Madness: a package for Multivariate Automatic Differentiation

Steven E. Pav *

February 7, 2020

Abstract

The **madness** package provides a class for automatic differentiation of 'multivariate' operations via forward accumulation. By 'multivariate,' we mean the class computes the derivative of a vector or matrix or multidimensional array (or scalar) with respect to a scalar, vector, matrix, or multidimensional array. The primary intended use of this class is to support the multivariate delta method for performing inference on multidimensional quantities. Another use case is the automatic computation of the gradient in parameter optimization (*e.g.*, in the computation of an MLE). Examples of the use of this package are given in the realm of quantitative finance.

1 Introduction

The **madness** package [13] provides the ability to automatically compute and accumulate the derivative of numerical quantities on concrete data via forward accumulation. [14, 4] It can compute the derivatives of multivariate functions–those producing multivariate output–with respect to a multivarite independent variable. While the derivatives are essentially computed symbolically, they are applied immediately to concrete data. Unlike previous attempts at automatic differentiation in **R**, `madness` takes a 'high level' approach. [2, 1] That is, rather than provide methods for computing the derivatives of a few basic operators like sum, product, exponent and some trigonometrics, which would be applied at the lowest level of more complicated functions, the eponymous `madness` class supports functions like the Cholesky factor, the matrix square root, matrix inversion, computing eigenvalues and so on. Because many of these linear algebra operations are typically computed at the lowest level in C code, a 'low level' approach which infects basic operations like sum and product could not be easily applied.

The target application is the multivariate delta method. Informally, the multivariate delta method claims that a function commutes with a consistent estimator of some population quantity, while the covariance gets 'wrapped' with the derivative of the applied function. That is, if β is some population quantity, and B is some consistent estimator of β with

$$\sqrt{n}\left(B - \beta\right) \xrightarrow{D} \mathcal{N}\left(\mathbf{0}, \Omega\right),$$

*shabbychef@gmail.com

based on n independent observations, and $f(\cdot)$ is some function which is continuous and non-zero at β, then

$$\sqrt{n}\left(f\left(B\right)-f\left(\beta\right)\right)\xrightarrow{D}\mathcal{N}\left(\mathbf{0},\frac{\mathrm{d}f\left(x\right)}{\mathrm{d}x}^{\top}\Omega\frac{\mathrm{d}f\left(x\right)}{\mathrm{d}x}\bigg|_{x=\beta}\right).$$

Practically speaking, this means that if you can compute a consistent estimator (*e.g.*, by taking a simple mean and relying on the central limit theorem), *and you can compute derivatives*, you can estimate the variance-covariance of some really weird estimators. The `madness` package aims to compute those derivatives for you.

Nota bene The **madness** package is in a state of flux. This document describes version 0.2.7, 0.2.6, 0.2.6 of the package, but should be applicable for more recent versions.

2 Basic usage

The `madness` class is an `S4` class with the following slots:

- The dependent variable, `val`, a multidimensional numeric.
- The derivative of the dependent variable with respect to some implicit independent variable, `dvdx`, a matrix. The matrix is stored in 'numerator layout,' where the derivative of a scalar with respect to a vector is a *row* vector. This is inconsistent with traditional representation of a gradient as a column, but notationally more convenient.
- Optionally the 'tag' of the value, `vtag` is stored. This keeps track of the operations applied to the value, and is useful for debugging.
- Optionally the 'tag' of the independent variable, `xtag` is stored. While this tag is optional, it is important to note that two `madness` objects with *different* `xtag` values cannot be used in the same computation. For example, attempting to add them results in an error, since they are considered to track the derivatives with respect to different independent variables.
- Optionally the variance-covariance of the independent variable is stored in `varx`. This is convenient for the multivariate delta method. One can call the `vcov` method on a `madness` object with a non-null `varx`, and the delta method will be applied.

2.1 Object construction

One can get data into a `madness` object by calling the `madness` function. The derivative `dvdx` will default to the identity matrix. That is, the constructor assumes that the dependent variable *is* the independent variable. The constructor also guesses the tags for the independent and dependent variables by the name of the input variable. The `show` method shows a `madness` object, just showing the head of the value and derivative:

```
require(madness)
set.seed(1234)
X_NAMED <- array(rnorm(3), dim = c(3, 1))
Xmad <- madness(X_NAMED)
show(Xmad)
```

```
## class: madness
##         d X_NAMED
##  calc: -----------
##         d X_NAMED
##   val: -1.2 ...
##  dvdx: 1 0 0 ...
##  varx:  ...
```

