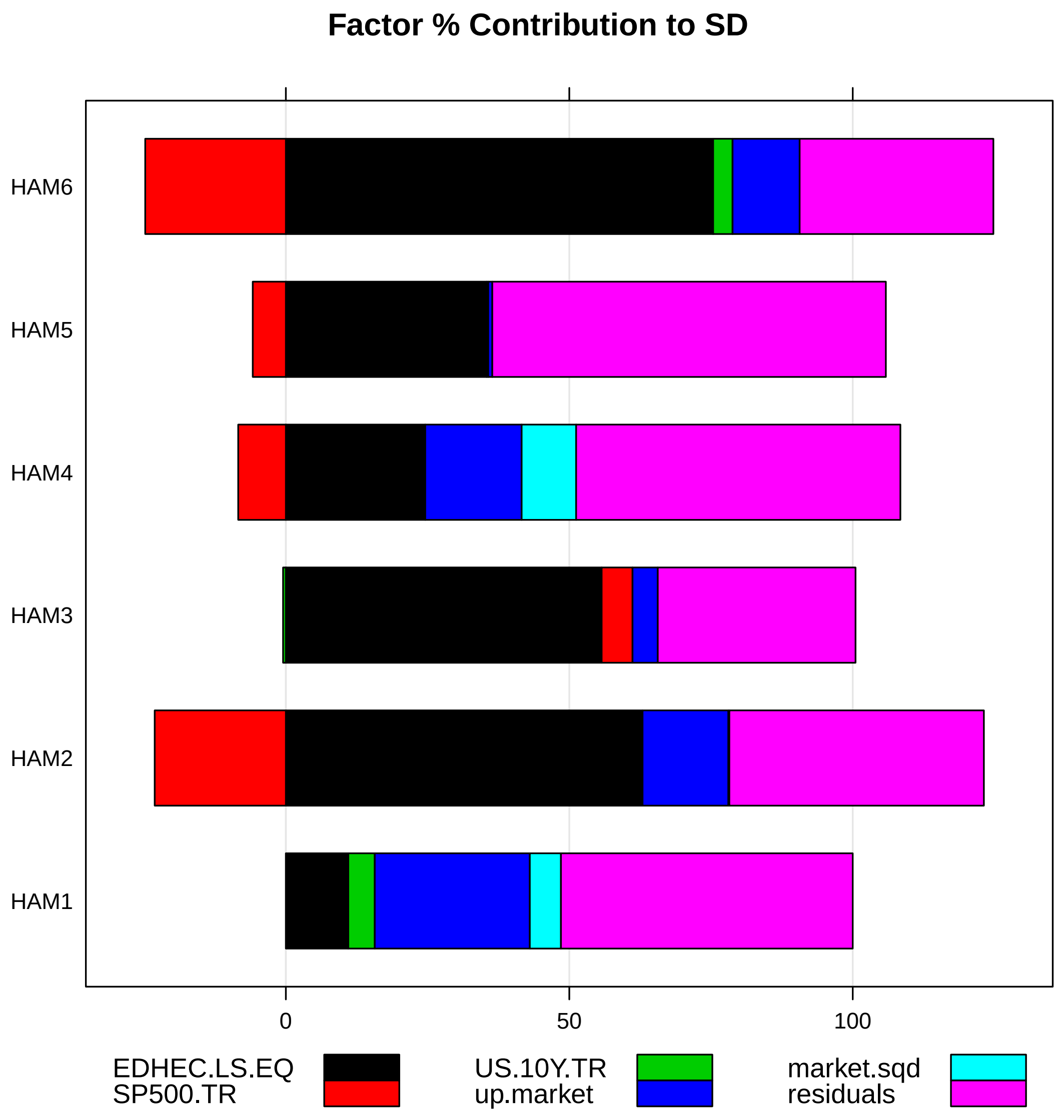


Figure 6: Percentage factor contribution to SD

```r
# plot the percentage component contributions to Sd
plot(fit.sub, which.plot.group=8, loop=FALSE)
```

