Risk/Performance Measure Standard Error Vignette

Xin Chen and Douglas R. Martin

August 28, 2016

Abstract

This document introduces the new R package EstimatorStandardError. This package is a result of the Google Summer of Code 2016 (GSoC) project Standard Error of Risk and Performance Measures for Non-Normal and Serially Correlated Asset Returns. First we provide a brief introduction to the theoretical background that the authors consider important. Then the structure of the package is explained to facilitate further development of the package. Finally we provide examples of how to use the package and take full advantage of what the package has to offer.

1 Theoretical Background

1.1 Risk/Performance Measures and their Nonparametric Sample Estimators

We define Risk/Performance Measures as functionals $\rho(F)$ of the return distribution function $F(r)$. Table 1 shows the functional representations of the risk/performance measures we cover in this vignette.

Mean	$\mu(F) = \int_{-\infty}^{\infty} r dF(r)$
Standard Deviation	$\sigma(F) = \sqrt{\int_{-\infty}^{\infty}(r-\mu(F))^2 dF(r)}$
Sharpe Ratio	$SR(F) = \frac{\mu(F) - r_f}{\sigma(F)}$
Value-at-Risk	$VaR(F;\beta) = q_\beta(F) = F^{-1}(\beta)$
Expected Shortfall	$ES(F;\beta) = \frac{1}{\beta}\int_{-\infty}^{q_\beta(F)} r dF(r)$
Sortino Ratio	$SoR(F; MAR) = \frac{\mu(F) - MAR}{SSD(F; MAR)}$
STARR Ratio	$STARR(F;\beta) = \frac{\mu(F) - r_f}{ES(F;\beta)}$
LPM(MAR,k)	$LPM(F; MAR, k) = \int_{-\infty}^{MAR}(MAR - r)dF(r)$
OmegaRatio(c)	$OmegaRatio(F; c) = \frac{\int_c^{\infty}(r-c)dF(r)}{\int_{-\infty}^{c}(c-r)dF(r)}$

Table 1: List of Risk/Perforamce Measures

In reality, the true distribution F is unknown so the true functional ρ is unknown and we need a way compute an estimator of the function based on observed returns data. There are two general ways of forming estimators of the functional ρ, parametric and non-parametric. For a discussion see Martin and Zhang (2015) available at http://ssrn.com/abstract=2747179. Here we take the non-parametric approach obtained by replacing F by the empirical distribution function F_n that puts mass $\frac{1}{n}$ at each return observation. This results in the sample estimators of the risk/performance measures $\rho(F)$ shown in Table 2.

Sample Mean	$\hat{\mu}_n(F_n) = \int_{-\infty}^{\infty} r dF_n(r)$
Sample SD	$\hat{\sigma}_n(F_n) = \sqrt{\int_{-\infty}^{\infty}(r-\hat{\mu}_n)^2 dF_n(r)}$
Sample SR	$\widehat{SR}_n(F_n) = \frac{\hat{\mu} - r_f}{\hat{\sigma}}$
Sample VaR	$\widehat{VaR}_n(F_n) = \hat{q}_\beta = F_n^{-1}(\beta)$
Sample ES	$\widehat{ES}_n(F_n;\beta) = \frac{1}{\beta}\int_{-\infty}^{\hat{q}_\beta} r dF_n(r)$
Sample SoR	$\widehat{SoR}_n(F_n; MAR) = \frac{\hat{\mu}_n - MAR}{\widehat{SemiDeviation}_n(F_n; MAR)}$
Sample STARR	$\widehat{STARR}_n(F_n) = \frac{\hat{\mu}_n - r_f}{\widehat{ES}_n}$
Sample LPM	$LPM_n(F_n; MAR, k) = \int_{-\infty}^{MAR}(MAR - r)dF_n(r)$
Sample OmegaRatio	$OmegaRatio_n(F_n; c) = \frac{\int_c^{\infty}(r-c)dF_n(r)}{\int_{-\infty}^{c}(c-r)dF_n(r)}$

Table 2: List of Nonparametric Sample Estimators for Risk/Performance Measures

The sample estimators are themselves random variables therefore have standard errors. The EstimatorStandard Error package provides two general methods for computing the standard error of these estimators for the case of independent and identically distributed (i.i.d.) returns and for serially correlated returns. One of the methods is the well-known bootstrap where the simple sampling withn replacement bootstrap works well for i.i.d. returns and the block bootstrap is needed for serially correlated returns. The other method is a completely new method that uses the statistical influence functions in a very simple way for the case of i.i.d. returns and in a much more sophisticated way for the case of serially correlated returns.

