Risk and Performance Estimators Standard Errors (RPESE)

Anthony Christidis, Doug Martin, Xin Chen

October 27, 2019

Abstract

The Risk and Performance Estimators Standard Errors package (`RPESE`) imlements a new method for computing accurate standard errors of risk and performance esimtators when returns are serially correlated as when they are uncorrelated. The new method makes use of the representation of a risk or performance estimators as a summation of a time series of influence-function (IF) transformed returns, and computes estimator standard errors using a sophisticated method of estimating the spectral density at frequency zero of the time series of IF-transformed returns. The initial release of `RPESE` allows users to compute accurate standard errors for six risk estimators, including the standard deviation, semi-standard deviation, value-at risk and expected shortfall, and eight performance estimators, including the Sharpe ratio, Sortino ratio, and expected shortfall ratio. This vignette provides basic instruction on how to use the `RPESE` package.

1 Introduction

The current finance industry practice in reporting risk and performance estimates for individual assets and portfolios seldom includes reporting estimate standard errors (SEs). For this reason, consumers of such reports have no way of knowning the statistical accuracy of the estimates. As a leading example, one seldom sees SE's reported for Sharpe ratios, and consequently cannot tell whether or not two Sharpe ratios for two different portfolio products are significantly different.

This motivated us to develop a Risk and Performance Estimator Standard Errors (`RPESE`) package for computing risk and performance estimator SE's that are accurate: (1) when

returns are serially correlated as well as when they are uncorrelated, and (2) account for fat-tailed and skewed non-normality of returns distributions. `RPESE` uses a new method for computing risk and performance estimators standard errors due to Chen and Martin [2018], henceforth (CM).

`RPESE` supports computing SEs for the six risk estimators shown in Table 1, and the eight performance estimators shown in Table 2. For each of the names in the Name column of the two tables, there is a corresponding R function with a slighly different name in `RPESE`, except for $LMP1$ and $LPM2$ in Table 1 there is a single function with an optional argument to choose between these two risk esimators, and for $SorR.\mu$ and $SorR.c$ in Table 2 there is a single function with an optional argument to choose between these two performance estimators.

Name	**Estimator Description**
SD	Sample standard deviation
SemiSD	Semi-standard deviation
LPM1	Lower partial moment of order 1
LPM2	Lower partial moment of order 2
ES	Expected shortfall with tail probability α
VaR	Value-at-risk with tail probability α

Table 1: Risk Estimator Names and Descriptions

Name	Estimator Description
Mean	Sample mean
SR	Sharpe ratio
SorR.μ	Sortino ratio with threshold the mean
SorR.c	Sortino ratio with threshold a constant c
ESratio	Mean excess return to ES ratio with tail probability α
VaRratio	Mean excess return to VaR ratio with tail probability α
RachRatio	Rachev ratio with lower upper tail probabilities α and β
OmegaRatio	Omega ratio with threshold c

Table 2: Performance Estimator Names and Descriptions

The `RPESE` package uses the `PerformanceAnalytics` (PA) package to compute the point estimators in Tables 1 and 2, and then computes the point estimate standard errors using the new method, which we now briefly describe.

The New Method

The basic elements of the new method are as follows. For a given risk or performance estimator, the time series of returns used to compute the estimate are transformed using the *influence function* (IF) of the estimator. For an introduction to influence functions, and derivations of the influence functions of the estimators in Tables 1 and 2, see **Zhang, Martin and Christidis (2019)**. It turns out that risk and performance estimators can be represented as the sample mean of the time series of influence-function transformed returns. It is well-known that the standardized (with respect to sample size) sum of a stationary time series has a variance that is well-approximated by the spectral density of the time series at zero frequency. Thus, computing the standard error of a risk or performance estimator reduces to estimating the spectral density at zero frequency of a standardized sum of the influence-function transformed returns. CM developed an effective method of doing so based on first computing the periodogram of the influence-function transformed returns, and then using a regularized general linear model (GLM) method for exponential and gamma distributions to fit a polynomial to the periodogram values. The regularization method used is an elastic net (EN) method that encourages sparsity of coefficients and is well-known in the machine learning literature. The intercept of such GLM fitting provides the an estimate of the spectral

density at zero frequency, and hnce a risk or performance estimator standard error. The interested reader can find the details in the CM paper.