3.3 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.

```
decomp2 <- fmVaRDecomp(fit.sub)
# get the factor model value-at-risk for all assets
decomp2$VaR.fm

##   HAM1   HAM2   HAM3   HAM4   HAM5   HAM6
## 0.0372 0.0310 0.0400 0.0848 0.0708 0.0317

# get the component contributions to VaR
decomp2$cVaR

##      EDHEC.LS.EQ SP500.TR US.10Y.TR up.market market.sqd residuals
## HAM1     0.00509  0.00000  0.001070   0.02536  -0.008106   0.01379
## HAM2     0.01981 -0.00635  0.000000   0.01226   0.000712   0.00461
## HAM3     0.02899  0.00288 -0.000356   0.00517   0.000000   0.00330
## HAM4     0.02702 -0.00706  0.000000   0.04091  -0.019293   0.04320
## HAM5     0.02830 -0.00319  0.002112   0.00119   0.000000   0.04243
## HAM6     0.02425 -0.00789  0.000544   0.00833   0.000000   0.00647

# get the marginal factor contributions to VaR
decomp2$mVaR

##      EDHEC.LS.EQ SP500.TR US.10Y.TR up.market market.sqd residuals
## HAM1      0.0211   0.0416 -0.005160    0.0472    0.00278     0.730
## HAM2      0.0123   0.0137 -0.003263    0.0248    0.00161     0.191
## HAM3      0.0230   0.0411 -0.002674    0.0457    0.00228     0.152
## HAM4      0.0249   0.0363  0.000647    0.0439    0.00266     1.032
## HAM5       0.0174    0.0153  0.007791     0.0226     0.00111      1.030
## HAM6       0.0187    0.0231 -0.002842     0.0275     0.00107      0.411

# get the percentage component contributions to VaR
decomp2$pcVaR

##      EDHEC.LS.EQ SP500.TR US.10Y.TR up.market market.sqd residuals
## HAM1         13.7     0.00     2.875     68.16     -21.78     37.07
## HAM2         63.8   -20.45     0.000     39.50       2.29     14.84
## HAM3         72.5     7.20    -0.891     12.94       0.00      8.25
## HAM4         31.9    -8.32     0.000     48.25     -22.76     50.95
## HAM5         40.0    -4.51     2.981      1.68       0.00     59.89
## HAM6         76.5   -24.88     1.714     26.27       0.00     20.42

# plot the percentage component contributions to VaR
plot(fit.sub, which.plot.group=10, loop=FALSE)
```

3.4 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 "modified".

```
decomp2 <- fmEsDecomp(fit.sub, method="historical")
# get the factor model expected shortfall for all assets
decomp2$ES.fm

##   HAM1   HAM2   HAM3   HAM4   HAM5   HAM6
## 0.0538 0.0370 0.0586 0.1153 0.1067 0.0411
```