One can get the value, the derivative, tags, and so on with eponymous getter methods, `val`, `dvdx`, `xtag`, `vtag`, `varx`:

```
show(val(Xmad))

##       [,1]
## [1,] -1.21
## [2,]  0.28
## [3,]  1.08

show(dvdx(Xmad))

##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    1    0
## [3,]    0    0    1
```

One can also construct a `madness` object via the `as.madness` function which calls the `coef` method and the `vcov` method on the input. So, for example, one can easily convert an object of class `lm` to a `madness`:

```
set.seed(456)
a_df <- data.frame(x = rnorm(1000), y = runif(1000),
    z = runif(1000))
a_df$v <- rowSums(a_df) + rnorm(nrow(a_df))
beta <- lm(v ~ x + y + z, data = a_df)
bmad <- as.madness(beta, vtag = "beta")
show(bmad)

## class: madness
##         d beta
##  calc: --------
##         d beta
##   val: -0.035 ...
##  dvdx: 1 0 0 0 ...
##  varx: 0.0065 -2.9e-05 -0.0056 -0.0055 ...
```

There are also two functions which construct a `madness` object from data:

- `twomoments`, which computes the sample mean and covariance of n independent observations of a p vector given in a $n \times p$ matrix.
- `theta`, which computes the uncentered second moment matrix of n independent observations of a p vector given in a $n \times p$ matrix.

Both methods allow one to feed in a more 'exotic' variance-covariance estimator than the default `stats::vcov`. More importantly, both methods properly take into account the symmetry of the output. If one blindly stuffed a *e.g.*, covariance

matrix into a `madness` object, one could easily overestimate the variance of ones estimate by effectively ignoring that any estimate has to be symmetric, and thus diagonal-mirrored elements do not vary independently.

```
set.seed(789)
X <- matrix(rnorm(1000 * 3), ncol = 3)
# one of these apparently does not run under
# alternative BLAS, and so CRAN precludes me from
# executing this code in the vignette, but I cannot
# tell which because I cannot use those alternative
# BLAS, so I am commenting out this code, which is
# absurd. Xmad <- theta(X) show(Xmad)

# more 'exotic' variance-covariance:
library(sandwich)
set.seed(1111)
X <- matrix(rnorm(100 * 2), ncol = 2)

# twom <- twomoments(X,vcov=sandwich::vcovHAC)
# show(twom)
```

2.2 Methods

Obviously, to be of maximal use, the `madness` class should support any method a reasonable user throws at it. Setting aside the definition of 'reasonable,' many methods have been implemented for the `madness` class: unary minus; element-wise binary sum, product, difference, ratio, power; matrix product and Kronecker product; accumulating sum and product; element-wise unary exponentiation, logarithm, and trigonometrics; `colSums`, `rowSums`, `colMeans`, `rowMeans`; matrix trace, determinant, matrix inverse, `solve`; Cholesky factor, symmetric square root, and `eigen`; matrix norms; `outer` with a limited set of functions; reshape operations; extracting lower, upper triangle, or diagonal; `cbind`, `rbind` and concatenation; subselecting elements.

Since not every conceivable function can be implemented, there is a method, `numderiv` which approximates derivatives numerically, producing a `madness` object. While symbolically computed derivatives are typically preferred, numerical approximations are preferred to an unusable half-solution. Indeed, the numerical approximations are used in the unit tests to ensure the derivatives are correctly computed. Moreover, the goal is to simplify the computation and use of derivatives, which is not aided by a dogmatic adherence to symbolic derivation.

Some example computations showing methods performed on `madness` objects:

```
set.seed(2223)
X <- matrix(runif(5 * 3), ncol = 3)
Y <- matrix(rnorm(length(X)), ncol = ncol(X))
Xmad <- madness(X, xtag = "v")
Ymad <- madness(Y, xtag = "v")

Zmad <- Xmad + Ymad
```

```
# hadamard product:
Zmad <- Xmad * Ymad
# matrix product:
Zmad <- t(Xmad) %*% Ymad
# equivalently
Zmad <- crossprod(Xmad, Ymad)

# can also interact with a scalar:
Zmad <- Xmad + Y
Zmad <- t(Xmad) %*% Y
# and so on.

# not sure _why_ you want to do these, but they can
# be done:
foo <- Xmad^Ymad
foo <- log(Xmad)
foo <- outer(Xmad, Y, "+")

# some sums and such:
cboth <- c(colSums(Xmad), colSums(Ymad))
xsum <- sum(Xmad)

# square matrix operations:
Zmad <- crossprod(Xmad, Ymad)
foo <- matrix.trace(Zmad)
foo <- det(Zmad)
invZ <- solve(Zmad)
invZ <- solve(Zmad, crossprod(Y, Y))

# and so on...
```

3 Examples

We further illustrate the use of the `madness` class with real examples.