2 RPESE Component Packages

The overall structure of `RPESE`, depicted in Figure 1, shows that `RPESE` makes use of the following two new packages:

- `RPEIF` (Influence Functions package)
- `RPEGLMEN` (Generalized Linear Model fitting with Elastic Net, for exponential and gamma distributions)

The purpose of `RPEIF` is to provide the analytic formula of the influence function, in order to compute the IF-transformed returns for each of the risk and performance estimators. For each risk and performance estimator in Tables 1 and 2, the `RPEGLMEN` package fits an EN regularized GLM polynomial fit to the periodogram of the time series of IF-transformed returns, using a GLM for exponential distributions or Gamma distributions. Figure 1 shows the relationship between the above two packages and the overall `RPESE` package.

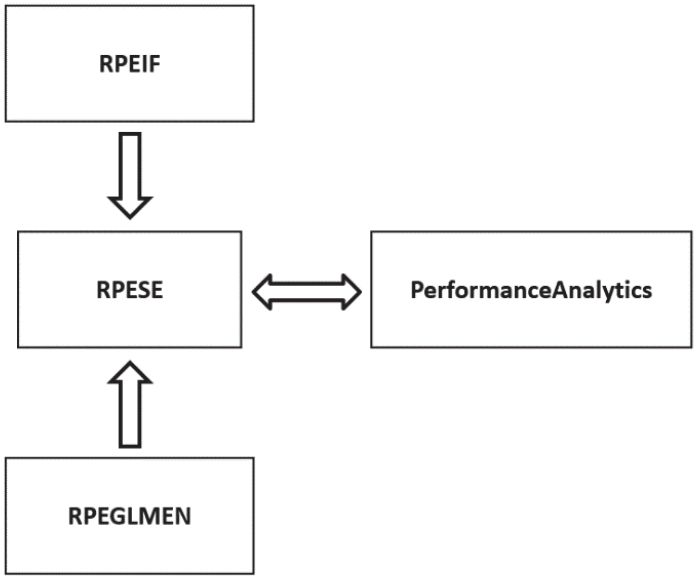


Figure 1: Packages Relations between `RPEIF`, `RPGLMEN`, `RPESE` and `PerformanceAnalytics`

3 How to Use RPESE

In the following sections, we show how to use `RPESE` to compute standard errors of risk and performance measures using the `edhec` hedge fund data.

3.1 Installing and Loading RPESE and Loading an Examples Data Set

In order to use `RPESE`, you don't need to manually install any other of the dependent packages as they will be installed automatically when `RPESE` is installed.

```
install.packages("RPESE")
```

To load `RPESE` and see the names of the functions in the package, use the code:

```
library(RPESE)
ls("package:RPESE")
```

```
##  [1] "ES.SE"           "ESratio.SE"      "EstimatorSE"
##  [4] "LPM.SE"          "Mean.SE"         "OmegaRatio.SE"
##  [7] "printSE"         "RachevRatio.SE"  "SemiSD.SE"
## [10] "SharpeRatio.SE"  "SortinoRatio.SE" "StdDev.SE"
## [13] "VaR.SE"          "VaRratio.SE"
```

