Robust Betas for PerformanceAnalytics

October 26, 2023

R. Douglas Martin and Dhairya Jain

Contents

1 Introduction

A time series Single Factor Model (SFM) has the general form

$$r_t = \alpha + \beta f_t + \epsilon_t \, t = 1, 2, \cdots, T \tag{1}$$

where r_t is a time series of asset returns, such as those for a stock, an ETF or a hedge fund, f_t is a factor return, ϵ_t is the error term, and α and β are unknown intercept and slope coefficients that need to be estimated based on observed asset and factor returns. The factor return f_t is typically either: (1) a market proxy $f_{M,t}$, such as the CRSP value-weighted market index $f_{M,t} = f_{CRSP,t}$ in the context of the Capital Asset Pricing Model (CAPM), or (2) an active manager's index benchmark such as the S&P500, Russell 1000, Russell 2000, and Russell 3000, among others.

The SFM errors for arbitrary intercept a and slope b values are defined as:

$$\epsilon_t(a,b) = r_t - a - bf_t,\ t = 1, 2, \cdots, T. \tag{2}$$

For estimates $a = \hat{\alpha}$ and $b = \hat{\beta}$, the above errors become the fitted model *residuals*

$$\hat{\epsilon}_t = \epsilon_t(\hat{\alpha}, \hat{\beta}) = r_t - \hat{\alpha} - \hat{\beta} f_t \tag{3}$$

and one has the fit-plus-residuals representation of the asset return:

$$r_t = \hat{\alpha} - \hat{\beta} f_t + \hat{\epsilon}_t. \tag{4}$$

The unknown coefficients α and β are currently almost universally estimated using the method of least squares (LS), which is computed by minimizing the sum of the squared errors $\sum_{t=1}^{T} \epsilon_t^2(a,b)$ with respect to a and b. Unfortunately, both asset returns and factor returns often contain outliers which adversely influence the LS intercept and slope estimates, and one needs a robust alternative which is computed as a complement to LS, or a replacement for LS, depending on the context.

In this Vignette we describe and illustrate the use of a highly robust estimator of SFM slope and intercept parameters. This estimator called the *mOpt* estimator (the Robust estimator), and it has an intuitive weighted-least-squares (WLS) interpretation that it minimizes the weighted sum of squared regression residuals $\sum_{t=1}^{T} w_t(a,b)\epsilon_t^2(a,b)$, where $w_t(a,b)$ is a special data-dependent weight function described in Section 3.

The *mOpt* estimator was recently introduced for time series factor models in Martin and Xia (2022), who used it to study the performance of mOpt relative to LS in estimating CAPM betas for the cross-section of liquid U.S. stocks from 1963 to 2018, and for fitting multifactor time series models such as the Fama-French 3 factor model.[1] The study revealed extensive adverse influence of outliers on the LS betas. In particular, it was shown that the LS and mOpt betas differ in absolute values by at least 0.3 for roughly 26% of microcap stocks, 14% of smallcaps and 7% of bigcaps, and by at least 0.5 for roughly 12% of microcap stocks, 5% of smallcaps and 2% of bigcaps.

Section 2 introduces the Robust SFM fitting functions in the PerformanceAnalytics R package, and illustrates their use for package `managers` data set. Section 3 provides some mathematical details for the mOpt Robust estimator, and Section 4 provides concluding comments.

[1]This paper appeared in the March 2022 issue of The Journal of Asset Management, and is available in opens source form at `https://link.springer.com/content/pdf/10.1057/s41260-022-00258-0.pdf`.

2 The PerformanceAnalytics Robust Betas Functions

PerformanceAnalytics package contains the following two main functions for computing LS and mOpt (Robust) SFM model fits:

- `chart.SFM`
- `SFM.fit.models`

The `chart.SFM` function computes both LS and Robust alphas and betas and overlays the corresponding straight line fits to a scatter plot of asset returns versus benchmark or market proxy returns. The `SFM.fit.models` function also computes both LS and Robust SFM fits, in order to provide: (a) a comparative tabular display of the LS and Robust alphas, betas, and related statistics, and (b) an optional display of any subset of 10 different comparative graphical displays of the LS and Robust SFM fits.

