RPESE: Risk and Performance Estimators Standard Errors with Serially Dependent Data

by Anthony-Alexander Christidis and R. Douglas Martin

Abstract The R package **RPESE** (Risk and Performance Estimators Standard Errors) implements a new method for computing accurate standard errors of risk and performance estimators when returns are serially dependent. The new method makes use of the representation of a risk or performance estimator 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. Two additional packages used by **RPESE** are introduced, namely **RPEIF** which computes and provides graphical displays of the IF of risk and performance estimators, and **RPEGLMEN** which implements a regularized Gamma generalized linear model polynomial fit to the periodogram of the time series of the IF-transformed returns. A Monte Carlo study shows that the new method provides more accurate estimates of standard errors for risk and performance estimators compared to well-known alternative methods in the presence of serial correlation.

Introduction

In the risk and portfolio management industry, risk and performance estimators are standard tools used in data-driven decision-making processes. The current industry practice in reporting risk and performance estimates for individual assets and portfolios typically do not include their standard error (SE) estimates. For this reason, consumers of such reports have no way of knowing the statistical accuracy of the estimates. As a leading example, one seldom sees SEs reported for Sharpe ratios, and consequently cannot tell whether or not two Sharpe ratios for two different portfolios or assets are significantly different. This motivated the development of the Risk and Performance Estimator Standard Errors package, **RPESE**, for computing risk and performance estimator standard errors that are accurate when returns that are serially correlated and can have fat-tailed and skewed non-normality of returns distributions. **RPESE** uses a new method for computing risk and performance estimators standard errors developed by Chen and Martin (2021).

In the quantitative finance literature, numerous statistical methods have been proposed to evaluate the variability of risk and performance estimators under the assumption of independent and identically distributed (i.i.d.) returns. These methods are however inadequate when returns are serially correlated. An example of this was provided by Lo (2002), who showed that the Sharpe ratio estimator standard error based on the assumption of i.i.d. returns is overly optimistic when returns are actually serially correlated. In terms of statistical methods to compute standard errors of estimators when working with time series data exhibiting serial correlation, there are two well-known methods in the literature: the nonparametric heteroskedasticity and autocorrelation consistent (HAC) covariance method (Newey and West, 1987; Andrews, 1991; Zeileis, 2004) and the nonparametric block bootstrap method (Kunsch, 1989; Politis and Romano, 1994). HAC covariance estimators, which rely on a kernel-based approach on the lag-k covariance estimates of the time series, possess desirable statistical properties such as consistency under heteroscedasticity and autocorrelation. Block bootstrap methods have been shown to provide accurate standard errors if they are well calibrated (Ledoit and Wolf, 2008). However, they rely on randomness which is an unappealing feature for industry practitioners as different results can be reported on the same data set, albeit small if a large enough number of replicates are used.

Unlike the nonparametric HAC and block bootstrap methods, the method implemented in the **RPESE** package is a fully parametric and deterministic method for computing risk and performance estimators standard errors for time series with serial correlation, which is an appealing feature for the purpose of constructing confidence intervals and tests. While IFs are commonly used tools in robust statistics, they are seldomly used in quantitative finance research and applications. The new method involves the representation of the risk and performance estimators using influence functions (IFs) and fitting a penalized Gamma generalized linear model (GLM) to the periodogram of the IF-transformed returns time series. The estimate of the variance of the estimator can be obtained from the spectral density estimate of the IF-transformed returns time series at zero frequency, a technique introduced by Heidelberger and Welch (1981).

The remainder of this article is organized as follows. The "New Methodology: Standard Errors via Influence Functions" section provides the theoretical and computational details of the new method to compute standard errors of risk and performance estimators. The "Influence Functions for Risk and

Performance Estimators" section presents five risk estimators and seven performance estimators with their influence functions, and introduces the **RPEIF** package with some sample code for the purpose of computing and providing graphical displays of such influence functions. The "Periodogram Based Generalized Linear Model Method" section describes the computational details of the Gamma GLM fit to the periodogram of the IF-transformed returns time series and briefly discusses the implementation in the **RPEGLMEN** package. The "Application: Hedge Funds Data Standard Errors with **RPESE**" section provides example code to compute standard errors for hedge funds returns data. The "Monte Carlo Study with IF-Based Standard Errors" section presents a benchmark comparison of the new method with two HAC based approaches for serially correlated data. The "Summary" section discusses future developments for the new method and the **RPESE** package.

New Methodology: Standard Errors via Influence Functions

IFs were first introduced by Hampel (1974), and were developed further by Hampel et al. (1986). In this section, we provide the definition and basic properties of influence functions, with a view toward their use for understanding the influence of outliers on risk and performance estimators, and for computing standard errors of such estimators for both uncorrelated and serially correlated returns.

Risk and Performance Estimator Functional Representations

The large-sample value (as sample size n tends to infinity) of a risk or performance estimator may be represented as a functional $T = T(F)$ of the marginal distribution function F of a time series $r_1, r_2, \ldots, r_n$ of returns.[1] For example the functional for the mean (expected value) is

$$\mu(F) = \int r dF(r) \tag{1}$$

and the functional for the standard deviation (returns volatility) is

$$\sigma(F) = \left[\int (r - \mu(F))^2 dF(r) \right]^{\frac{1}{2}}. \tag{2}$$

Given a functional representation $T(F)$ of an estimator, a finite-sample *non-parametric* estimator T_n is easily obtained by replacing the unknown distribution F by the empirical distribution F_n that has a jump of height $1/n$ at each of the observed returns values $r_1, r_2, \ldots, r_n$:

$$T_n = T(F_n) = T(r_1, r_2, \ldots, r_n). \tag{3}$$

For example, the finite-sample non-parametric estimators of the mean and standard deviation are the sample mean and sample standard deviation, respectively:

$$\hat{\mu}_n = \frac{1}{n} \sum_{t=1}^{n} r_t \qquad \hat{\sigma}_n = \left[\frac{1}{n} \sum_{t=1}^{n} (r_t - \hat{\mu}_n) \right]^{\frac{1}{2}}. \tag{4}$$

We note that one can also derive parametric estimators from parametric functional representation obtained by replacing F by F_θ, where θ is the parameter vector for a parametric distribution function. In this case one obtains the finite-sample estimator by replacing the unknown parameter by its estimator, typically the maximum-likelihood estimator (MLE). See for example, Martin and Zhang (2019) for a treatment of parametric and non-parametric expected shortfall (ES) estimators for normal and t-distributions. However, the current version of the **RPEIF** package only deals with non-parametric risk and performance estimators.