3.1 The Sharpe ratio

The Sharpe ratio is arguably the most popular metric for comparing the historical (or backtested) performance of assets. It is, however, a sample statistic, and represents a noisy estimate of some population parameter, which we will call the *signal-noise ratio*. The asymptotic standard error of the Sharpe ratio was given by Johnson and Welch, Jobson and Korkie, and others. [8, 7, 10] This statistic, and its approximate standard error, can easily be computed with a `madness` object, here applied to the Fama-French 3 factors weekly returns. [**?**] The data were downloaded from French's website, and comprise 4800 weeks of data, from 1926-07-02 to 2018-06-29.

```
data(wff3)
wff3$Mkt_RF <- wff3$Mkt - wff3$RF
ff3 <- wff3[, c("Mkt_RF", "SMB", "HML")]
# compute first and second moments: (beware: this
```

```
# method will not scale to larges numbers of
# assets!)
two <- twomoments(ff3, diag.only = TRUE)
# annualization factor:
ope <- 52
srs <- sqrt(ope) * two$mu/sqrt(two$sigmasq)

show(val(srs))

##      [,1]
## [1,] 0.43
## [2,] 0.21
## [3,] 0.43

show(vcov(srs))

##        [,1]     [,2]     [,3]
## [1,] 0.0111  0.00103  0.00194
## [2,] 0.0010  0.01084 -0.00039
## [3,] 0.0019 -0.00039  0.01029

# for comparison:
library(SharpeR)
show(sr_vcov(as.matrix(ff3), ope = ope))

## $SR
##        [,1]
## Mkt_RF 0.43
## SMB    0.21
## HML    0.43
##
## $Ohat
##        Mkt_RF      SMB      HML
## Mkt_RF 0.0111  0.00103  0.00194
## SMB    0.0010  0.01084 -0.00039
## HML    0.0019 -0.00039  0.01029
##
## $p
## [1] 3
```

In fact, here we have illustrated the computation of the Sharpe ratio of not a single asset, but of three assets. We can perform tests of equality of the signal-noise ratio of different assets. [9, 18]

```
# test whether SMB has same signal-noise as HML:
testv <- t(srs) %*% array(c(0, -1, 1), dim = c(3, 1))
# now the Wald statistic:
wald <- as.numeric(val(testv))/sqrt(diag(vcov(testv)))
show(wald)

## [1] 1.5
```

Here we demonstrate the computation of the Wald statistic: a quantity of interest, typically assumed to be zero under the null hypothesis, divided by its

approximate standard error. In this case the Wald statistic is nowhere near the 'magical' value of 2, and we have little reason to doubt the null hypothesis that `SMB` and `HML` have the same signal-noise ratio.

3.1.1 Fighting overfit of the Sharpe ratio

The following recipe for quantitative strategy development is widely followed in industry:

1. Write a piece of code which converts historical data to predicted returns or target portfolio at point in time.
2. Backtest the code with all available historical data.
3. If the Sharpe ratio of the backtested returns is not satisfactory, add more features to the trading strategy code, and repeat the backtest cycle.
4. When the backtested Sharpe ratio is high enough, productionalize the model.

When presented in this way, one suspects such a practice would yield unsatisfactory results[1]. Numerous tests have been devised to fight this kind of 'data-snooping' bias. [17, 6, 5]

Here we develop another approach to overfitting which models signal-noise ratio in terms of the various attributes of the trading strategy being tested. Formally, suppose that one records l 'features' about each strategy which has been backtested. Let $\boldsymbol{f}_i$ be the vector of features pertaining to the i^{th} strategy, for $i = 1, 2, \ldots, k$. For example, suppose one is testing a moving average crossover strategy. The features vector might be the lengths of the two averaging windows. More elaborate strategies might have long feature vectors, with information about lookback windows for features, which features are included, how the predictive model was constructed, the form of the covariance estimator, what instruments are hedged out, how portfolio optimization is performed, and so on.

Letting ζ_i be the signal-noise ratio of this strategy, the simplest linear model posits that $\zeta_i = \boldsymbol{f}_i{}^{\top}\boldsymbol{\beta}$. When testing this model, one should take care to express the features in such a way that would allow arbitrarily high signal-noise ratio by extrapolating away from the tested feature set. This may require some imagination.