In addition the following function allows the user to compute either mOpt or LS robust SFM fits, with mOpt the default, for any combination of one or more sets of asset returns and one or more benchmarks:

- `SFM.coefficients`

In order to use these functions, an R user needs to first install the current version of PerformanceAnalytics from CRAN (`https://cran.r-project.org/web/packages/PerformanceAnalytics/index.html`), and load it with:

```
library(PerformanceAnalytics)
```

2.1 The `managers` Data Set

The following examples will use the `xts` time series data set `managers` included with PerformanceAnalytics, so we first load this data set and determine its class, dimensions, and names with the code:

```
data(managers)
class(managers)

## [1] "xts" "zoo"

dim(managers)

## [1] 132  10
```

```r
names(managers)
```

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

The results show that `managers` is an `xts` data object consisting of 10 time series for 132 months. Now we replace the last 4 names of `managers` with shorter convenient names with no spaces:

Next we use the function `tsPlotMP` from the PCRA package to create the time series plots shown in Figure 1.

```r
PCRA::tsPlotMP(managers, scaleType = "same", axis.cex = 0.5, stripText.cex = 0.5)
```

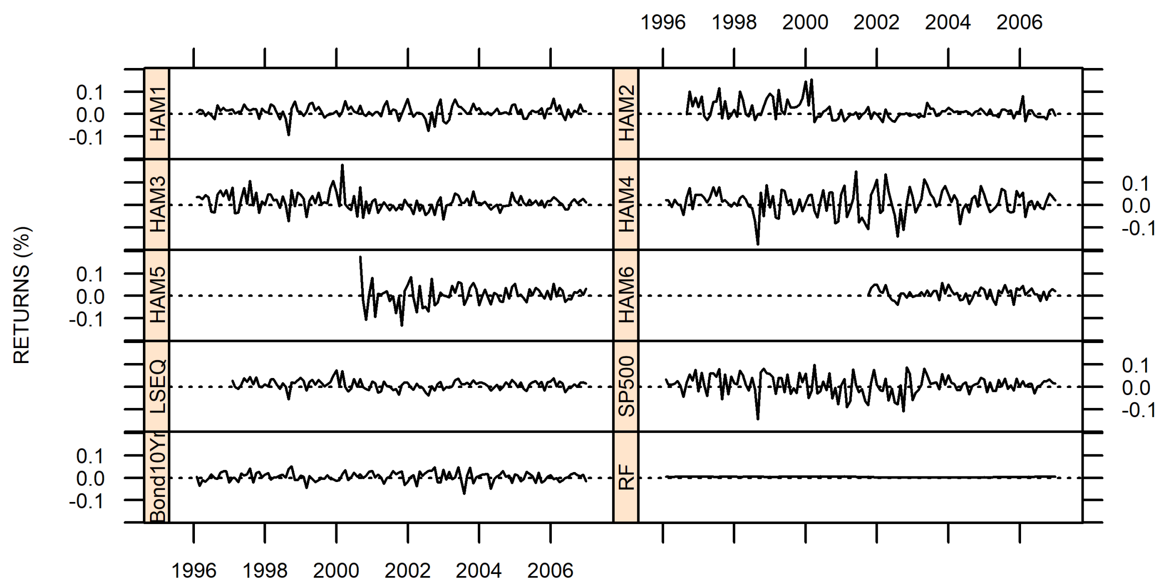

Figure 1: Time Series of managers Data

The figure shows that some of the times series data begins in January 1996, but some series begin later, and all series continue until December 2008. It will be convenient for the calculations below to use the maximum time window of the managers data such that none of the time series have missing data. This maximum length window is easily determine using the `na.omit` and `range` functions as follows:

```r
range(index(na.omit(managers)))

## [1] "2001-09-30" "2006-12-31"
```