Estimator Influence Function Definition

IFs are based on the use of the following mixture distribution perturbation of a fixed target distribution $F(x)$:

$$F_\gamma(x) = (1 - \gamma)F(x) + \gamma \delta_r(x),\ 0 \leq \gamma < 1/2 \tag{5}$$

[1] The term functional refers to a function whose domain is an infinite dimensional space, e.g., the space of distribution functions.

where $\delta_r(x)$ is a point mass discrete distribution function with a jump of height one located at value r . The IF of an estimator with functional form $T(F)$ is defined as:

$$IF(r;T,F) = \lim_{\gamma \to 0} \frac{T(F_\gamma) - T(F)}{\gamma} = \frac{d}{d\gamma} T(F_\gamma)|_{\gamma=0} \tag{6}$$

The IF is a special directional derivative (i.e., a Gateaux derivative) of the functional $T(F)$ in the direction of a point mass distributions δ_r, evaluated at F. It is straightforward, and more or less tedious, to derive formulas for the IFs of risk and performance estimators. For example, the IF of the sample mean is:

$$IF(r;\mu;F) = r - \mu \tag{7}$$

where $\mu = \mu(F)$ depends on the underlying returns marginal distribution F. The above influence function has the property that its expected value is zero, which is a reflection of the general property than an influence function has zero expected value (Hampel, 1974):

$$E\left[IF(r;T,F)\right] = 0. \tag{8}$$

A Key Influence Function Property

A key IF property is that for well behaved estimator functionals, the finite-sample estimator $T_n = T(F_n) = T(r_1, r_2, \ldots, r_n)$ can be expressed in terms of the sample mean of IF transformed returns as

$$T_n - T(F) = \frac{1}{n} \sum_{t=1}^{n} IF(r_t;T,F) + remainder \tag{9}$$

where the remainder goes to zero in a probabilistic sense as $n \to \infty$. Thus the finite sample variance of T_n is approximately given by

$$Var\left(T_n\right) = Var\left[\frac{1}{n} \sum_{t=1}^{n} IF(r_t;T,F)\right] \tag{10}$$

and in the special case where the returns r_t are i.i.d., the IF-transformed returns are i.i.d., and the variance of T_n reduces to

$$Var\left(T_n\right) = \frac{1}{n} E[IF^2(r_1;T,F)]. \tag{11}$$

and the expectation on the right-hand side can be evaluated empirically as the sample mean of the squared influence functions.

However, when the $r_t, t = 1, 2, \ldots, n$ are serially dependent, such as in the case of serially correlated AR(1) returns or serially uncorrelated but dependent GARCH(1,1) returns, the IF-transformed returns time series $IF(r_t;T,F)$ will generally have serial correlation that needs to be accounted for in calculating the variance on the the right-hand-side of (10). In particular,

$$\frac{1}{n} Var\left[\sum_{i=1}^{n} IF(r_t;T,F)\right] = C_{IF}(0) + 2\sum_{k=1}^{n-1}\left(1 - \frac{k}{n}\right) C_{IF}(k), \tag{12}$$

where $C_{IF}(k)$ is the symmetric lag-k covariance between $IF(r_t)$ and $IF(r_{t+k})$. Spectral analysis theory, extensively used in science and engineering, shows that the variance of the sum of the values of a serially correlated stationary time series is given by the spectral density of the time series at zero frequency. Thus the problem of estimating the variance (10), with possibly serially correlated returns, reduces to the problem of estimating the spectral density at zero frequency of the the time series $IF(r_t;T,F)/n$. Chen and Martin (2021) show how to do this by a polynomial Gamma GLM fitting method, with elastic net (EN) regularization, that works well when the returns are serially correlated, as well as when they are uncorrelated. Their methodology is implemented in the **RPESE** package, which in turn makes fundamental use of the **RPEIF** package to compute and provide graphical displays for the influence functions of popular risk and performance estimators, and the **RPEGLMEN** to fit a polynomial Gamma GLM with EN penalty to the IF-transformed returns time series periodogram.

Influence Functions for Risk and Performance Estimators

The current version (1.2.2) of the **RPESE** package supports six risk and nine performance estimators. The risk estimators are the sample standard deviation (SD), the semi-standard deviation (SemiSD), the lower partial moment of order one and two (LPM1 and LPM2), the expected shortfall (ES) and the

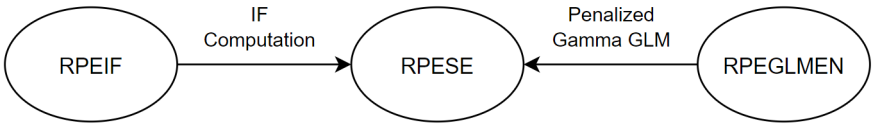

Figure 1: Package relations between **RPEIF**, **RPESE** and **RPEGLMEN**.

value-at-risk (VaR), the latter two risk estimators with tail probability α. The performance estimators are the mean (Mean), a robust M-estimator of the mean (robMean) of ψ-type[2], the Sharpe ratio (SR), the downside Sharpe ratio (DSR), the Sortino ratio (SoR) with constant threshold c, the expected shortfall ratio (ESratio) and the value-at-risk ratio (VaRratio) with tail probability α, the Rachev ratio (RachevRatio) with lower and upper tail probability α and β, and the Omega ratio (OmegaRatio) with constant threshold c.

The formulas for these risk and performance estimators and their functional representations, and the derivations of their influence function formulas, are given in Zhang et al. (2021). In Table 1, the names of the functions in the **RPEIF** and **RPESE** packages for the risk and performance estimators are provided.

Risk	**RPEIF**	**RPESE**
SD	`IF.SD`	`SD.SE`
SemiSD	`IF.SemiSD`	`SemiSD.SE`
LPM1 or LPM2	`IF.LPM`	`LPM.SE`
ES	`IF.ES`	`ES.SE`
VaR	`IF.VaR`	`VaR.SE`
Performance	**RPEIF**	**RPESE**
Mean	`IF.Mean`	`Mean.SE`
robMean	`IF.robMean`	`robMean.SE`
SR	`IF.SR`	`SR.SE`
DSR	`IF.DSR`	`DSR.SE`
SoR	`IF.SoR`	`SoR.SE`
ESratio	`IF.ESratio`	`Esratio.SE`
VaRratio	`IF.VaRratio`	`VaRratio.SE`
RachevRatio	`IF.RachevRatio`	`RachevRatio.SE`
OmegaRatio	`IF.OmegaRatio`	`OmegaRatio.SE`

Table 1: Functions in **RPEIF** and **RPESE** for the risk and performance estimators.

The RPEIF Package

The functions in the **RPEIF** package listed in Table 1 are used for two distinct purposes. The first is to evaluate an estimator IF at a set of data values, and plot them to display a graph of the influence function. This allows the user to explore the different shapes of the IFs of different estimators. The second and primary purpose of these IF functions is to compute IF-transformed time series of returns, as a first step in the overall method of computing standard errors for risk and performance estimators.

To briefly demonstrate the usage of the **RPEIF** package, the `edhec` data set available in the **PerformanceAnalytics** package will be used. This data set contains hedge fund returns from January, 1997 to November, 2019. The data can be loaded as an `xts` time series object with the following R code.

```
install.packages("PerformanceAnalytics")
data(edhec, package = "PerformanceAnalytics")
class(edhec)

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

The hedge fund names in `edhec` are too long to display well in plots. The following code replaces those long names with shorter names.

[2]Commonly referred to as a robust M-estimator of "location" in the literature.