We will use from the library `PerformanceAnalytics` the data set `edhec` to demonstrate the functionality of `RPESE`. The `PerformanceAnalytics xts` data set `edhec`, consisting of the returns of a number of hedge funds January, 1997 to August, 2008, is used in the examples in this vignette. Load `edhec`, confirm the object's class and see the names of the hedge funds with the following code.

```
data(edhec, package='PerformanceAnalytics')
class(edhec)
```

```
## [1] "xts" "zoo"
```

```
names(edhec)
```

```
##  [1] "Convertible Arbitrage"  "CTA Global"
##  [3] "Distressed Securities"  "Emerging Markets"
##  [5] "Equity Market Neutral"  "Event Driven"
##  [7] "Fixed Income Arbitrage" "Global Macro"
##  [9] "Long/Short Equity"      "Merger Arbitrage"
## [11] "Relative Value"         "Short Selling"
## [13] "Funds of Funds"
```

Since the hedge fund names are too long for convenient display, it will be best to use shorter two or three letter labels, with the following code:

```
names(edhec) <- c("CA", "CTA", "DIS", "EM","EMN", "ED", "FIA",
                  "GM", "LS","MA", "RV", "SS", "FOF")
```

3.2 Functions in RPESE

To see what functions are contained in the `RPESE` package, use the code line:

```
ls("package:RPESE")
```

```
##  [1] "ES.SE"           "ESratio.SE"      "EstimatorSE"
##  [4] "LPM.SE"          "Mean.SE"         "OmegaRatio.SE"
##  [7] "printSE"         "RachevRatio.SE"  "SemiSD.SE"
## [10] "SharpeRatio.SE"  "SortinoRatio.SE" "StdDev.SE"
## [13] "VaR.SE"          "VaRratio.SE"
```

3.3 Basic Functionality

To compute the standard errors of risk and performance measure estimators listed in Table 1, the argument `SE` must be set to `TRUE`. Also there is the option to return any of the standard error computation methods available in `RPESE`, namely `IFiid`, `IFcor`, `BOOTiid` and `BOOTcor`. See the documentation of the `RPESE` package for more details.

As a basic example, let's consider the Sharpe Ratio estimate and its computed standared error for the Convertible Arbitrage hedge fund. We print the output using the `printSE` function from `RPESE`.

```
# Sharpe Ratio SE computation for single hedge fund
Sharpe.out <- SharpeRatio.SE(edhec[,"CA"], p=0.95,
                             se.method=c("IFiid","IFcor","IFcorAdapt",
                                         "BOOTiid","BOOTcor", "none"))
# Print output
printSE(Sharpe.out)

##           SR      IFiid     IFcor IFcorAdapt   BOOTiid   BOOTcor
## CA 0.3196702 0.08084346 0.1312414 0.08674379 0.1179093 0.2036365
```

The risk and performance measures functions in allow the user to return the standard errors for more than one asset or portfolio at the same time.

```
# Sharpe Ratio SE computation for all hedge funds in data set
Sharpe.out <- SharpeRatio.SE(edhec, p=0.95,
                             se.method=c("IFiid","IFcor","IFcorAdapt",
                                         "BOOTiid","BOOTcor", "none"))
# Print output
printSE(Sharpe.out)

##             SR      IFiid      IFcor IFcorAdapt    BOOTiid    BOOTcor
## CA  0.31967021 0.08084346 0.13128238 0.08683742 0.13421480 0.19476942
## CTA 0.25822687 0.08084346 0.08509684 0.08509684 0.07885853 0.04223346
## DIS 0.43347113 0.08084346 0.11972432 0.08993596 0.11026873 0.18698761
## EM  0.21378651 0.08084346 0.10721319 0.10391561 0.10076870 0.10969582
## EMN 0.66652818 0.08084346 0.09909553 0.09909553 0.17576875 0.31532242
## ED  0.41537720 0.08084346 0.11172863 0.10118312 0.10992327 0.13113941
## FIA 0.29855572 0.08084346 0.12153432 0.09637284 0.14469677 0.22476763
## GM  0.45079543 0.08084346 0.08579923 0.08582137 0.07793082 0.05070621
## LS  0.34995636 0.08084346 0.10272975 0.10272975 0.09816289 0.12055368
## MA  0.60751282 0.08084346 0.10725244 0.10595569 0.13582965 0.12592479
## RV  0.50788010 0.08084346 0.11105067 0.09673653 0.14711330 0.18809460
## SS  0.07552172 0.08084346 0.09390980 0.09390980 0.07941756 0.06019738
## FOF 0.32497445 0.08084346 0.10605875 0.10221232 0.09393911 0.12085646
```