In order to avoid using last four months fraction of the year 2001, we use the following code line to delete those four months and rename the result:

```r
mgrs <- managers["2002/"]  # 5 full years from 2002 through 2006
```

2.2 Use of `chart.SFM`

The function `chart.SFM` computes LS and mOpt robust alphas and betas, and makes a graphical display of the resulting LS and mOpt straight line fits, superimposed on the scatter plot of asset returns and factor returns. The arguments of `chart.SFM` are viewed with:

```r
args(chart.SFM)

## function (Ra, Rb, Rf = 0, main = NULL, ylim = NULL, xlim = NULL,
##     family = "mopt", xlab = NULL, ylab = NULL, legend.loc = "topleft",
##     makePct = FALSE)
## NULL
```

NOTE: With the default NULL optional arguments for the `xlim` and `ylim` axes limits, `chart.SFM` uses sensible data dependent values, as will be seen in resulting plot in the figure below. You can obtain more information about the arguments of `chart.SFM` by using the `help()` function:

```r
help(chart.SFM)
```

Figure 2 shows the result of using `chart.SFM` for the HAM6 returns and the S&P500 as the benchmark.

```r
chart.SFM(mgrs$HAM6, mgrs$SP500, mgrs$RF, makePct = TRUE)
```

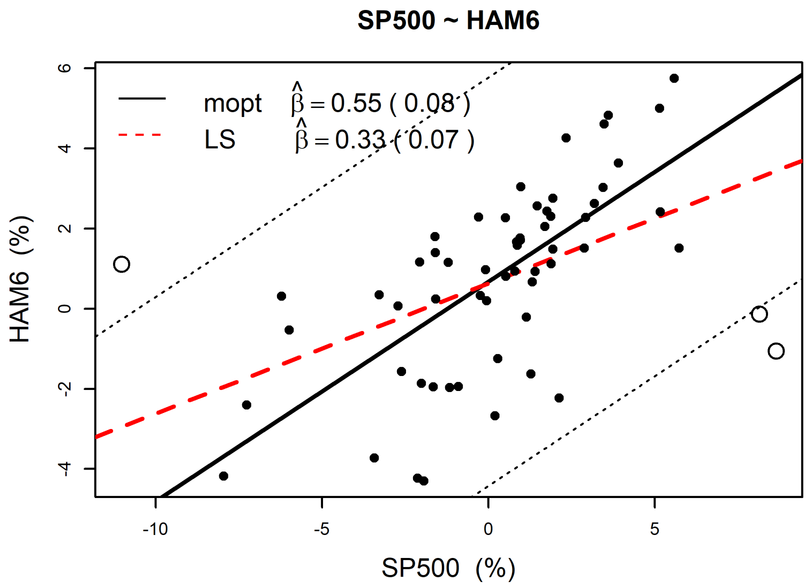

Figure 2: Robust mOpt Beta and LS Beta for HAM6

You can use a simple `for` loop to plot the mOpt and LS fits for any subset of the HAM1 through HAM6 funds and LSEQ, for example you do so for HAM3 through HAM6 with the code line:

```
for(k in 3:6){
    chart.SFM(mgrs[,k], mgrs$SP500, mgrs$RF, makePct = T,
        main = names(mgrs[,k]))
}
```