```
colnames(edhec) <- c("CA", "CTAG", "DIS", "EM", "EMN", "ED", "FIA", "GM", "LS", "MA",
                     "RV", "SS", "FoF")
```

The following functions are available in **RPEIF**:

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

```
 [1] "IF"             "IF.DSR"         "IF.ES"          "IF.ESratio"
 [5] "IF.LPM"         "IF.Mean"        "IF.OmegaRatio"  "IF.RachevRatio"
 [9] "IF.robMean"     "IF.SD"          "IF.SemiSD"      "IF.SoR"
[13] "IF.SR"          "IF.VaR"         "IF.VaRratio"    "nuisParsFn"
```

In order to compute the values and plot the shapes of influence functions using the IF functions in Table 1, nuisance parameters need to be specified, and there are two basic methods of doing so. Using the `nuisParsFn` function, nuisance parameters can be generated by specifying "typical" values based on some assumed returns distribution. For the risk measure estimators and performance measure estimators, the `nuisParsFn` function assumes by default that the returns follow the normal distribution, with monthly mean return of $\mu = 1\%$, risk-free rate $rf = 0\%$, monthly volatility of $\sigma = 5\%$ (the corresponding annual mean and volatility are 12% and 17.3%, respectively), and in addition assumes by default that $c = 0$ for LPM and SoR, $\alpha = 0.10$ for VaR and ES, and in addition $\beta = 0.10$ for the Rachev ratio. Thus:

```
args(nuisParsFn)
```

```
## function (mu = 0.01, sd = 0.05, c = 0, alpha = 0.1, beta = 0.1)
```

To generate nuisance parameters by using a mean return of 2% instead of 1% and a volatility of 15% instead of 5% (the defaults) for the purpose of displaying the IF plots for the SD and the SR, the `nuisPars` argument can be used as in the following code, with the plots shown in Figure 2.

```
par(mfrow = c(2, 1))
outSD <- IF.SD(evalShape = T, IFplot = T, nuisPars = nuisParsFn(mu = 0.02, sd = 0.15))
outSR <- IF.SR(evalShape = T, IFplot = T, nuisPars = nuisParsFn(mu = 0.02, sd = 0.15))
```

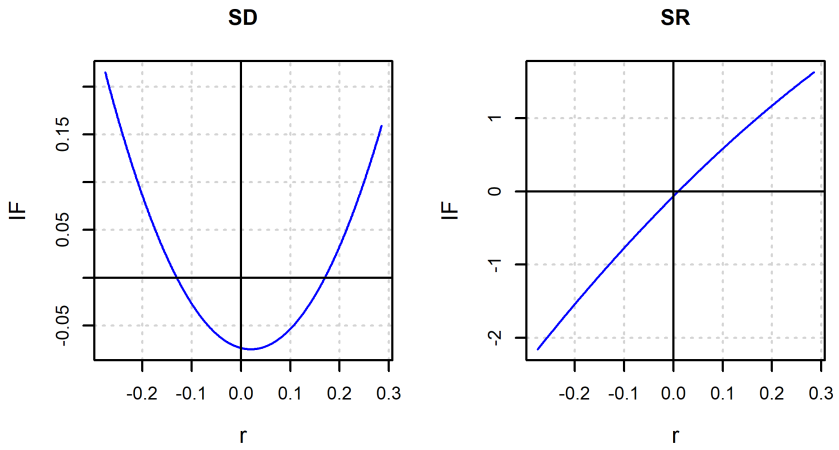

Figure 2: IF shapes of SD and SR with user-specified nuisance parameters.

The second way to specify nuisance parameter values is to estimate them from a returns time series of interest, the latter of which is specified by using the argument `returns` of the IF functions. For example, the Convertible Arbitrage (CA) hedge fund time series can be used for this purpose by using the following code, the results of which are shown in Figure 3.

```
par(mfrow = c(2, 1))
outSD <- IF.SD(returns = edhec$CA, evalShape = T, IFplot = T)
outSR <- IF.SR(returns = edhec$CA, evalShape = T, IFplot = T)
```

To plot the IF-transformed returns of the SD and SR estimators, use the following code to generate the output shown in Figure 4:

```
SDiftr <- IF.SD(returns = edhec$CA)
SRiftr <- IF.SR(returns = edhec$CA)
```

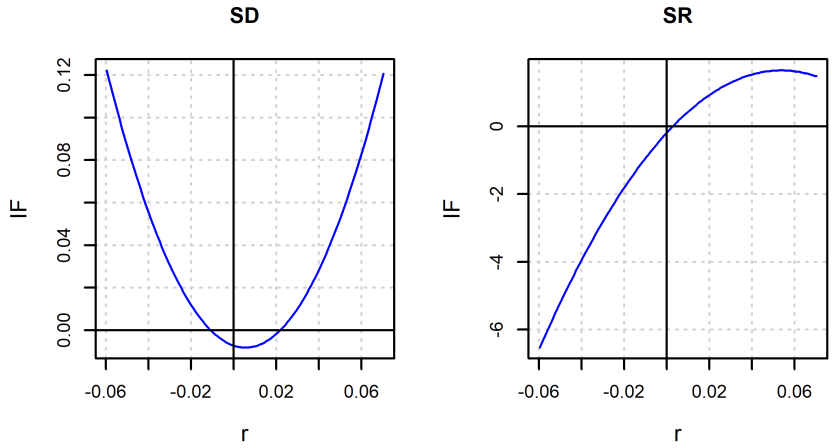

Figure 3: IF shapes of SD and SR with estimated nuisance parameters.