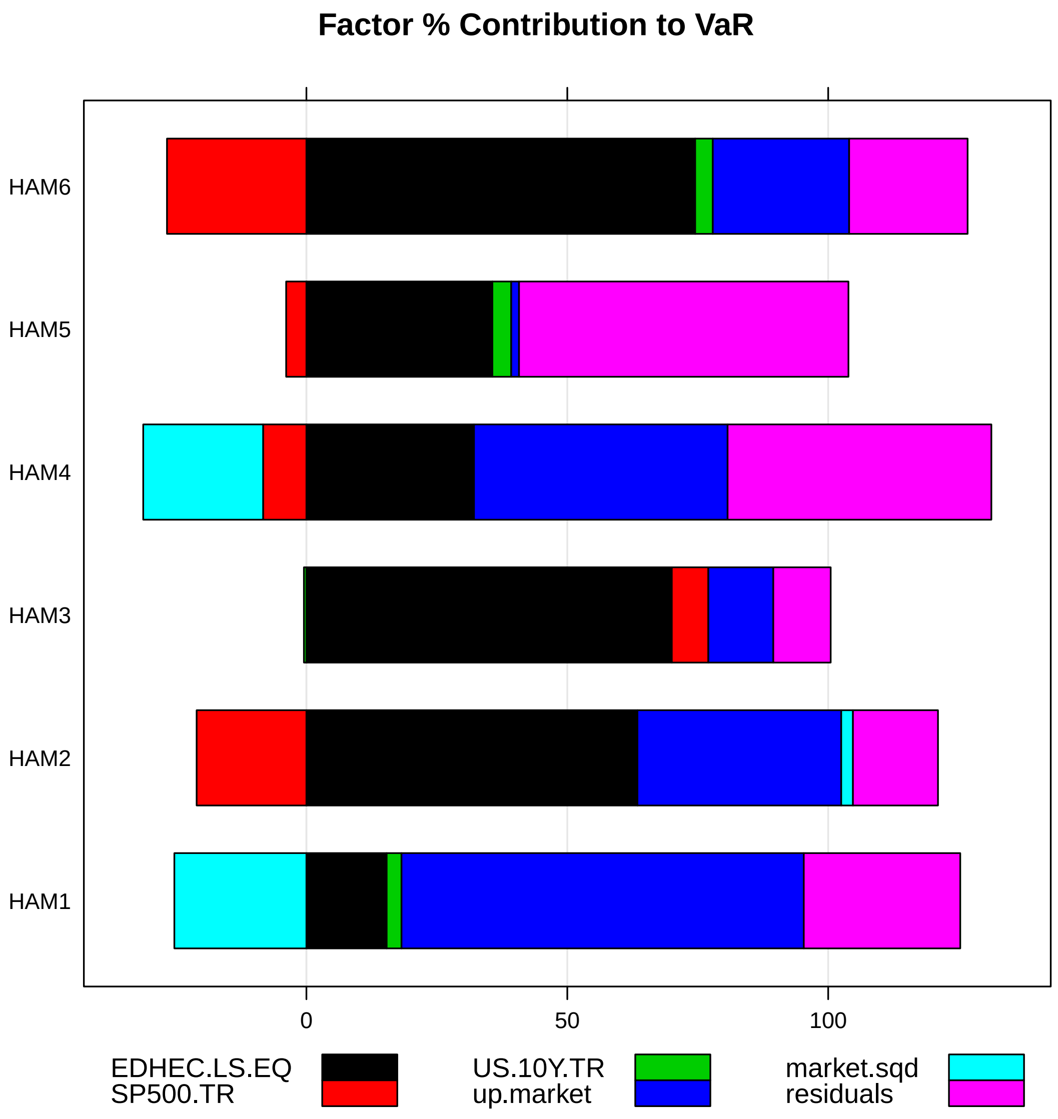


Figure 7: Percentage factor contribution to VaR

```
# get the component contributions to ES
decomp2$cES

##      EDHEC.LS.EQ SP500.TR US.10Y.TR up.market market.sqd residuals
## HAM1     0.00686  0.00000   0.00430  0.000000    0.02056    0.0221
## HAM2     0.01795 -0.00363   0.00000 -0.013210   -0.00166    0.0376
## HAM3     0.02879  0.00536  -0.00161  0.000000    0.00000    0.0261
## HAM4     0.03230 -0.01490   0.00000  0.000000    0.05325    0.0446
## HAM5     0.00395 -0.00164  -0.00247 -0.000756    0.00000    0.1076
## HAM6     0.03232 -0.01342   0.00250  0.000000    0.00000    0.0197

# get the marginal factor contributions to ES
decomp2$mES

##      EDHEC.LS.EQ SP500.TR US.10Y.TR up.market market.sqd residuals
## HAM1     0.02835  0.07625  -0.02075    0.0000   -0.00705      1.17
## HAM2     0.01117  0.00783  -0.00921   -0.0268   -0.00375      1.55
## HAM3     0.02283  0.07654  -0.01211    0.0000   -0.00687      1.20
## HAM4     0.02979  0.07659  -0.01395    0.0000   -0.00735      1.07
## HAM5     0.00243  0.00785  -0.00911   -0.0144   -0.00146      2.61
## HAM6     0.02486  0.03929  -0.01309    0.0000   -0.00199      1.25

# get the percentage component contributions to ES
decomp2$pcES

##      EDHEC.LS.EQ SP500.TR US.10Y.TR up.market market.sqd residuals
## HAM1        12.7     0.00      7.99     0.000      38.18      41.1
## HAM2        48.5    -9.80      0.00   -35.690      -4.48     101.5
## HAM3        49.1     9.15     -2.75     0.000       0.00      44.5
## HAM4        28.0   -12.93      0.00     0.000      46.20      38.7
## HAM5         3.7    -1.54     -2.31    -0.708       0.00     100.9
## HAM6        78.6   -32.63      6.09     0.000       0.00      47.9

# plot the percentage component contributions to ES
plot(fit.sub, which.plot.group=9, loop=FALSE)
```