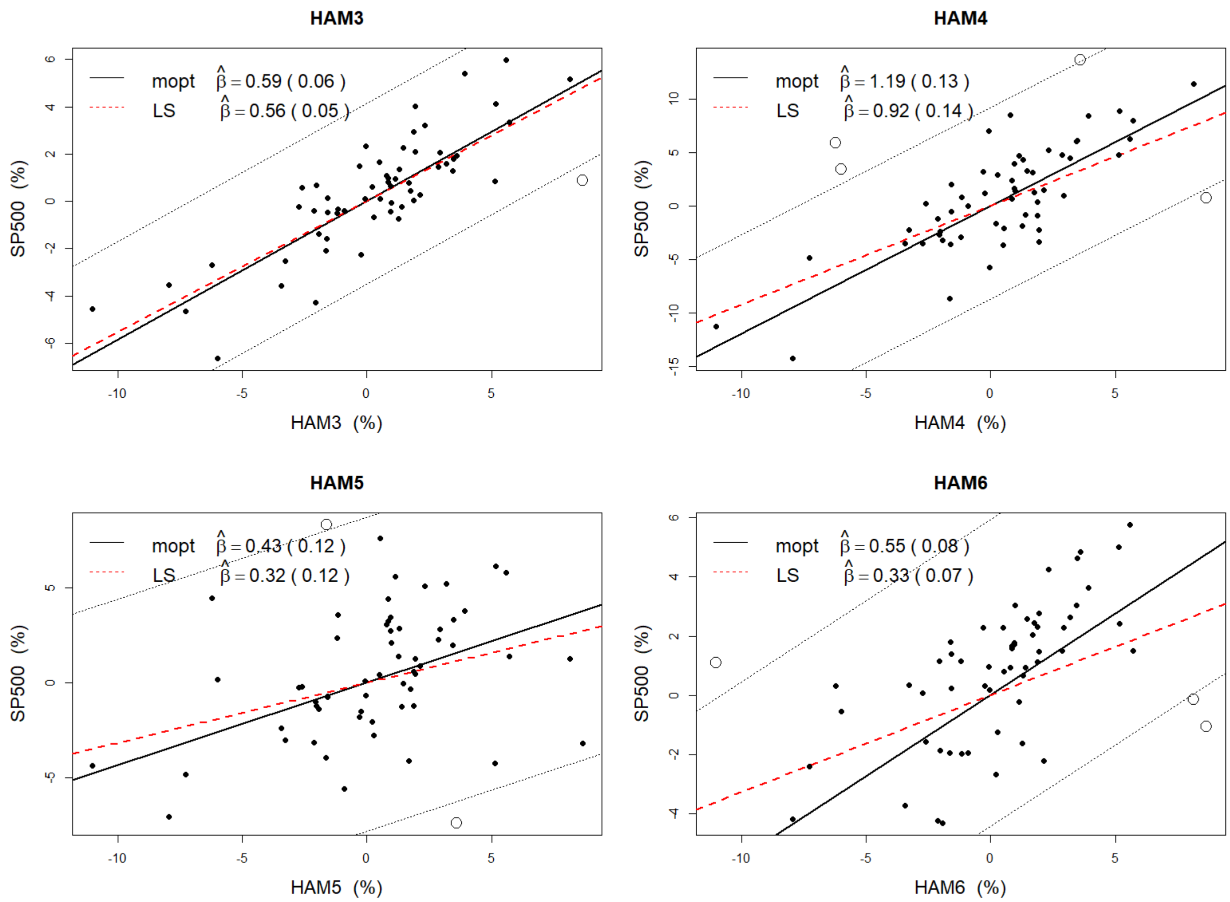

Figure 3: Robust mOpt Beta and LS Beta for HAM3 - HAM6

2.3 Use of `SFM.fit.models`

The function `SFM.fit.models` has two main capabilities. The first is to compute both LS and Robust alpha and beta coefficient estimates, along with their standard errors and t-statistics, and display them in easy-to-compare table form. The second is to make side-by-side graphical displays of Robust and LS model fitting results. First, we find out what the arguments of `SFM.fit.models` are with:

```
args(SFM.fit.models)

## function (Ra, Rb, Rf = 0, family = "mopt", which.plots = NULL,
##     plots = TRUE)
## NULL
```