```
par(mfrow = c(3, 1))
plot(edhec$CA, lwd = 0.8, ylab = "Returns", main = "CA Hedge Fund Returns")
plot(SDiftr, lwd = 0.8, main = "IF.SD Transformed Returns")
plot(SRiftr, lwd = 0.8, main = "IF.SR Transformed Returns")
```

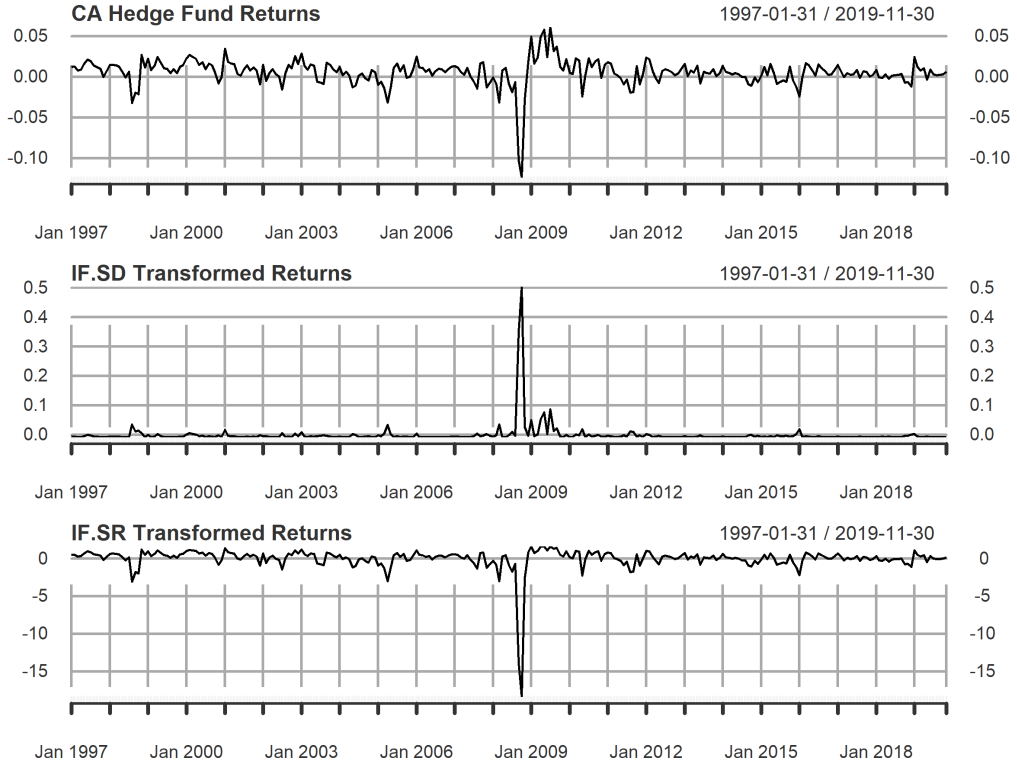

Figure 4: SD and SR IF-transformed returns for CA hedge fund.

Returns data are prone to outliers which adversely influence parameter estimates and inflate the standard errors of risk and performance estimators. A very reliable outlier cleaning method that shrinks outliers can be obtained based on a robust location M-estimator and an associated robust scale estimator $\hat{s}$. A location M-estimator is computed as a solution of the equation

$$\sum_{t=1}^{n} \psi_{mOpt}\left(\frac{r_t - \hat{\mu}_M}{\hat{s}}\right) = 0 \tag{13}$$

where $\psi_{mOpt} = \psi_{mOpt}(x)$ is an optimal bias robust "psi" function, and $\hat{s}$ is a robust scale estimate of the residuals $\epsilon_t = r_t - \hat{\mu}_M$. For an introduction to location M-estimators and their computation, see

Sections 2.3 and 2.7 of Maronna et al. (2019). The optimal bias robust function ψ_{mOpt} is the default used by the function `locScaleM` in the **RobStatTM** package. The `robMean` function in **RPESE** computes the estimates $\hat{\mu}_M$ and $\hat{s}$ using `locScaleM`.

Based on a location estimate $\hat{\mu}_M$ and associated scale estimate $\hat{s}$, it is natural to define returns r_t that fall outside the interval $[\hat{\mu}_M - 3 \cdot \hat{s}, \hat{\mu}_M + 3 \cdot \hat{s}]$ as outliers for a 95% efficiency. Such outliers are then "cleaned" by shrinking them to the nearest boundary of that interval. For the Fixed Income Arbitrage (FIA) hedge fund returns, using the optimal bias robust ψ_{mOpt} function, it turns out that the above interval is $[-0.01171, 0.02231]$. Correspondingly, the FIA hedge fund returns with values less than -0.01171, or greater than 0.02231, are detected as outliers and shrunk accordingly.

The following code results in Figure 5, which shows the sample mean IF-transformed FIA returns (which are equal to FIA returns minus the very small mean of the FIA returns) in the top plot, and the outlier cleaned IF tranformaed FIA returns in the bottom plot.

```
iftrFIA <- IF.Mean(returns = edhec$FIA)
iftrFIAclean <- IF.Mean(returns = edhec$FIA, cleanOutliers = T)
par(mfrow = c(2, 1))
plot(iftrFIA, main = "FIA IF-transformed Returns", lwd = 0.8)
plot(iftrFIAclean, main = "Outlier Cleaned FIA IF-transformed Returns", lwd = 0.8)
```

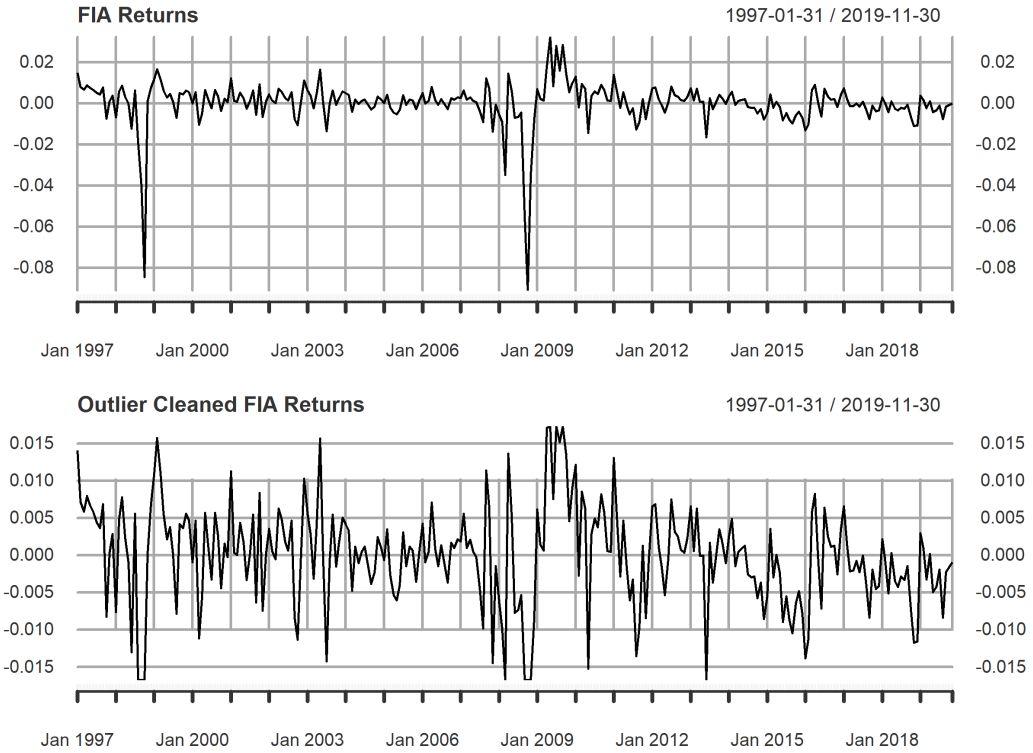

Figure 5: IF and outlier cleaned IF-transformed FIA returns.

Spectral density function estimation is a frequently used method in the field of signal processing, and in other engineering and science applications. Prewhitening is a technique often used to improve the performance of spectral density estimators. Since the core of the method described in Chen and Martin (2021) is estimation of a spectral density at frequency zero of an IF-transformed returns time series IF_t, it is natural to be able to use prewhitening of that time series. The prewhitened variant of the basic IF-based SE method in the **RPESE** package implement such prewhitening. A prewhitened version IF_t^{pw} of the IF_t time series by using the prewhitening transformation

$$IF_t^{pw} = IF_t - \hat{\rho} IF_{t-1} \tag{14}$$

where $\hat{\rho}$ is a lag-one serial correlation coefficient estimat of the IF_t. In general the IF_t^{pw} series is not a serially uncorrelated (white noise) series, but it has considerably less serial correlation than IF_t, and a periodogram estimator based on IF_t^{pw} will suffer from relatively little bias compared with one based on IF_t. Since outliers can have adverse influence not only on risk and performance estimators, but also on the estimator $\hat{\rho}$ used for prewhitening, it is always a good idea to clean outliers before prewhitening. This can done for example using the following code, which results in the plot shown in Figure 6.