1.2 Influence Functions

Let $T(F)$ be a functional of space of cumulative distribution functions (distributions functions, DF's for short) and define the perturbation of F as $F_\gamma = (1-\gamma)F + \gamma\delta_r$, where δ_r is the probability

measure that puts mass one at point r. The influence function of T introduced by (Hampel 1974)

$$IF(r;T,F) = \lim_{\gamma\to 0}\frac{T(F_\gamma)-T(F)}{\gamma} \tag{1}$$

$$= \frac{d}{d\gamma}T(F_\gamma)\Big|_{\gamma=0} \tag{2}$$

The influence function is a Gâteaux (directional) derivative of the functional T in the direction of δ_r evaluated at F. A basic property of the influence function is that

$$\mathbb{E}\{IF(r;T,F)\} = 0 \tag{3}$$

Table 3 provides the formulas of the influence functions of the risk/performance measures covered in this package, and Figure 1 shows the plots of these influence functions for the case of a normal distribution with mean and standard deviation both equal to one.

Mean	$r-\mu(F)$
Standard Deviation	$(r-\mu(F))^2-\sigma^2(F)$
Sharpe Ratio	$\frac{r-\mu(F)}{\sigma(F)}-\frac{\mu(F)}{2\sigma^3(F)}((r-\mu(F))^2-\sigma^2(F))$
Value-at-Risk	$\frac{\alpha-\mathbb{1}(r\le VaR_\alpha(F))}{f(VaR_\alpha(F))}$
Expected Shortfall	$\frac{-r+VaR_\alpha(F)}{\alpha}\mathbb{1}(r\le VaR_\alpha(F))+VaR_\alpha(F)-ES_\alpha(F)$
Sortino Ratio	$-\frac{SoR(F)}{2\sigma_-^2(F)}(r-MAR)^2\mathbb{1}(r\le MAR)+\frac{r-\mu(F)}{\sigma_-(F)}+\frac{SoR(F)}{2}$
STARR	$\frac{r-\mu(F)}{ES_\alpha(F)}-\frac{STARR(F)}{ES_\alpha(F)}\left[\frac{-r-VaR_\alpha(F)}{\alpha}\mathbb{1}(r\le VaR_\alpha(F))+VaR_\alpha(F)-ES_\alpha(F)\right]$
LPM(MAR,1)	$(MAR-r)\,I\,(r\le MAR)-LPM(MAR,1)$
LPM(MAR,2)	$\frac{(r-MAR)^2\,I\,(r\le MAR)-LPM^2\,(MAR,2)}{2LPM(MAR,2)}$
OmegaRatio	$\frac{1}{\Omega}\left[(r-c)\,I\,(r\ge c)-\Omega_+\right]-\frac{\Omega_+}{\Omega_-^2}\left[(c-r)\,I\,(r\le c)-\Omega_-\right]$

Table 3: Influence Functions

Figure 1 shows the plots of Risk and performance meansure influence functions for normal distribution $N(1,1)$.