Use the first code line below to compute the Robust and LS coefficients with no graphics, and save the result of the fitted models object fitHAM6, displays. Then use the next 3 code lines to see what the class of fitHAM6 is, display the LS and Robust alpha and beta coefficient estimates, and display a complete comparative LS and Robust statistics summary:

```
fitHAM6 <- SFM.fit.models(mgrs$HAM6, mgrs$SP500, Rf = mgrs$RF,
    plots = FALSE)
class(fitHAM6)

## [1] "lmfm"       "fit.models"

round(coef(fitHAM6),3)

##         (Intercept)  Beta
## LSFit        0.006 0.325
## RobFit       0.007 0.548

summary(fitHAM6)

##
## Calls:
##  LSFit: lm(formula = Alpha ~ Beta, data = merged, subset = subset)
## RobFit: RobStatTM::lmrobdetMM(formula = Alpha ~ Beta, data = merged,
##     subset = subset, control = RobStatTM::lmrobdet.control(family = family))
##
## Residual Statistics:
##               Min       1Q   Median      3Q     Max
##  LSFit: -0.04504 -0.01249 0.003436 0.01277 0.04062
## RobFit: -0.06474 -0.01175 0.003893 0.01185 0.06478
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept):  LSFit: 0.006331   0.002632   2.405   0.0194 *
##              RobFit: 0.006753   0.002392   2.824   0.0065 **
##
##        Beta:  LSFit: 0.325048   0.073839   4.402 4.67e-05 ***
##              RobFit: 0.547858   0.080665   6.792 6.55e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual Scale Estimates:
##  LSFit: 0.02028 on 58 degrees of freedom
## RobFit: 0.01943 on 58 degrees of freedom
##
## Multiple R-squared:
##  LSFit: 0.2504
## RobFit: 0.2364
```

To make a wide variety of plots that compare Robust and LS SFM model fitting results use

```
SFM.fit.models(mgrs$HAM6,mgrs$SP500,Rf = mgrs$RF)
```

which results in the following output in the Console:

```
Make plot selections (or 0 to exit):

1:  Normal QQ Plot of Residuals

2:  Kernel Density Estimate of Residuals

3:  Residuals vs.  Mahalanobis Distance

4:  Residuals vs.  Fitted Values

5:  Sqrt Residuals vs.  Fitted Values

6:  Response vs.  Fitted Values

7:  Residuals vs.  Index (Time)

8:  Overlaid Normal QQ Plot of Residuals

9:  Overlaid Kernel Density Estimate of Residuals

10:  Scatter Plot with Overlaid Fit(s)

Selection:
```

The first time you try this, we suggest that you enter each of the choices 1 through 10 after Selection, and after each selection you with see the corresponding plot type, then enter 0 to exit from the graphics display menu. If you just want a particular subset of graphical displays, e.g., types 2 and 7, just enter 2 to see the first plot and then enter 7 after Selection, followed by 0 to Exit. Alternatively, use of the command

```
SFM.fit.models(mgrs$HAM6, mgrs$SP500, Rf = mgrs$RF, which.plots = c(2,7))
```

results in the following in the Console:

```
Hit <Return> to see next plot:
```

Then pressing Enter results in display of the type 2 plot in the top of Figure 4, and pressing Enter again results in display of the bottom 7 plot in Figure 4, and then the above line in the Console disappears. Try it out.