```
iftrFIAcl <- IF.Mean(returns = edhec$FIA, cleanOutliers = T)
PWiftrFIAcl <- IF.Mean(returns = edhec$FIA, cleanOutliers = T, prewhiten = T)
par(mfrow = c(2, 1))
plot(iftrFIAcl, main = "FIA Outlier Cleaned Returns", lwd = 0.8)
plot(PWiftrFIAcl, main = "Prewhitened FIA Outlier Cleaned Returns", lwd = 0.8)
```

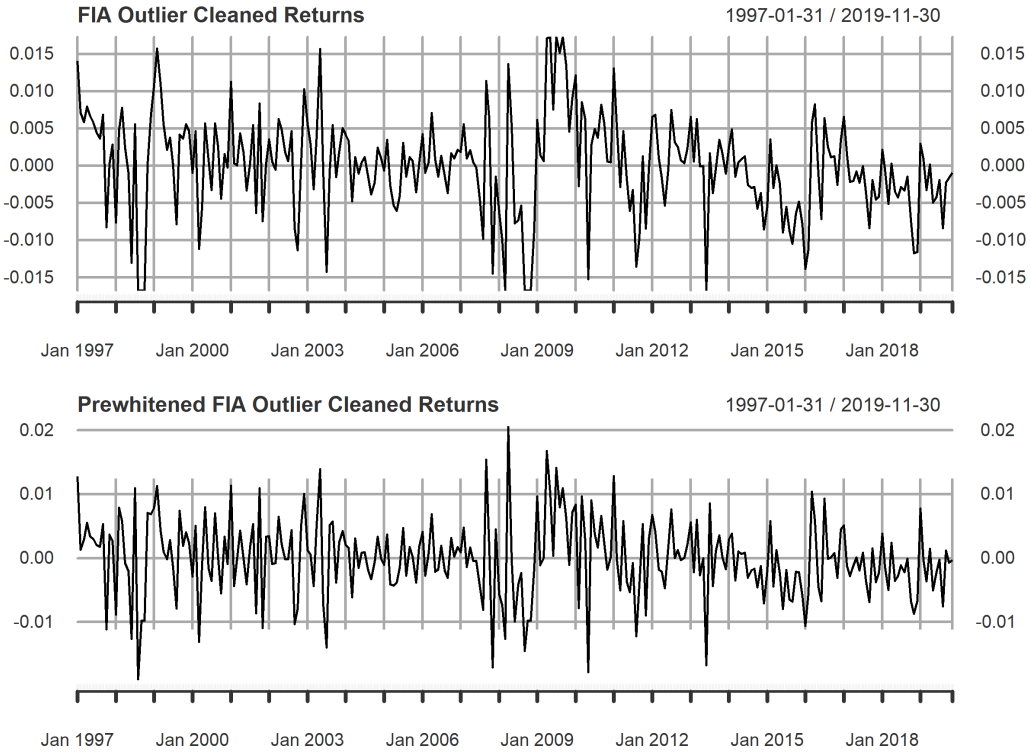

Figure 6: Outlier cleaned IF-transformed FIA returns and its prewhitened version.

For more information on all the features and visualization tools in the **RPEIF** package, see https://CRAN.R-project.org/package=RPEIF, where a reference manual and vignette are provided.

Periodogram Based GLM Method

The problem of estimating the variance (10) of a risk or performance estimator in the presence of serially correlated returns reduces to estimating the spectral density at zero frequency of the time series $IF(r_t; T, F)/n$. To do so, Chen and Martin (2021) proposed to fit a polynomial Gamma GLM method with elastic net (EN) regularization to the periodogram of the time series $IF(r_t; T, F)/n$. Because of the computationally intensive nature of the cross-validation procedure for the tuning parameter λ and the fact that the implementation must be efficient enough to handle multiple assets and portfolio returns, Chen et al. (2018) developed an accelerated proximal gradient descent algorithm for Gamma GLM with EN regularization.

The RPEGLMEN Package

The accelerated proximal gradient descent algorithm by Chen et al. (2018) was implemented in the **RPEGLMEN** package. Multicore processing is available in the penalized Gamma GLM functions within the package. The EN-penalized Exponential GLM is available as a special case of the Gamma GLM. For code examples and more details on the parallelization features in **RPEGLMEN**, see https://CRAN.R-project.org/package=RPEGLMEN, where a reference manual and vignette are provided.

Application: Hedge Funds Data Standard Errors with RPESE

The following functions are available in **RPESE** are:

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

```
 [1] "DSR.SE"         "ES.SE"          "ESratio.SE"     "EstimatorSE"
 [5] "LPM.SE"         "Mean.SE"        "OmegaRatio.SE"  "printSE"
 [9] "RachevRatio.SE" "robMean.SE"     "SD.SE"          "SemiSD.SE"
[13] "SoR.SE"          "SR.SE"          "VaR.SE"         "VaRratio.SE"
```

The only argument that is required for the standard error computing functions in **RPESE** from Table 1 is the `data` argument, and if only the data argument is supplied then the function uses the defaults of the other arguments. The arguments for the `SD.SE` and `SR.SE` functions given below show that the default GLM distribution is the exponential distribution for both the SD risk estimator and the SR performance estimator, but the `se.methods` defaults are the pair of `IFiid` and `IFcor` for the SD estimator and the pair `IFiid` and `IFcorPW` for the SR estimator.

```
args(SD.SE)
```

```
## function (data, se.method = c("IFiid", "IFcor", "IFcorAdapt",
##     "IFcorPW", "BOOTiid", "BOOTcor")[1:2], cleanOutliers = FALSE,
##     fitting.method = c("Exponential", "Gamma")[1], d.GLM.EN = 5,
##     freq.include = c("All", "Decimate", "Truncate")[1], freq.par = 0.5,
##     corOut = c("none", "retCor", "retIFCor", "retIFCorPW")[1],
##     return.coef = FALSE, ...)
```

```
args(SR.SE)
```

```
## function (data, rf = 0, se.method = c("IFiid", "IFcor", "IFcorAdapt",
##     "IFcorPW", "BOOTiid", "BOOTcor")[c(1, 4)], cleanOutliers = FALSE,
##     fitting.method = c("Exponential", "Gamma")[1], d.GLM.EN = 5,
##     freq.include = c("All", "Decimate", "Truncate")[1], freq.par = 0.5,
##     corOut = c("none", "retCor", "retIFCor", "retIFCorPW")[1],
##     return.coef = FALSE, ...)
```

The standard error of the SD for the hedge funds in the `edhec` package can be computed, using the default arguments, with the following code:

```
SRout <- SR.SE(edhec)
```

The result returned by the function is a list. A more compact display of the results, with rounding to three digits by default, can be obtained using the `printSE` function, whose arguments are

```
args(printSE)
```

```
## function (SE.data, round.digit = 3, round.out = TRUE)
```

with the resulting output with this function for the SR is given below.