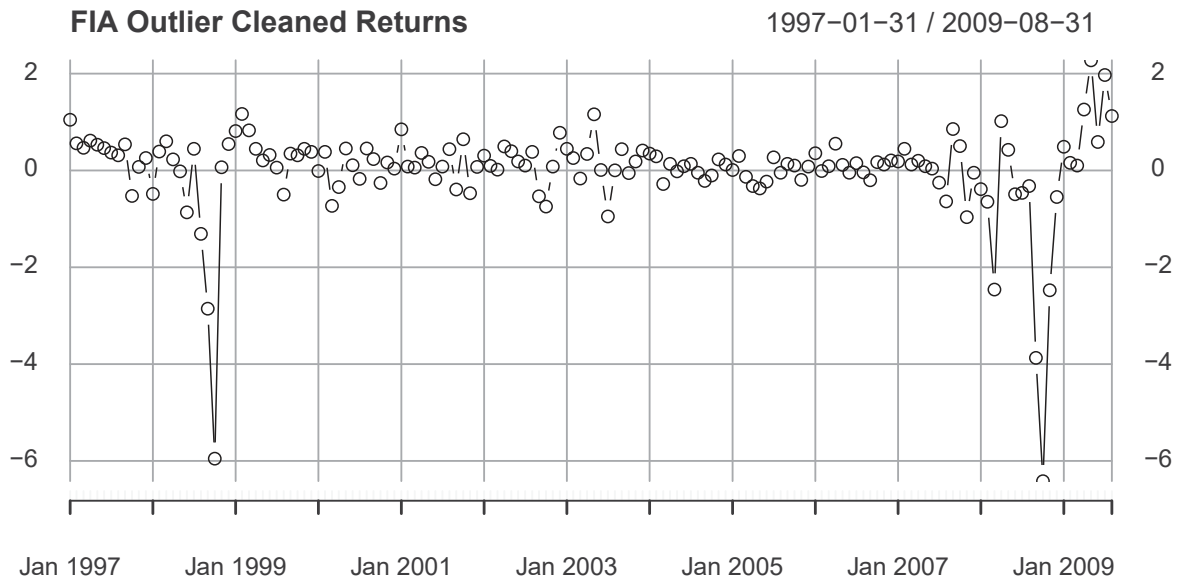
We may look at the documentation of the function for the Sharpe Ratio as well via the following code:

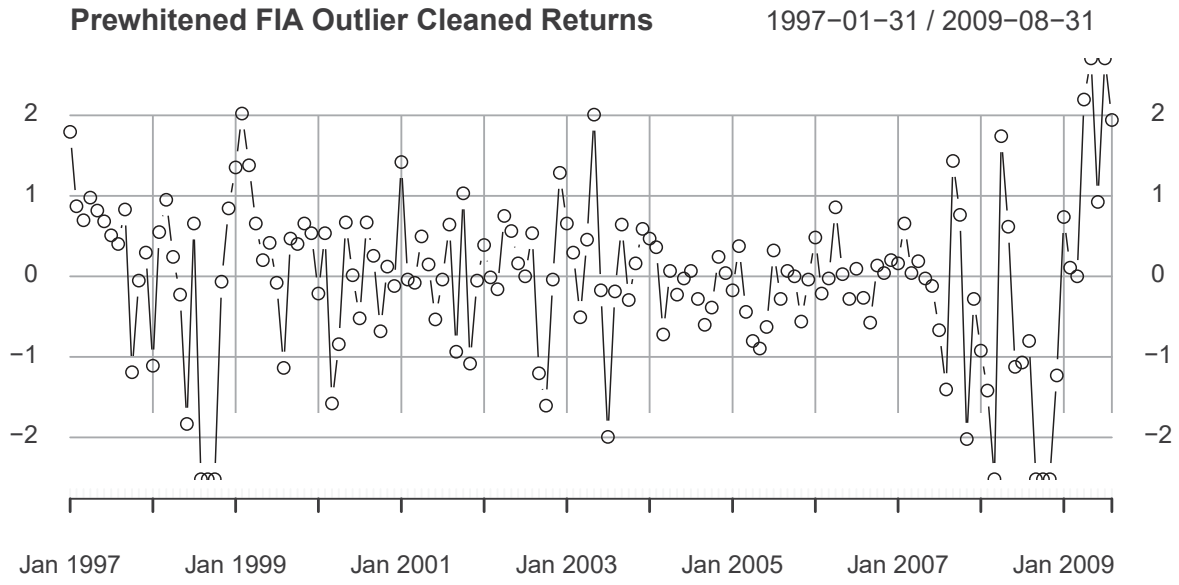
```
# Documentation of the ES.SE function
?SharpeRatio.SE
```