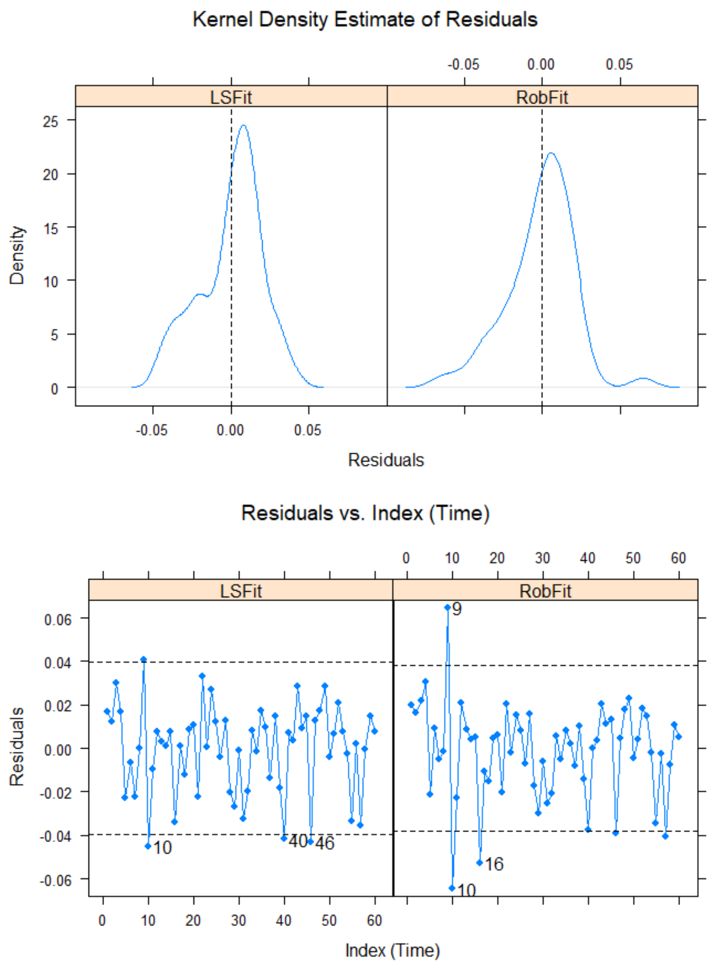

Figure 4: The Top Plot is Type 2 and the Bottom Plot is Type 7

2.4 Use of SFM.coefficients

The function `SFM.coefficients` was designed to support computing either Robust mOpt (the default) or LS multiple single factor model (SFM) for one or more assets and one or more benchmarks. Here are the arguments of `SFM.coefficients`:

```
args(SFM.coefficients)

## function (Ra, Rb, Rf = 0, subset = TRUE, ..., method = "Robust",
##     family = "mopt", digits = 3, benchmarkCols = T, Model = F,
##     warning = T)
## NULL
```

Here we use the first four managers HAM1, HAM2, HAM3, HAM4 as the assets, and the SP500 and Bond10Yr as the benchmarks:

```
funds <- mgrs[, c("HAM1", "HAM2", "HAM3", "HAM4")]
benchmarks <- mgrs[, c("SP500", "Bond10Yr")]
```

Now we make Robust and LS fits of the four mangers to the two benchmarks, and examine the class of the resulting `fit.Rob` (`fit.LS` has the same matrix class).

```
(fit.ROB <- SFM.coefficients(funds, benchmarks, method ="Robust"))

##      Alpha : SP500 Alpha : Bond10Yr Beta : SP500 Beta : Bond10Yr
## HAM1         0.006            0.012        0.595          -0.334
## HAM2         0.001            0.002        0.184          -0.275
## HAM3         0.003            0.008        0.586          -0.331
## HAM4         0.003            0.017        1.190          -0.286

(fit.LS <- SFM.coefficients(funds, benchmarks, method ="LS"))

##      Alpha : SP500 Alpha : Bond10Yr Beta : SP500 Beta : Bond10Yr
## HAM1         0.006            0.011        0.599          -0.426
## HAM2         0.002            0.004        0.216          -0.215
## HAM3         0.002            0.007        0.555          -0.378
## HAM4         0.008            0.015        0.923          -0.429

class(fit.ROB)

## [1] "matrix" "array"

summary(fit.ROB)
```

```
##  Alpha : SP500      Alpha : Bond10Yr   Beta : SP500     Beta : Bond10Yr
##  Min.   :0.00100   Min.   :0.00200   Min.   :0.1840   Min.   :-0.3340
##  1st Qu.:0.00250   1st Qu.:0.00650   1st Qu.:0.4855   1st Qu.:-0.3317
##  Median :0.00300   Median :0.01000   Median :0.5905   Median :-0.3085
##  Mean   :0.00325   Mean   :0.00975   Mean   :0.6388   Mean   :-0.3065
##  3rd Qu.:0.00375   3rd Qu.:0.01325   3rd Qu.:0.7438   3rd Qu.:-0.2833
##  Max.   :0.00600   Max.   :0.01700   Max.   :1.1900   Max.   :-0.2750
```

We note that since `method = "Robust"` is the default, that argument may be omitted in the first code line above.