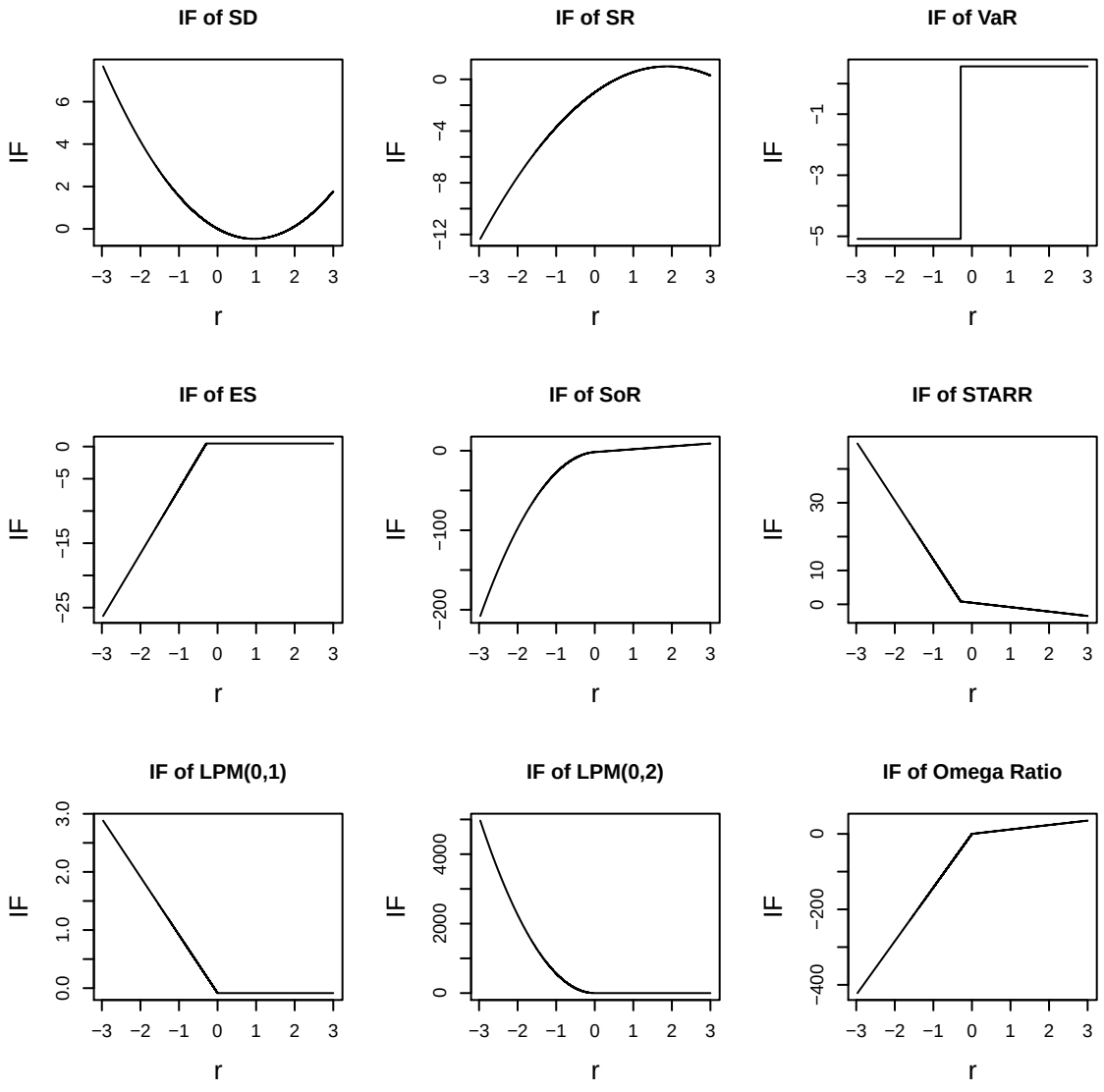

Figure 1: Influence Functions of Risk/Performance Measures

1.3 Standard Error of Nonparametric Sample Estimators via Influence Functions

Filipova (1962) showed that the difference between the estimator $T(F_n)$ and its asymptotic value $T(F_\theta)$ can be expressed as the following linear combination of influence functions of the returns at each time

$$\sqrt{n}(T(F_n) - T(F_\theta)) = \frac{1}{\sqrt{n}} \sum_{t=1}^{n} IF(r_t; T, F_\theta) + remainder \tag{4}$$

where the remainder goes to zero as $n \to \infty$ in a probabilistic sense. Using equation (3), for i.i.d. returns the asymptotic variance is given by

$$V(T(F_n)) = \mathbb{E}\{IF^2(r_1; T, F)\} \tag{5}$$

For finite samples the non-parametric estimator of the asymptotic variance is obtained by replacing F with the empirical distributionn function F_n:

$$\hat{V}_n(T(F_n)) = \frac{1}{n} \sum_{t=1}^{n} IF^2(r_t; T, F_n) \tag{6}$$

For serially correlated returns of length and ignoring the remainder term, the variance of the right-hand side of (4) is

$$\hat{V}_n(T(F_n)) = \sum_{\ell=-(n-1)}^{n-1} (1 - \frac{|\ell|}{n}) Cov\{IF(r_1), IF(r_{1+\ell})\} \tag{7}$$

$$= \sum_{\ell=-(n-1)}^{n-1} (1 - \frac{|\ell|}{n}) C_{IF}(\ell) \tag{8}$$

As $n \to \infty$, the asymptotic variance is given by

$$V(T(F_n)) = \sum_{\ell=-\infty}^{\infty} C(\ell) \tag{9}$$

$$= \sum_{\ell=-\infty}^{\infty} \mathbb{E}\{IF(r_1; T, F) IF(r_{1+\ell}; T, F)\} \tag{10}$$

The finite sample estimate of this quantity will be discussed in further detail in the following section.

In practice a finite sample standard error is usually computed by dividing a finite-sample estimate of the square root of the asymptotic variance formula by $\sqrt{n}$.

1.4 Computing Standard Errors for Serially Correlated Data using a Frequency Domain Representation and Generalized Linear Model Fitting

The finite sample estimate of the standard error for i.i.d data has already been discussed in the previous section. When the data is serially correlated, one is tempted to simply plug in the sample covariance estimate. Howver, this is not a good idea for several reasons. First of all one cannot use the covariance estimates for all avilable lags because they are poorly estimated at large lags. Second, one cannot simply truncate them as some specified lag because amonth other things it can result in negative estimate for the standard error. In the econometric literature (for example see Newey and West, 1987) one kind of solution has been dominant, which is to use a kernel weighted sum of the sample covariances as an approximation of the estimator asymptotic variance.

This package implements a new and better method of computing standard errors for serially correlated returns based on a frequency domain representation of the variance of the estimator variance with serial correlation. It takes advantage of the following relationship.

- Spectral Density and Autocovariance Functions are a Fourier Transform pair

$$S(f) = \sum_{\ell=-\infty}^{\infty} C(\ell) exp\{-2\pi i f \ell\} \tag{11}$$

$$C(\ell) = \int_{-1/2}^{1/2} S(f) exp\{2\pi i f \ell\} df \tag{12}$$

- Relationship between Spectral Density and Asymptotic Variance

$$S(0) = \sum_{\ell=-\infty}^{\infty} C(\ell) \tag{13}$$

$$= V(T(F_n)) \tag{14}$$

This implies that the variance is equal to the spectral density at frequency zero. Therefore, our method consists of roughly three steps.

1. Compute the periodograms of the time series of the influence functions.[1]

$$IF(r_t; T, F) = \frac{d}{d\gamma} T(F_\gamma)\Big|_{\gamma=0} \tag{15}$$

2. Fit a polynomial to the periodogram using Generalized Linear Model with Elastic Net regularizatoin.

3. Use the fitted polynomial to estimate the spectral density at frequency zero, which is equal to the variance we are looking for.

Technical details about this method are provided in the Appendix.

2 Structure of the EstimatorStandardError Package

This package implements two types methods of computing the standard error of nonparametric sample estimators. The first type is the influence function approach introduced in the previous section. The second type is a bootstrap method. Wrapper functions are implemented in this package that uses the R boot package function boot() for i.i.d data or tsboot() for serially correlated time series data from the boot package. For details, refer to the documentation of the R boot package. The structure of our package is illustrated in Figure 2.

[1]Note that the raw peridogrmams are not good approximations of the spectral density and suitably smoothed periodograms provide much better estimates (Percival and Walden 1993).