```
printSE(SRout)
```

```
##          SR IFiid IFcor IFcorPW
## CA    0.338 0.096 0.117   0.203
## CTAG  0.180 0.060 0.057   0.057
## DIS   0.392 0.080 0.108   0.134
## EM    0.194 0.069 0.084   0.092
## EMN   0.543 0.110 0.116   0.124
## ED    0.372 0.079 0.100   0.111
## FIA   0.377 0.113 0.134   0.183
## GM    0.371 0.054 0.057   0.057
## LS    0.315 0.066 0.079   0.085
## MA    0.560 0.092 0.098   0.104
## RV    0.504 0.097 0.117   0.169
## SS   -0.041 0.061 0.072   0.072
## FoF   0.275 0.066 0.083   0.093
```

Standard Errors Methods

The `se.method` argument is particularly important, and the standard error computation methods for the various choices for this argument are as follow.

- "IFiid": This results in an influence function (IF) method based computation of a standard error assuming i.i.d. returns.
- "IFcor": This is the basic IF method computation of a standard error that takes into account serial dependence in the returns when the correlation is not too large. This is often the best method for risk estimators.
- "IFcorAdapt": This IF based method adaptively interpolates between IFcor and IFcorPW which is a good approach when the user is uncertain of the degree of serial dependence in the returns.
- "IFcorPW": This IF based method uses pre-whitening of the IF-transformed returns and is useful when serial correlation is large.
- "BOOTiid": This choice results in computing a bootstrap standard error assuming i.i.d. returns.
- "BOOTcor": This choice uses a block bootstrap method to compute a standard error that takes into account serial dependence of returns.

The two default choices of methods are:

- "IFiid" and "IFcor" for risk estimators, and for performance estimators when returns serial correlation are known to be small, and
- "IFiid" and "IFcorPW" for performance estimators when returns correlations are unknown and may be large.

The following code results in standard errors for all thirteen of the edhec hedge funds, using five different methods methods.

```
SRout <- SR.SE(edhec, se.method = c("IFiid","BOOTiid","IFcor","IFcorPW","BOOTcor"))
printSE(SRout)
```

```
##          SR IFiid BOOTiid IFcor IFcorPW BOOTcor
## CA    0.338 0.096   0.101 0.117   0.203   0.146
## CTAG  0.180 0.060   0.062 0.057   0.057   0.039
## DIS   0.392 0.080   0.087 0.108   0.134   0.110
## EM    0.194 0.069   0.068 0.084   0.092   0.074
## EMN   0.543 0.110   0.098 0.116   0.124   0.184
## ED    0.372 0.079   0.076 0.100   0.111   0.084
## FIA   0.377 0.113   0.113 0.134   0.183   0.155
## GM    0.371 0.054   0.059 0.057   0.057   0.052
## LS    0.315 0.066   0.058 0.079   0.085   0.072
## MA    0.560 0.092   0.092 0.097   0.104   0.066
## RV    0.504 0.097   0.109 0.099   0.169   0.129
## SS   -0.041 0.061   0.065 0.072   0.072   0.076
## FoF   0.275 0.066   0.073 0.083   0.094   0.087
```

The value of including IFiid, along with IFcor and IFcorPW is that it allows the user to see whether or not serial correlation results in a difference in the standard error that assumes i.i.d. returns and the standard error that takes into account serial dependence. If there is no serial correlation there will not be much difference, but if there is serial correlation the difference can be considerable.

The BOOTiid and BOOTcor methods are provided for users who want to see how these bootstrap methods of computing standard errors compare with the IF based methods. Previous numerical experiments indicate that BOOTiid generally agrees quite well with IFiid, but that BOOTcor is not as consistent in giving values similar to those of IFcor.

Outliers Cleaning

The following code may be used to compare SR standard errors without and with outlier cleaning.

```
SRout <- SR.SE(edhec, se.method = "IFcorPW", cleanOutliers = F)
SRoutClean <- SR.SE(edhec, se.method = "IFcorPW", cleanOutliers = T)
clean.compare <- data.frame(SRout$IFcorPW$se, SRoutClean$IFcorPW$se)
names(clean.compare) <- c("With Outliers", "Outliers Cleaned")
row.names(clean.compare) <- names(edhec)
round(clean.compare, 3)
```

```
##      With Outliers Outliers Cleaned
## CA           0.203            0.116
## CTAG         0.057            0.057
```

```
## DIS          0.134           0.127
## EM           0.092           0.086
## EMN          0.124           0.099
## ED           0.113           0.101
## FIA          0.183           0.116
## GM           0.057           0.060
## LS           0.085           0.083
## MA           0.104           0.094
## RV           0.169           0.106
## SS           0.072           0.073
## FoF          0.093           0.084
```

Correlations of Returns and IF-transformed Returns

The (lag-1) correlations of the returns and IF-transformed returns time series can be computed as part of the output using the `corOut` argument. The options are `"retCor"`, `"retIFCor"` and `"retIFCorPW"`. Below is example code to return the correlations for the returns and the IF-transformed returns.

```
SRretCor <- SR.SE(edhec, corOut = c("retCor", "retIFCor"))
printSE(SRretCor)
```

```
##          SR IFiid IFcorPW retCor retIFCor
## CA    0.338 0.096   0.202  0.565    0.554
## CTAG  0.180 0.060   0.057 -0.008   -0.039
## DIS   0.392 0.080   0.134  0.492    0.486
## EM    0.194 0.069   0.092  0.296    0.280
## EMN   0.543 0.110   0.124  0.284    0.153
## ED    0.372 0.079   0.111  0.341    0.334
## FIA   0.377 0.113   0.183  0.500    0.440
## GM    0.371 0.054   0.057  0.057    0.060
## LS    0.315 0.066   0.086  0.216    0.252
## MA    0.560 0.092   0.104  0.277    0.130
## RV    0.504 0.097   0.169  0.424    0.480
## SS   -0.041 0.061   0.072  0.154    0.154
## FoF   0.275 0.066   0.093  0.311    0.334
```

Gamma and Exponential Distributions

In addition to the EN-penalized Gamma GLM available for the model-fitting step to the IF-transformed returns time series, the EN-penalized Exponential GLM is also available in **RPEGLMEN**. While the periodogram has an exponential distribution asymptotically, further research may show theat the Gamma distribution provides better overall results, particularly for non-normally distributed returns. The argument `fitting.method` specifies the GLM choice, and the `d.GLM.EN` argument specifies the polynomial degree for the model. By way of example, the following code computes standard errors of the SR for the exponential and Gamma distributions.

```
Clean.out <- SR.SE(edhec, se.method=c("IFiid","IFcor","IFcorPW"), cleanOutliers = T)
GammaClean.out <- SR.SE(edhec, se.method=c("IFiid","IFcor","IFcorPW"), cleanOutliers = T,
                        fitting.method = "Gamma")
GammaExp.comparison <- cbind(printSE(Clean.out)[,4], printSE(GammaClean.out)[,4])
colnames(GammaExp.comparison) <- c("IFcorPW-Exponential", "IFcorPW-Gamma")
rownames(GammaExp.comparison) <- names(edhec)
GammaExp.comparison
```

```
##      IFcorPW-Exponential IFcorPW-Gamma
## CA                 0.116         0.116
## CTAG               0.057         0.051
## DIS                0.127         0.128
## EM                 0.086         0.101
## EMN                0.099         0.114
## ED                 0.101         0.109
## FIA                0.116         0.120
## GM                 0.060         0.061
```

```
## LS                0.083         0.081
## MA                0.094         0.094
## RV                0.106         0.098
## SS                0.073         0.071
## FoF               0.084         0.092
```

Decimation and Truncation of Frequencies of Discrete Fourier Transform

There is an option in the SE functions to use a decimated or truncated percentage of the frequencies of the discrete Fourier transforms for the periodogram in the fitting of the Gamma distributions. Decimation implies that only certain frequencies are used, and they will be equally spaced selections from the frequencies. Truncation implies that only a certain percentage of the frequencies (i.e. only the first frequencies until a certain point) will be used. By default, the standard error functions use all the frequencies. If the argument `freq.include` is set to "Decimate" or "Truncate" a value of 0.5 is used for the `freq.par` argument: every second frequency is used in the decimation case, and only the first half of the frequencies are used in the truncation case. Below is some sample code demonstration.