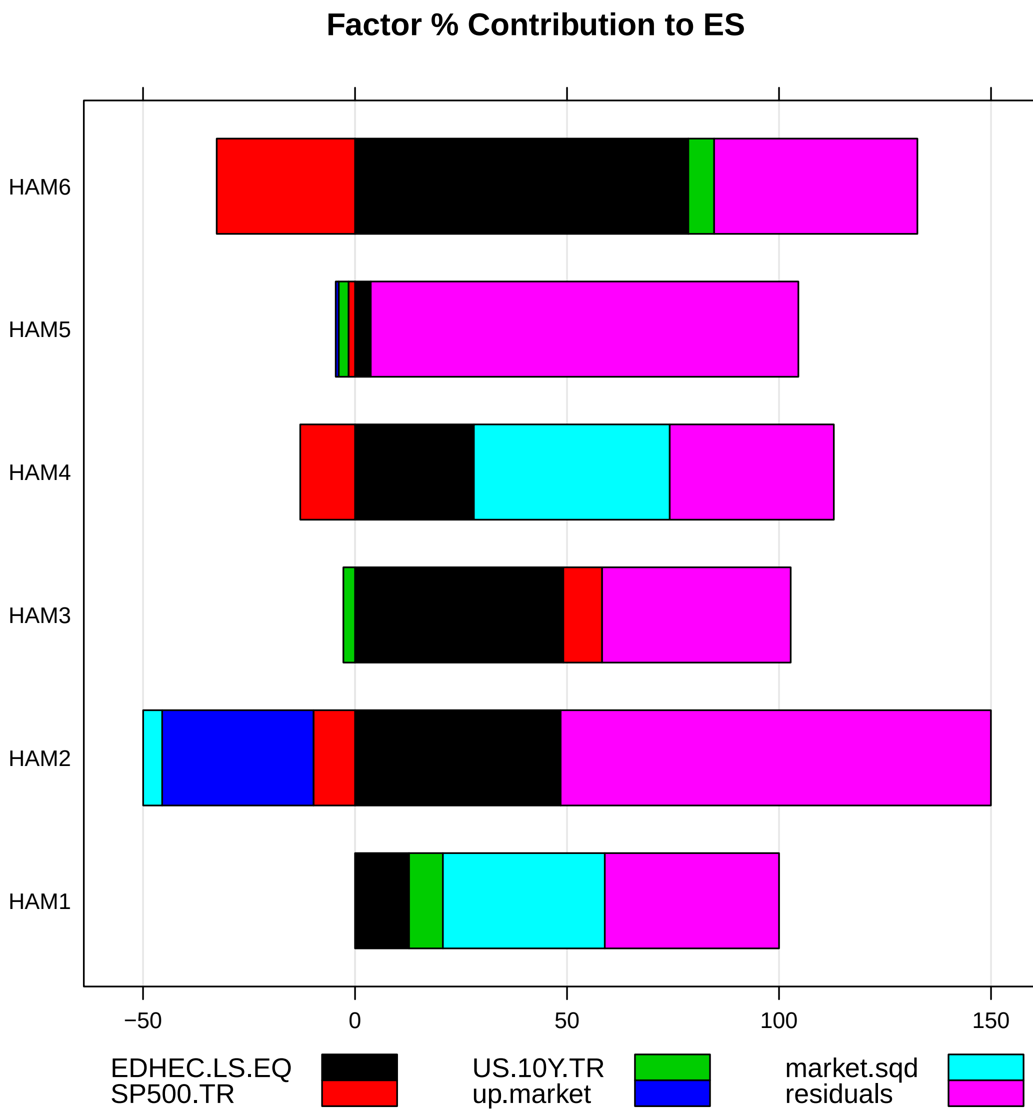


Figure 8: Percentage factor contribution to ES

4 Plot

Many types of individual asset (Figure 1) and group plots (Figures 2-8) have already been demonstrated. Let's take a look at all available arguments for plotting a "tsfm" object.

```
## S3 method for class 'tsfm'
plot(x, which.plot.group=NULL, max.show=6, plot.single=FALSE, asset.name,
     which.plot.single=NULL, colorset=(1:12), legend.loc="topleft", las=1,
     VaR.method="historical", loop=TRUE, ...)
```

4.1 Menu and looping

If the plot type argument (`which.plot.single` or `which.plot.group`) is not specified, a menu prompts for user input and the corresponding plot is output. Then, the menu is repeated (default) for user convenience in plotting multiple characteristics. Selecting '0' from the menu exits the current `plot.tsfm` call. Alternately, setting `loop=FALSE` will exit after plotting any one chosen characteristic without the need for menu selection.

For group plots (the default), the first `max.show` assets are plotted. For individual plots, `asset.name` is necessary if multiple assets were modeled in `x` and `plot.single=TRUE`. However, if the `fitTsfm` object `x` only contains one asset's factor model fit, `plot.tsfm` can infer this automatically, without user input.

Remarks:

- CUSUM plots (individual asset plot options 10, 11 and 12) are applicable only for `fit.method="OLS"`.
- Rolling estimates (individual asset plot option 13) is not applicable for `variable.slection="lars"`.
- `VaR.method` applies to group plots 9 and 10 (see menu in the next section), which are factor model risk ES and VaR decompositions respectively.