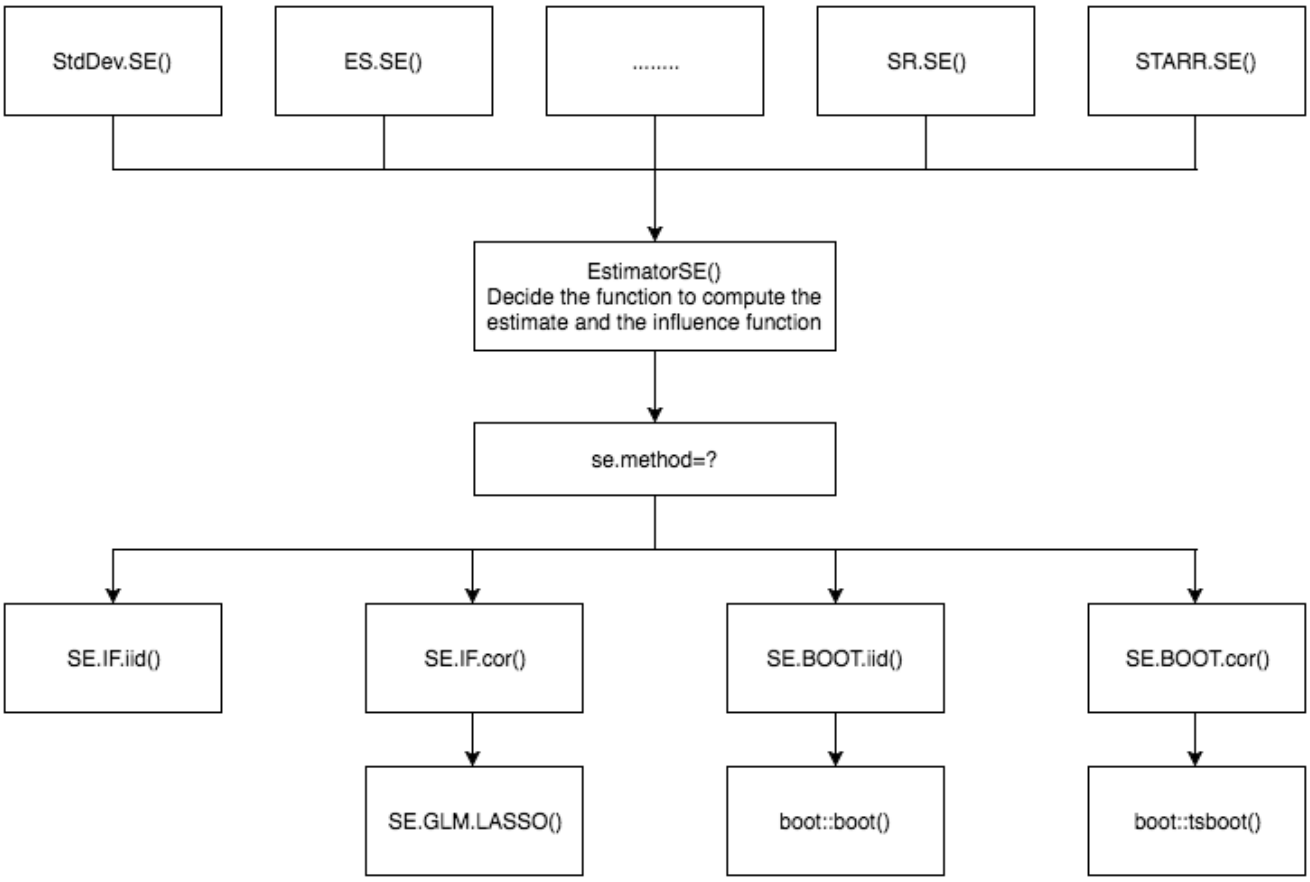

Figure 2: Structure of EstimatorStandardError Package

This structure facilitates future expansion of the package by separating the code associated with specific estimators from the code associated with computing standard errors. To add more estimators, one only need to create three new functions. The first function is the wrapper function xxx.SE(), such as ES.SE(). The second function is the function to compute the estimate, such as ES(). The third fucntion is the influence function xxx.IF(), such as ES.IF(). Then add the new function into the EstimatorSE() function.

3 Using the EstimatorStandardError Package

In this section, we show how to use the EstimatorStandardError package.