By default, the benchmark Alpha and Beta results are in columns, and those of the assets are in rows. This is because portfolio managers often have many assets in their portfolio and only a few benchmarks. Note that `fit.Rob` and `fit.LS` are R matrix objects, and the results are printed with the default digits = 3. You can get the robust fit results displayed with benchmarks in rows, and 6 significant digits with the code line:

```
SFM.coefficients(funds, benchmarks, benchmarkCols = FALSE, digits = 6)

##                        HAM1      HAM2      HAM3      HAM4
## Alpha : SP500      0.005602  0.001172  0.003275  0.002534
## Alpha : Bond10Yr   0.011842  0.002482  0.007719  0.017016
## Beta : SP500       0.594613  0.183761  0.586422  1.189505
## Beta : Bond10Yr   -0.334156 -0.274526 -0.330747 -0.285967
```

You can use the function `SFM.alpha` if you only want alpha estimates, and use `SFM.beta` if you only want beta estimates. For example the following gives robust alphas

```
SFM.alpha(funds, benchmarks, digits = 4)

##      Alpha : SP500 Alpha : Bond10Yr
## HAM1        0.0058           0.0110
## HAM2        0.0023           0.0044
## HAM3        0.0024           0.0071
## HAM4        0.0078           0.0148
```

and the following gives LS betas:

```r
SFM.alpha(funds, benchmarks, method = "LS", digits = 2)

##      Alpha : SP500 Alpha : Bond10Yr
## HAM1          0.01             0.01
## HAM2          0.00             0.00
## HAM3          0.00             0.01
## HAM4          0.01             0.01
```

3 The mOpt Robust SFM Fit Mathematical Details

The SFM model 1 can be written in the following form:

$$r_t = \tilde{\mathbf{f}}_t'\boldsymbol{\theta} + s\epsilon_t,\ \ t = 1, 2, \cdots, T$$

where $\tilde{\mathbf{f}}_t' = (1, f_t)$, $\boldsymbol{\theta} = (\alpha, \beta)'$, and ϵ_t is a standardized error term that is scaled by the scale parameter s. The robust estimate $\hat{\boldsymbol{\theta}} = (\hat{\alpha}, \hat{\beta})$ is a solution of the weighted least squares (WLS) estimating equation

$$\sum_{i=1}^{T} w_t \tilde{\mathbf{f}}_t \left(r_t - \tilde{\mathbf{f}}_t' \hat{\boldsymbol{\theta}}\right) = 0 \tag{5}$$

where the w_t are the data–dependent weights

$$w_t = w_t(\hat{\boldsymbol{\theta}}; \hat{s}) = w_{mOpt}\left(\frac{r_t - \tilde{\mathbf{f}}_t' \hat{\boldsymbol{\theta}}}{\hat{s}}\right) \tag{6}$$

and the shape of the weight function $w_{mOpt}(t)$ is shown in Figure 5.