4.2 Group plots

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

```
plot(fit.sub)


## Make a plot selection (or 0 to exit):
```

```
##
##  1: Factor model coefficients: Alpha
##  2: Factor model coefficients: Betas
##  3: Actual and Fitted asset returns
##  4: R-squared
##  5: Residual Volatility
##  6: Factor Model Residual Correlation
##  7: Factor Model Return Correlation
##  8: Factor Contribution to SD
##  9: Factor Contribution to ES
## 10: Factor Contribution to VaR
##
## Selection:
```

Here's an example where the looping is disabled and the number of assets displayed is restricted to 4.

```
plot(fit.sub, which.plot.group=3, max.show=4, legend.loc=NULL, loop=FALSE)

## Displaying only the first 4 assets, since the number of assets > 'max.show'
```

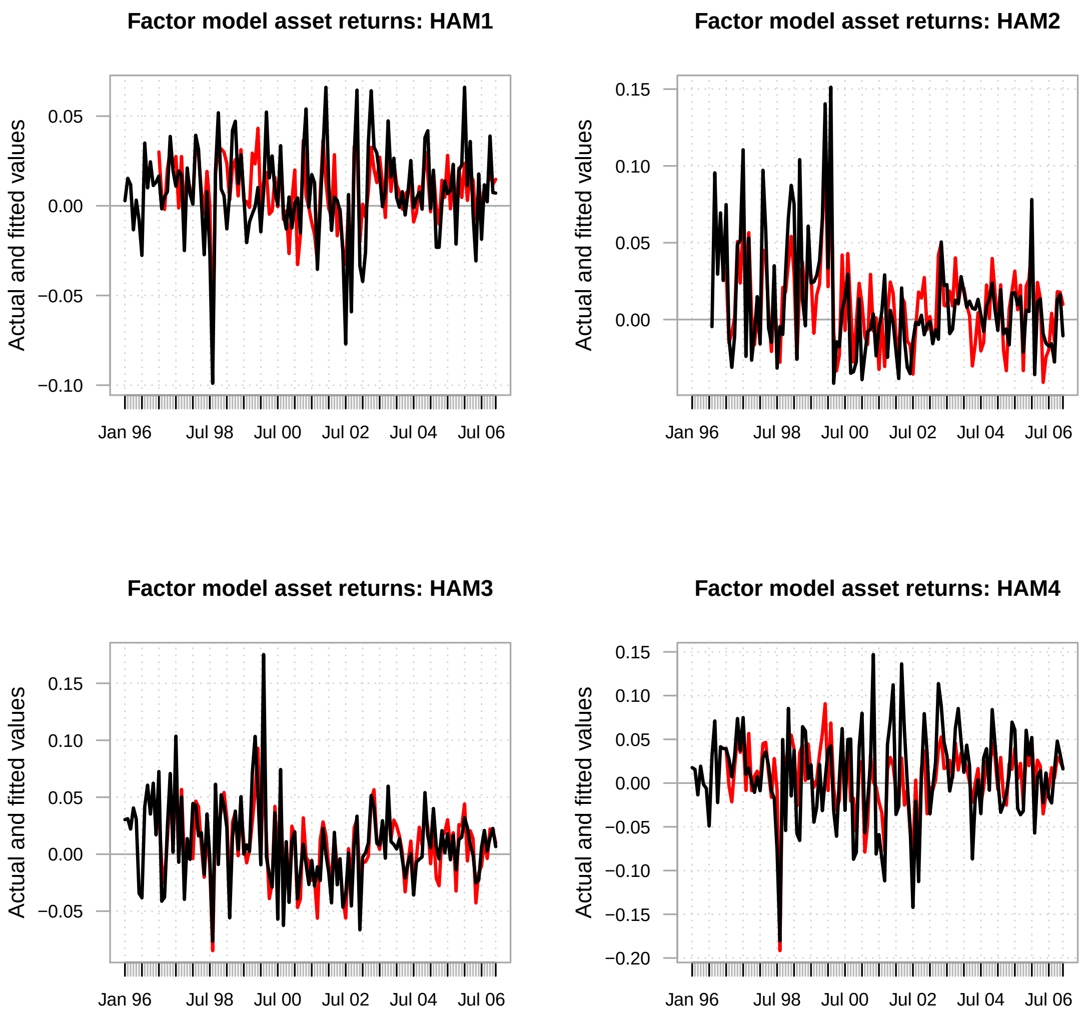


Figure 9: Actual and fitted factor model returns for the 1st 4 assets

4.3 Individual plots

Setting `plot.single=TRUE` enables individual asset plots. If there are more than one asset fit by the fit object, `asset.name` is also necessary. Here's the individual plot menu.

```
plot(fit.sub, plot.single=TRUE, asset.name="HAM1")


## Make a plot selection (or 0 to exit):
##
##  1: Time series plot of actual and fitted asset returns
##  2: Time series plot of residuals with standard error bands
##  3: Time series plot of squared residuals
##  4: Time series plot of absolute residuals
##  5: SACF and PACF of residuals
##  6: SACF and PACF of squared residuals
##  7: SACF and PACF of absolute residuals
##  8: Histogram of residuals with normal curve overlayed
##  9: Normal qq-plot of residuals
## 10: CUSUM test-Recursive residuals
## 11: CUSUM test-OLS residuals
## 12: Recursive estimates (RE) test of OLS regression coefficients
## 13: Rolling estimates over a 24-period observation window
##
## Selection:
```

Here are a few more examples.

```
plot(fit.sub, plot.single=TRUE, asset.name="HAM1", which.plot.single=2,
     loop=FALSE)
```

```
plot(fit.sub, plot.single=TRUE, asset.name="HAM1", which.plot.single=7,
     loop=FALSE)
```

```
plot(fit.sub, plot.single=TRUE, asset.name="HAM1", which.plot.single=8,
     loop=FALSE)
```