```
SRall <- SR.SE(edhec, cleanOutliers = T, freq.include = "All")
SRdecimate <- SR.SE(edhec, cleanOutliers = T, freq.include = "Decimate",
                    freq.par = 0.5)
SRtruncate <- SR.SE(edhec, cleanOutliers = T, freq.include = "Truncate",
                    freq.par = 0.5)
frequency <- cbind(printSE(SRall)[,3], printSE(SRdecimate)[,3],
                   printSE(SRtruncate)[,3])
colnames(frequency) <- c("IFcorPW-All", "IFcorPW-Decimate", "IFcorPW-Truncate")
rownames(frequency) <- names(edhec)
frequency
```

```
##      IFcorPW-All IFcorPW-Decimate IFcorPW-Truncate
## CA         0.116            0.121            0.113
## CTAG       0.057            0.057            0.057
## DIS        0.127            0.132            0.126
## EM         0.086            0.094            0.088
## EMN        0.099            0.109            0.102
## ED         0.101            0.105            0.100
## FIA        0.116            0.122            0.116
## GM         0.060            0.060            0.061
## LS         0.083            0.086            0.083
## MA         0.094            0.097            0.093
## RV         0.106            0.116            0.109
## SS         0.073            0.079            0.072
## FoF        0.084            0.087            0.088
```

Monte Carlo Study with IF-Based Standard Errors

To assess the accuracy of risk and performance estimators standard errors for the proposed method in comparison with alternative methods available for practitioners, a Monte Carlo simulation is carried out for the SR estimator. The standard error methods included in the simulation study are the IF-based method, the Newey-West (NW) HAC method (Newey and West, 1987) and the Andrews (AN) HAC method (Andrews, 1991), as well as their prewhitened versions using the `nse.andrews` and `nse.nw` functions in the **nse** package[3]. The Monte Carlo study compares the rejection probabilities of 95% confidence intervals based on the standard error estimates of the methods. The simulations are conducted using $N = 5{,}000$ replications of AR(1) processes with sample sizes $n = 120, 240$. The normal and t-distribution with $df = 5$ are considered for the innovations of the AR(1) processes, where the mean $\mu = 1\%$ and the volatility $\sigma = 15\%$. The simulation process is as follows:

1. A sample of size n is simulated from an AR(1) process, where the innovations follow either the normal or t-distribution ($df = 5$) with mean $\mu = 1\%$ and volatility $\sigma = 15\%$.
2. The SR is estimated using the sample mean and standard deviation, $\widehat{SR} = \frac{\hat{\mu} - rf}{\hat{\sigma}}$.
3. Steps 1. and 2. are repeated $N = 5{,}000$ times resulting in the estimates $\widehat{SR}_1, \widehat{SR}_2, \ldots, \widehat{SR}_N$. The "true" standard error of the SR is computed as the sample standard deviation of the N estimates.

[3]The **nse** package is a wrapper for the **sandwich** package.

4. For each of the N simulated time series of returns, the standard error of the SR is computed using the IF, HAC AN and NW methods, as well as for their prewhitened versions.
5. Based on a normal approximation for the SR, a nominal 5% error rate confidence interval is computed for each $\widehat{SR}_i, i = 1, 2, \ldots, N$, where the confidence interval is given by

$$\left(\widehat{SR}_i - t_{\alpha/2,n-1} \times SE_i,\ \widehat{SR}_i + t_{\alpha/2,n-1} \times SE_i\right),$$

and $t_{\alpha/2,n-1}$ is the $(1-\alpha/2)$-th quantile of the t-distribution with $n-1$ degrees of freedom. The rejection probabilities are computed as the fraction of the N replicates for which the replicate confidence interval does not contain the true SR.

Steps 1-6 of the Monte Carlo simulation study are repeated for each combination of $n = 60, 120, 240$ and $\phi = 0, 0.1, \ldots, 0.5$. The results for the normally and t-distributed innovations are provided in Tables 2 and 3, respectively, for a nominal error rate of 5%. For all of the methods, the prewhitening step improved the rejection probabilities. The IF-PW method was the best method overall, achieving rejection probabilities that were closest to 5% when averaging over all values of ϕ. In particular, the IF-PW method was also a highly competitive method even in the i.i.d. case ($\phi = 0$) for both the normally and t-distributed innovations, across all sample sizes considered. For a more extensive benchmark study of the new method and HAC-based methods, including a comparison of the methods under generalized autoregressive conditional heteroscedasticity (GARCH) models, see Chen and Martin (2021).

	AR(1) Parameter Values						
	$\phi = 0$	$\phi = 0.1$	$\phi = 0.2$	$\phi = 0.3$	$\phi = 0.4$	$\phi = 0.5$	**Avg. RP**
$n = 60$							
IF-PW	5.5	5.3	5.4	6.2	5.9	7.0	5.9
AN-PW	6.0	6.2	6.2	7.6	7.5	9.3	7.1
NW-PW	7.4	7.6	7.1	8.2	8.0	9.7	8.0
IF	5.5	5.5	5.8	7.2	8.0	11.6	7.3
AN	5.3	6.4	7.0	9.5	10.3	13.1	8.6
NW	8.4	9.0	8.9	10.1	10.9	13.7	10.2
$n = 120$							
IF-PW	5.2	4.6	5.0	6.0	5.5	5.8	5.3
AN-PW	5.5	4.8	5.4	6.5	6.7	7.3	6.0
NW-PW	6.5	5.9	6.2	7.0	7.1	7.8	6.7
IF	5.3	4.8	5.6	7.0	7.8	10.4	6.8
AN	5.2	5.5	7.0	8.4	8.8	10.2	7.5
NW	7.1	6.7	7.7	8.7	9.1	10.8	8.4
$n = 240$							
IF-PW	4.7	5.3	5.4	4.8	5.4	5.4	5.2
AN-PW	4.8	5.4	5.7	5.3	6.0	6.5	5.6
NW-PW	5.1	5.8	6.0	5.6	6.1	6.5	5.8
IF	4.8	5.5	5.9	5.8	7.8	10.4	6.7
AN	4.6	6.2	7.1	6.7	7.8	8.7	6.8
NW	5.4	6.3	7.2	7.0	7.9	8.8	7.1

Table 2: Rejection probabilities (RP) of the standard error methods for normally distributed innovations. The nominal error rate is 5%.

	AR(1) Parameter Values						
	$\phi = 0$	$\phi = 0.1$	$\phi = 0.2$	$\phi = 0.3$	$\phi = 0.4$	$\phi = 0.5$	Avg. RP
$n = 60$							
IF-PW	6.6	6.7	6.9	7.2	6.7	7.9	7.0
AN-PW	7.0	7.4	8.1	8.2	8.2	9.9	8.1
NW-PW	8.2	8.1	9.1	8.8	8.8	10.2	8.9
IF	6.7	6.8	7.5	8.6	8.6	12.0	8.4
AN	6.6	7.8	9.7	10.7	11.3	13.4	9.9
NW	9.5	9.7	10.9	11.4	11.8	14.0	11.2
$n = 120$							
IF-PW	6.2	5.7	6.2	5.9	6.9	5.8	6.1
AN-PW	6.5	5.9	6.8	6.4	7.8	7.1	6.7
NW-PW	7.7	7.1	7.6	7.3	8.4	7.4	7.6
IF	6.4	5.9	6.9	7.1	9.6	9.9	7.6
AN	6.0	6.7	8.3	8.5	10.8	10.2	8.4
NW	7.8	8.0	9.0	9.3	11.1	10.4	9.3
$n = 240$							
IF-PW	5.5	5.7	5.4	5.7	5.9	6.1	5.7
AN-PW	5.7	5.9	5.5	6.1	6.4	6.6	6.0