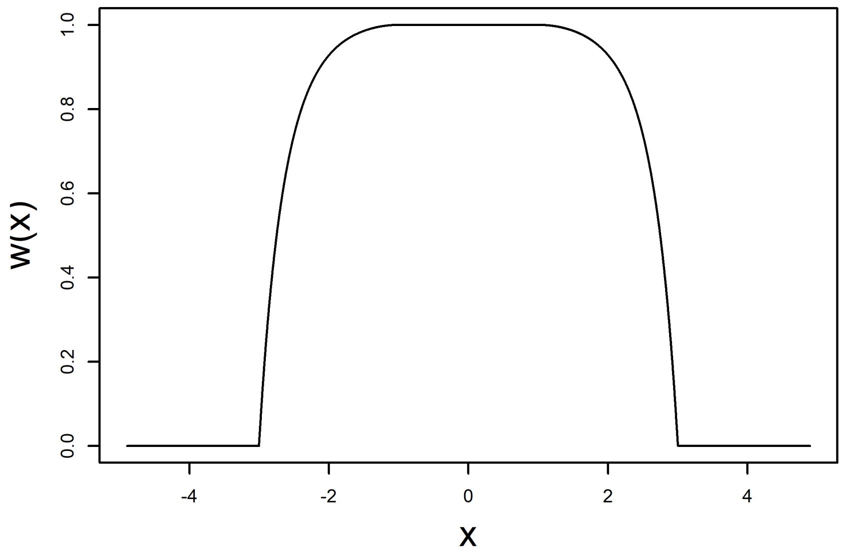

Figure 5: The mOpt weight function ($c = 3.00$)

The mOpt weight function gives a weight of 1 to all sufficiently small robustly scaled residuals $\hat{\epsilon}_t = (r_t - \tilde{\mathbf{f}}_t'\hat{\boldsymbol{\theta}})/\hat{s}$, and smoothly transitions to zero weight for robustly scaled residuals whose absolute is greater than 3.00. All asset and factor returns pairs $(r_t, \tilde{\mathbf{f}}_t)$ whose scaled residuals have absolute values larger than 3.0 are are *rejected* by the by the $\hat{\boldsymbol{\theta}}$ estimator. For normally distributed data and true parameter values, the probability that such a pair is rejected is only 0.27%, and the estimator is essentially equivalent to the LS estimator.

The weights $w_t = w_t(\hat{\boldsymbol{\theta}}; \hat{s})$ depend on the values of $\hat{\boldsymbol{\theta}}$, r_t, and $\tilde{\mathbf{f}}_t$. Consequently. the WLS equation (5) is a nonlinear function of the data $(r_t, f_t), t = 1, \cdots, T$, and $\hat{\boldsymbol{\theta}}$ must be computed with some type of iterative nonlinear algorithm. It is quite convenient that the estimate $\hat{\boldsymbol{\theta}}$ may be expressed in the nonlinear weighted least squares (WLS) mathematical form

$$\hat{\boldsymbol{\theta}} = \left(\sum_{i=1}^{T} w_t(\hat{\boldsymbol{\theta}}; \hat{s}) \tilde{\mathbf{f}}_t \tilde{\mathbf{f}}_t' \right)^{-1} \left(\sum_{i=1}^{T} w_t(\hat{\boldsymbol{\theta}}; \hat{s}) \tilde{\mathbf{f}}_t r_t \right) \tag{7}$$

which lends itself to the iterated weighted least squares (IRWLS) algorithm:

$$\hat{\boldsymbol{\theta}}^{k+1} = \left(\sum_{i=1}^{T} w_t(\hat{\boldsymbol{\theta}}^k; \hat{s}) \tilde{\mathbf{f}}_t \tilde{\mathbf{f}}_t' \right)^{-1} \left(\sum_{i=1}^{T} w_t(\hat{\boldsymbol{\theta}}^k; \hat{s}) \tilde{\mathbf{f}}_t r_t \right), \; k = 0, 1, 2, \cdots . \tag{8}$$

The mOpt Robust estimator is computed with the above IRWLS algorithm, using a highly robust but inefficient initial estimate $\hat{\boldsymbol{\theta}}^0$. For further details, see Martin and Xia (2022).

References

Martin, R. D. and Xia, D. Z. (Mar. 2022). “Efficient Bias robust Regression for Time series Factor Models”. In: *Journal of Asset Management*, pp. 1–20. ISSN: 1470-8272. DOI: 10.1057/s41260-022-00258-0. URL: https://link.springer.com/content/pdf/10.1057/s41260-%20022-00258-0.pdf.