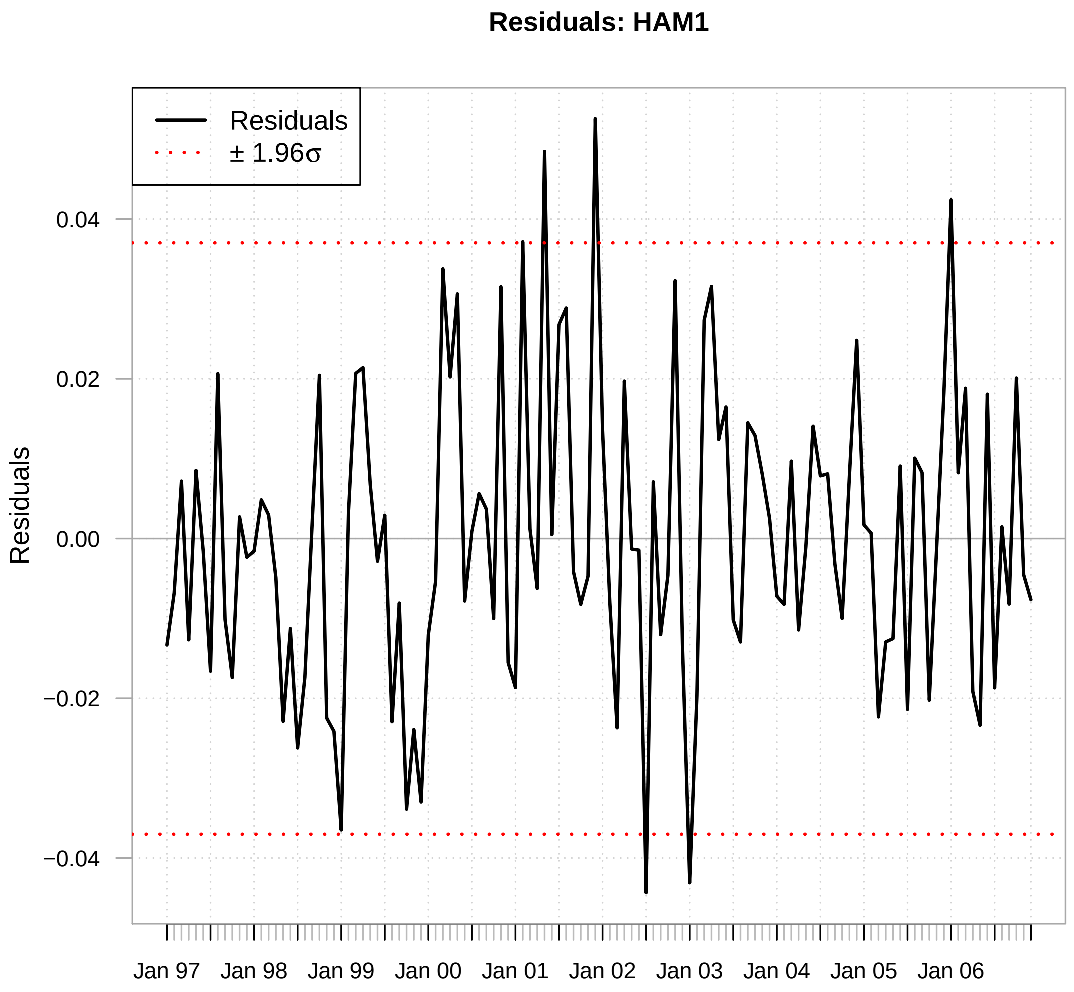


Figure 10: Time series plot of residuals with standard error bands: HAM1

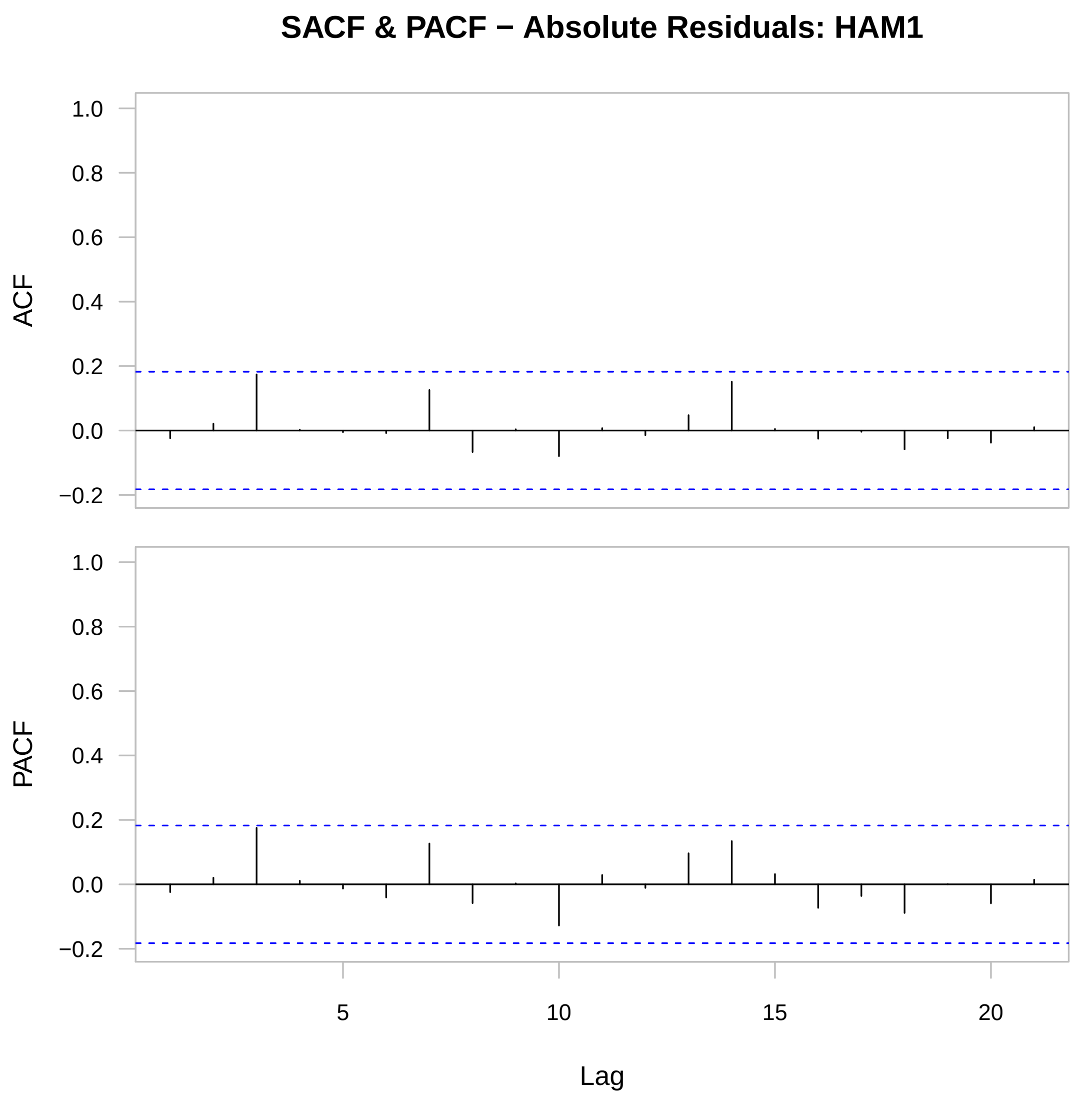


Figure 11: SACF and PACF of absolute residuals: HAM1

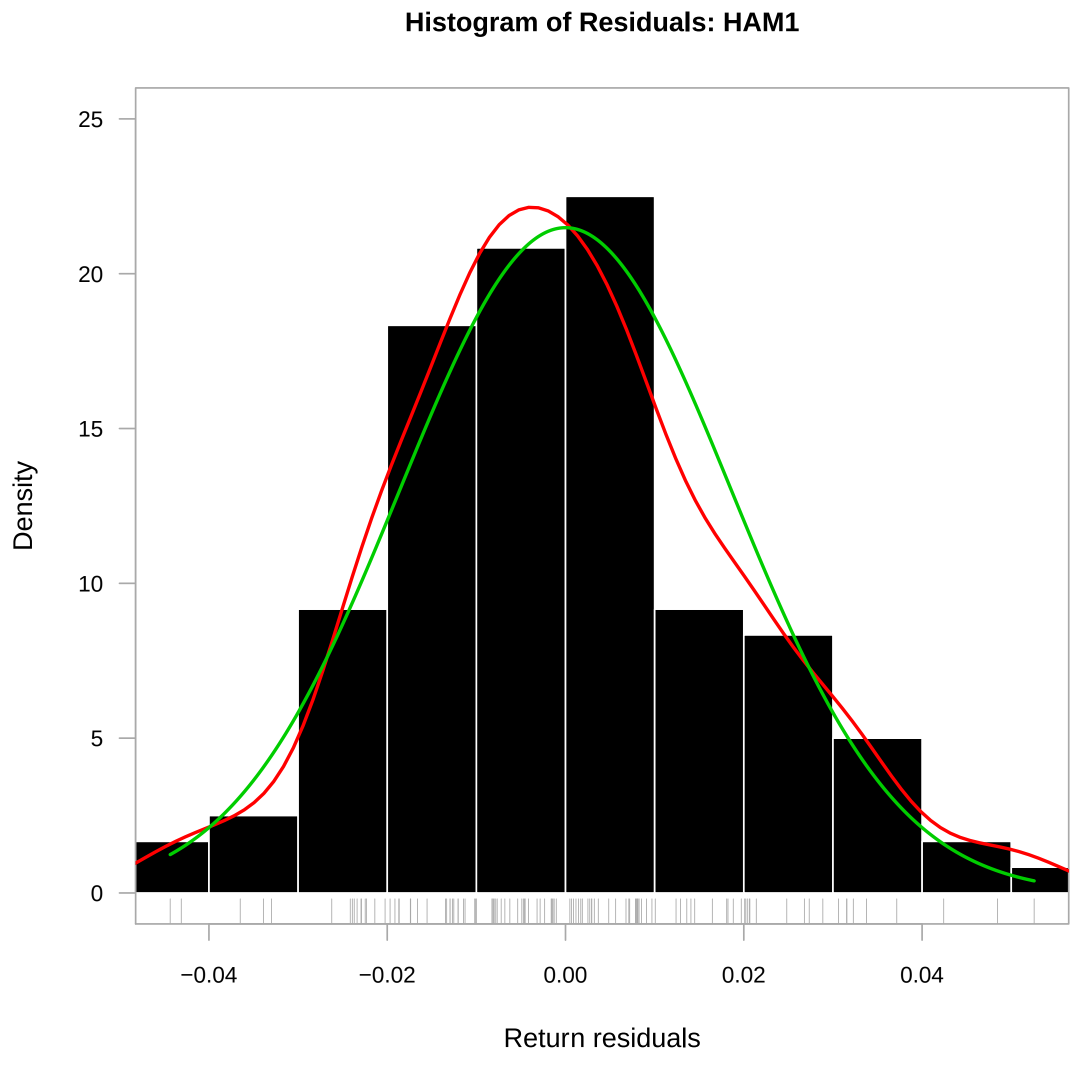


Figure 12: Histogram of residuals with normal curve overlayed for HAM1

References

N.-F. Chen, R. Roll, and S. A. Ross. Economic forces and the stock market. *Journal of business*, pages 383–403, 1986.

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

R. D. Henriksson and R. C. Merton. On market timing and investment performance. ii. statistical procedures for evaluating forecasting skills. *Journal of business*, pages 513–533, 1981.

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

W. F. Sharpe and W. Sharpe. *Portfolio theory and capital markets*, volume 217. McGraw-Hill New York, 1970.

J. Treynor and K. Mazuy. Can mutual funds outguess the market. *Harvard business review*, 44 (4):131–136, 1966.

E. Zivot and W. Jia-hui. Modeling financial time series with s-plus springer-verlag. 2006.