3.4 Outlier Cleaning

There is also the outlier cleaning capability from `RPEIF` that gets passed on to `RPESE`. First, notice the difference in the IF TS with and without outlier cleaning.

```
library(RPEIF)
IFout <- IF.SR(returns = edhec[,"FIA"], cleanOutliers = F, IFprint = T)
IFout.clean <- IF.SR(returns = edhec[,"FIA"], cleanOutliers = T, IFprint = T)
par(mfrow = c(2,1))
plot(IFout,type = "b", main = "FIA Outlier Cleaned Returns", lwd = .8)
plot(IFout.clean,type = "b", main = "Prewhitened FIA Outlier Cleaned Returns",
     lwd = .8)
```






```
par(mfrow = c(1,1))
```

We apply outlier cleaning to the computation of the Sharpe Ratio standard errors estimates.

```
# Sharpe Ratio SE computation for all hedge funds in data set
# Use outlier cleaning
clean.out <- SharpeRatio.SE(edhec, p=0.95,
                            se.method=c("IFiid","IFcor","IFcorAdapt",
                                        "BOOTiid","BOOTcor", "none"),
                            cleanOutliers = T)
# Print output
clean.comparison <- cbind(printSE(Sharpe.out)[,3:4], printSE(clean.out)[,3:4])
colnames(clean.comparison) <- c("IFcor", "IFcorAdapt",
                                "IFcor-Clean", "IFcorAdapt-Clean")
clean.comparison
```

```
##          IFcor IFcorAdapt IFcor-Clean IFcorAdapt-Clean
## CA  0.13128238 0.08683742  0.12568988       0.08205583
## CTA 0.08509684 0.08509684  0.08509684       0.08509215
## DIS 0.11972432 0.08993596  0.11972573       0.08988874
## EM  0.10721319 0.10391561  0.10622872       0.10301645
## EMN 0.09909553 0.09909553  0.10158696       0.10158696
## ED  0.11172863 0.10118312  0.11099375       0.10059428
## FIA 0.12153432 0.09637284  0.11437099       0.09408347
## GM  0.08579923 0.08582137  0.08572900       0.08572900
## LS  0.10272975 0.10272975  0.10249600       0.10249600
## MA  0.10725244 0.10595569  0.11062128       0.10910546
## RV  0.11105067 0.09673653  0.11142322       0.09352001
## SS  0.09390980 0.09390980  0.09483834       0.09450709
## FOF 0.10605875 0.10221232  0.10484908       0.10124688
```