3.1 Installation

Currently the package is only on GitHub, from where it can be installed with the following R commands:

```
require(devtools)
install_github("chenx26/EstimatorStandardError")
#--------- Install h2o package
# install.packages('h2o',repos =
# 'http://cran.us.r-project.org/') Need to have already
# installed Java Development Kit (JDK) to get h2o running
```

The current version of the package uses the "h2o" package, which is a front-end to the Java H2O package. In order to use h2o and H2O you need to install the latest version of Java Development Kit (JDK) at `http://www.oracle.com/technetwork/java/javase/downloads/index.html`.

3.2 Example Code

The following code shows how to use the main functionalities provided in this package. Two observations stand out from the results. The first is that for most of the hedge funds the IFiid and IFcor yield very similar S.E.'s, indicating lack of serial correlation impact on the S.E. However for three of the funds, CA, EMN and RV, the IFcor S.E. is substantially larger than the IFiid. The second is that there is reasonable agreement between BOOTiid and IFiid, and between BOOTcor and IFcor when the number of bootstrap replicates is sufficiently large (in our example code we use 500 replicates, which was determined sufficient to get good block bootstrap estimates).

```
#--------- Load the package
library(EstimatorStandardError)

#--------- Initialize h2o instance
h2o.init()

##
## H2O is not running yet, starting it now...
##
## Note:  In case of errors look at the following log files:
##     /var/folders/yk/zznqdm8x1xz0zqfhsswcwjg80000gp/T//RtmponjhKD/h2o_cathyliu_start
##     /var/folders/yk/zznqdm8x1xz0zqfhsswcwjg80000gp/T//RtmponjhKD/h2o_cathyliu_start
##
##
```

```
## Starting H2O JVM and connecting: .. Connection successful!
##
## R is connected to the H2O cluster:
##     H2O cluster uptime:         1 seconds 818 milliseconds
##     H2O cluster version:        3.8.3.3
##     H2O cluster name:           H2O_started_from_R_cathyliu_pnz848
##     H2O cluster total nodes:    1
##     H2O cluster total memory:   0.12 GB
##     H2O cluster total cores:    8
##     H2O cluster allowed cores:  2
##     H2O cluster healthy:        TRUE
##     H2O Connection ip:          localhost
##     H2O Connection port:        54321
##     H2O Connection proxy:       NA
##     R Version:                  R version 3.3.0 (2016-05-03)
##
## Note:  As started, H2O is limited to the CRAN default of 2 CPUs.
##        Shut down and restart H2O as shown below to use all your CPUs.
##            > h2o.shutdown()
##            > h2o.init(nthreads = -1)

#--------- Load data
data(edhec)
colnames(edhec)=c("CA", "CTAG", "DIS", "EM",
                  "EMN", "ED", "FIA", "GM", "L/S",
                  "MA", "RV", "SS", "FoF")

#--------- Set number of replicates for bootstrapping
nboot=500

#--------- Set random seed
set.seed(123)

#--------- Expected Shortfall estimates and their S.E.'s
res.ES=ES.SE(edhec, p=.95, method="historical",nsim = nboot,
        se.method = c("IFiid","IFcor","BOOTiid","BOOTcor"))
```

```r
printSE(res.ES, round.digit = 3)
```

```
##        ES   IFiid   IFcor BOOTiid BOOTcor
## 1  -0.049 (0.018) (0.027) (0.013) (0.021)
## 2  -0.045 (0.004) (0.004) (0.004) (0.004)
## 3  -0.043 (0.013) (0.013) (0.011) (0.013)
## 4  -0.089 (0.024) (0.024) (0.023) (0.022)
## 5  -0.019 (0.008) (0.013) (0.007) (0.009)
## 6  -0.044 (0.011) (0.011)  (0.01)  (0.01)
## 7  -0.042 (0.016) (0.016) (0.013) (0.016)
## 8  -0.023 (0.003) (0.003) (0.003) (0.003)
## 9  -0.044 (0.009) (0.009) (0.007) (0.009)
## 10 -0.025 (0.006) (0.006) (0.006) (0.004)
## 11 -0.031  (0.01) (0.015) (0.009)  (0.01)
## 12 -0.111 (0.013) (0.013) (0.012) (0.019)
## 13 -0.039 (0.009) (0.009) (0.008)  (0.01)
```

```r
#--------
# We recommend that you try out the following code for
# risk estimators Standard Deviation (volatility) and
# Value-at-Risk (VaR), and for the performance
# estimators Sharpe ratio, Sortino ration and STARR ratio.
# Just remove the comment indicators #.

#--------- Value-at-Risk and the standard error
#res.VaR=VaR.SE(edhec, p=.95, method="historical",nsim = nboot,
#    se.method = c("IFiid","IFcor","BOOTiid","BOOTcor"))
#printSE(res.VaR, round.digit = 3)

#--------- Standard Deviation and the standard error
#res.StdDev=StdDev.SE(edhec,nsim = nboot,
#    se.method = c("IFiid","IFcor","BOOTiid","BOOTcor"))
#printSE(res.StdDev, round.digit = 3)

#--------- Sharpe Ratio and the standard error
#res.SR=SharpeRatio.SE(edhec, FUN = "StdDev",nsim = nboot,
```

```
#    se.method = c("IFiid","IFcor","BOOTiid","BOOTcor"))
#printSE(res.SR, round.digit = 3)

#--------- Sortino Ratio and the standard error
#res.SoR=SortinoRatio.SE(edhec, MAR = 0.1, nsim = nboot,
#    se.method = c("IFiid","IFcor","BOOTiid","BOOTcor"))
#printSE(res.SoR, round.digit = 3)

#--------- STARR ratio and the standard error
#res.STARR=STARR.SE(edhec, alpha = 0.1,nsim = nboot,
#    se.method = c("IFiid","IFcor","BOOTiid","BOOTcor"))
#printSE(res.STARR, round.digit = 3)

#--------- LPM and the standard error
#res.LPM=LPM.SE(edhec, nsim = nboot, const = 0, k = 1,
#    se.method = c("IFiid","IFcor","BOOTiid","BOOTcor"))
#printSE(res.LPM, round.digit = 5)

#--------- OmegaRatio and the standard error
#res.Omega=OmegaRatio.SE(edhec, nsim = nsim, const = 0,
#    se.method = c("IFiid","IFcor","BOOTiid","BOOTcor"))
#printSE(res.Omega, round.digit = 5)

#--------- Close h2o instance
h2o.shutdown(prompt=FALSE)

## [1] TRUE
```