NW-PW	6.1	6.3	5.9	6.3	6.7	6.9	6.4
IF	5.6	5.9	5.9	6.8	8.2	10.2	7.1
AN	5.6	6.8	6.9	7.6	8.3	8.9	7.3
NW	6.4	7.0	6.6	7.7	8.4	8.9	7.5

Table 3: Rejection probabilities (RP) of the standard error methods for t-distributed innovations with $df = 5$. The nominal error rate is 5%.

Summary

This article introduced the **RPESE** package, as well as the related packages **RPEIF** and **RPEGLMEN**, to compute standard errors for risk and performance estimators using the new methodology in Chen and Martin (2021). The new methodology involves the representation of risk and performance estimators in terms of their IF-transformed returns, and fitting a polynomial Gamma GLM to the spectral density of the IF time series. The **RPEIF** package implements the IF computation for six risk estimators and nine performance estimators, and provides graphical visualization tools for the IFs. The **RPEGLMEN** implements an accelerated proximal gradient algorithm for the computation of the EN-penalized polynomial Gamma GLM applied to the spectral density of the IF-transformed returns, which includes multicore parallelization capabilities.

Code examples, real data applications and benchmark simulation studies in this article demonstrate the user-friendly software implementation of the method to assess the accuracy of risk and performance estimators, providing financial risk and portfolio managers with a powerful open-source toolbox. We note that the **PerformanceAnalytics** package uses the **RPESE** package to make its standard errors for serially dependent data available to the large base of quantitatiave finance users.

There is much further research that can be accomplished for the methodology of Chen and Martin (2021) and its software implementation in **RPESE**. New risk and performance estimators can be introduced to both **RPEIF** and **RPESE**. Alternative model selection methods could be applied in **RPESE** in addition to EN regularization, such as AIC or BIC based model selection. The proposed methodology could be applied to multivariate time series to study the joint behavior of risk and

performance estimators. In the latter case, a surface fitting method will be required to apply to the multivariate spectral density of the multivariate IF-transformed returns.

Software and Data Availability

The package is available from the Comprehensive R Archive Network at https://CRAN.R-project.org/package=RPESE, where a reference manual and vignette are provided. The development website is available at https://github.com/AnthonyChristidis/RPESE. The scripts to replicate the code examples, the hedge fund data results and the Monte Carlo simulation are available at https://github.com/AnthonyChristidis/RPESE_RJournal_Simulation.

Acknowledgments

The software packages introduced in this article were developed while Anthony-Alexander Christidis was funded by a Google Summer of Code Award under the supervision of Dr. R. Doug Martin, Professor Emeritus of Applied Mathematics and Statistics at the University of Washington.

Bibliography

D. W. Andrews. Heteroskedasticity and autocorrelation consistent covariance matrix estimation. *Econometrica: Journal of the Econometric Society*, pages 817–858, 1991. URL https://www.jstor.org/stable/2938229. [p]

X. Chen and R. D. Martin. Standard errors of risk and performance estimators for serially correlated returns. *Journal of Risk*, 23:1–41, 2021. URL https://doi.org/10.21314/JOR.2020.446. [p]

X. Chen, A. Y. Aravkin, and R. D. Martin. Generalized linear model for gamma distributed variables via elastic net regularization. 2018. URL https://arxiv.org/abs/1804.07780. [p]

F. R. Hampel. The influence curve and its role in robust estimation. *Journal of the American Statistical Association*, 69(346):383–393, 1974. URL https://www.jstor.org/stable/2285666. [p]

F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel. *Robust Statistics: The Approach Based on Influence Functions*. Wiley, 1986. URL https://doi.org/10.1002/9781118186435. [p]

P. Heidelberger and P. D. Welch. A spectral method for confidence interval generation and run length control in simulations. *Communications of the ACM*, 24(4):233–245, 1981. URL https://doi.org/10.1145/358598.358630. [p]

H. R. Kunsch. The jackknife and the bootstrap for general stationary observations. *The Annals of Statistics*, pages 1217–1241, 1989. URL https://www.jstor.org/stable/2241719. [p]

O. Ledoit and M. Wolf. Robust performance hypothesis testing with the sharpe ratio. *Journal of Empirical Finance*, 15(5):850–859, 2008. URL https://doi.org/10.1016/j.jempfin.2008.03.002. [p]

A. W. Lo. The statistics of sharpe ratios. *Financial Analysts Journal*, 58(4):36–52, 2002. URL https://doi.org/10.2469/faj.v58.n4.2453. [p]

R. Maronna, R. Martin, V. Yohai, and M. Salibian-Barrera. *Robust Statistics: Theory and Methods (with R) Wiley*. 2019. URL https://doi.org/10.1002/9781119214656. [p]

R. D. Martin and S. Zhang. Nonparametric versus parametric expected shortfall. *Journal of Risk*, 21(6):1–41, 2019. URL https://ssrn.com/abstract=2747179. [p]

W. K. Newey and K. D. West. A simple, positive semi-definite, heteroskedasticity and autocorrelation consistent covariance matrix. *Econometrica*, 55(3):703–708, 1987. ISSN 00129682, 14680262. URL http://www.jstor.org/stable/1913610. [p]

D. N. Politis and J. P. Romano. The stationary bootstrap. *Journal of the American Statistical Association*, 89(428):1303–1313, 1994. URL https://doi.org/10.1080/01621459.1994.10476870. [p]

A. Zeileis. Econometric computing with hc and hac covariance matrix estimators. *Journal of Statistical Software*, 2004. URL https://doi.org/10.18637/jss.v011.i10. [p]

S. Zhang, R. D. Martin, and A.-A. Christidis. Influence functions for risk and performance estimators. *Journal of Mathematical Finance*, 11(1), 2021. URL https://doi.org/10.4236/jmf.2021.111002. [p]

Anthony-Alexander Christidis
Department of Statistics
University of British Columbia
Vancouver, British Columbia, Canada
anthony.christidis@stat.ubc.ca

R. Doug Martin
Department of Applied Mathematics
University of Washington
Seattle, Washington, United States
doug@amath.washington.edu