3.5 Prewhitening

Another option in `RPEIF` is the availability to prewhiten the influence functions time series. This option is passed onto the `RPESE` package.

```
# Sharpe Ratio SE computation for all hedge funds in data set
# Use prewhitening
cleanPW.out <- SharpeRatio.SE(edhec, p=0.95,
                            se.method=c("IFiid","IFcor","IFcorAdapt",
```

```
                                          "BOOTiid","BOOTcor", "none"),
                            cleanOutliers = T,
                                                        prewhiten = T)
# Print output
cleanPW.comparison <- cbind(printSE(clean.out)[,3:4],
                            printSE(cleanPW.out)[,3:4])
colnames(cleanPW.comparison) <- c("IFcor", "IFcorAdapt",
                                  "IFcor-CleanPW", "IFcorAdapt-Clean")
cleanPW.comparison
```

```
##          IFcor IFcorAdapt IFcor-CleanPW IFcorAdapt-Clean
## CA  0.12568988 0.08205583    0.06857791       0.08205583
## CTA 0.08509684 0.08509215    0.08072240       0.08509684
## DIS 0.11972573 0.08988874    0.06878123       0.08992396
## EM  0.10622872 0.10301645    0.07513254       0.10266629
## EMN 0.10158696 0.10158696    0.07384398       0.10158548
## ED  0.11099375 0.10059428    0.07276966       0.10059428
## FIA 0.11437099 0.09408347    0.07546486       0.08012414
## GM  0.08572900 0.08572900    0.08062512       0.08571254
## LS  0.10249600 0.10249600    0.07636046       0.10249423
## MA  0.11062128 0.10910546    0.07568619       0.10910546
## RV  0.11142322 0.09352001    0.07267214       0.09397071
## SS  0.09483834 0.09450709    0.08017577       0.09483834
## FOF 0.10484908 0.10124688    0.07476891       0.10119772
```

If `se.method` is set to `IFcorAdapt`, then the `prewhiten` option has no effect as the standard error estimate is a weighted average between `IFcor` with and without prewhitening. See Section 4.2 in Chen and Martin [2018] for more details.

3.6 Exponential and Gamma Distributions

The `RPEGLMEN` package used in the fitting of the penalized GLM model to the periodogram of the influence functions time series allows for two different distributions: the exponential distribution (default) and the Gamma distribution.

```r
# Sharpe Ratio SE computation for all hedge funds in data set
# Use Gamma distribution
Gamma.out <- SharpeRatio.SE(edhec[,], p=0.95,
                            se.method=c("IFiid","IFcor","IFcorAdapt",
                                        "BOOTiid","BOOTcor", "none"),
                            cleanOutliers = T, fitting.method = "Gamma")
# Print output
Gamma.comparison <- cbind(printSE(clean.out)[,3:4],
                          printSE(Gamma.out)[,3:4])
colnames(Gamma.comparison) <- c("IFcor", "IFcorAdapt",
                                "IFcor-Gamma", "IFcorAdapt-Gamma")
Gamma.comparison
```

```
##          IFcor IFcorAdapt IFcor-Gamma IFcorAdapt-Gamma
## CA  0.12568988 0.08205583  0.16785164       0.08866322
## CTA 0.08509684 0.08509215  0.07626128       0.06595057
## DIS 0.11972573 0.08988874  0.16992492       0.11348607
## EM  0.10622872 0.10301645  0.13537639       0.09661254
## EMN 0.10158696 0.10158696  0.14007690       0.12603020
## ED  0.11099375 0.10059428  0.14951598       0.13093543
## FIA 0.11437099 0.09408347  0.13605974       0.10418297
## GM  0.08572900 0.08572900  0.08250677       0.08685945
## LS  0.10249600 0.10249600  0.11413836       0.11164911
## MA  0.11062128 0.10910546  0.13533226       0.13291935
## RV  0.11142322 0.09352001  0.13492591       0.11926165
## SS  0.09483834 0.09450709  0.09409264       0.09130823
## FOF 0.10484908 0.10124688  0.13960894       0.13329769
```

References

X. Chen and R. D. Martin. Standard errors of risk and performance measure estimators for serially correlated returns. 2018. URL https://ssrn.com/abstract=3085672.