Appendix

We use a frequency domain based method to compute standard errors of risk and performance measure estimators as follows. The use of a simpler form of the method that was not adequate for our application was introduced by Heidelberger and Welch (1981).

1. Compute the periodograms $\mathbb{I}(n/N)$ and the corresponding discrete Fourier Transform (DFT) freqeuncies $f_n = n/N$

2. Build the matrix for independent variables

$$F = \begin{bmatrix} 1 & f_1 & \cdots & f_1^d \\ \vdots & \vdots & \vdots & \vdots \\ 1 & f_N & \cdots & f_N^d \end{bmatrix} \tag{16}$$

3. Fit a GLM-EN model (details in the following section) with input matrix R and response vector $\mathbb{I}(n/N)$

4. Predict the value of the GLM-EN model at frequency 0 to get $\hat{S}(0)$

The Generalized Linear Model with Elastic Net Regularization (GLM-EN) method assumes that the relationsip between the dependent variables and the independent variables can be expressed as

$$g(\mu_i) = x_i\beta \tag{17}$$

Where $g(z)$ is the link function, μ_i is the expectation of the ith dependent variable, x_i is the vector of the ith independent variable and β is the vector of the coefficients.

Then β is computed by solving the following optimization problem:

$$\hat{\beta} = \arg\min_{\beta,x} \mathit{Deviance}(\beta, x, y) + \lambda(\alpha\|\beta\|_1 + (1-\alpha)\|\beta\|_2) \tag{18}$$

- The ℓ_1 norm encourages sparse solution, which does the job of model selection
- The ℓ_2 norm ensures that the optimization problem is strictly convex while encouraging grouping effect (highly correlated variables have similar regression coefficients)