One collects the backtested returns on the k strategies, then computes the Sharpe ratio of these, along with the variance-covariance matrix of these. One can then use linear regression to estimate $\boldsymbol{\beta}$. By performing this calculation with a `madness` object, one can compute the marginal Wald statistics associated with each element of the feature vector. Here we present a simple example using fake backtested returns. First imagine some process (hidden here) generates the returns on 1771 of data over 400. Moreover, the returns are some linear combination of 25 latent returns. The loadings on 5 of these are observed as the features of the different strategies, including all those with non-zero signal-noise ratio. The true $\boldsymbol{\beta}$ in this case is $[0.2, -0.1, 0, 0, 0]^{\top}$. Then proceed as follows:

```
show(dim(Rets))
```

[1] Actually, this depends on the background rate of profitable trading strategies. If one could randomly stumble upon strategies with high signal-noise ratio, this recipe might be fruitful. This is not commonly experienced, however.

```
## [1] 1771  400

show(dim(F_mat))

## [1]   5 400

# use madness.
two <- twomoments(Rets, diag.only = TRUE)
srs <- two$mu/sqrt(two$sigmasq)

# the normal equations method. This is typically
# numerically unstable and not recommended, but I
# have not implemented QR factorization yet...
betahat <- solve(tcrossprod(F_mat, F_mat), F_mat %*%
    srs)
show(val(t(betahat)))

##      [,1]   [,2]   [,3]   [,4]   [,5]
## [1,] 0.17 -0.071 -0.012 -0.012 -0.012

marginal_wald <- val(betahat)/sqrt(diag(vcov(betahat)))
show(t(marginal_wald))

##      [,1] [,2]  [,3]  [,4]  [,5]
## [1,]  6.3 -2.5 -0.51 -0.44 -0.41
```

In this case, with 400 backtests of 1771 days of returns, the marginal Wald statistics correctly identify the first two features as significantly non-zero.

3.2 The ex-factor Sharpe ratio

Loosely, the *information ratio* is the Sharpe ratio of returns *in excess of some non-constant benchmark*. This assumes that the proper 'beta' of the investment with respect to the benchmark is exactly one. A more pessimistic model of the returns of an asset is essentially that of Arbitrage Pricing Theory, which expresses the returns of an asset as the linear combination of the returns of some common risk factors. For the purposes of estimating whether an investment strategy has any idiosyncratic 'alpha', this is equivalent to regressing the historical returns against the historical returns of the risk factors, and assessing whether the intercept term is significantly non-zero.

Rather than perform a hypothesis test, we can perform inference on the intercept term divided by the volatility, here given the unfortunate name of *ex-factor signal-noise ratio*. The model is as follows:

$$x_t = \beta_0 1 + \sum_i^{l-1} \beta_i f_{i,t} + \epsilon_t, \tag{1}$$

where x_t is the return of the asset at time t, $f_{i,t}$ is the value of some i^{th} 'factor' at time t, and the innovations, ϵ, are assumed to be zero mean, and have standard deviation σ. Here we have forced the zeroth factor to be the constant one, $f_{0,t} = 1$.

Given n observations, let F be the $n \times l$ matrix whose rows are the observations of the factors (including a column that is the constant 1), and let $\boldsymbol{x}$ be the

n length column vector of returns; then the multiple linear regression estimates are

$$\hat{\boldsymbol{\beta}} =_{\text{df}} \left(\mathsf{F}^\top \mathsf{F}\right)^{-1} \mathsf{F}^\top \boldsymbol{x}, \qquad \hat{\sigma} =_{\text{df}} \sqrt{\frac{\left(\boldsymbol{x} - \mathsf{F}\hat{\boldsymbol{\beta}}\right)^\top \left(\boldsymbol{x} - \mathsf{F}\hat{\boldsymbol{\beta}}\right)}{n - l}}. \tag{2}$$

We can then define a *ex-factor Sharpe ratio* as follows: let $\boldsymbol{v}$ be some non-zero vector, and let r_0 be some risk-free, or disastrous, rate of return. Then define

$$\hat{\zeta}_g =_{\text{df}} \frac{\hat{\boldsymbol{\beta}}^\top \boldsymbol{v} - r_0}{\hat{\sigma}}. \tag{3}$$

The ex-factor signal-noise ratio appears in a transform of the 'theta' matrix which encompasses both first and second moments of a distribution. [12] Let

$$\tilde{\boldsymbol{x}}_i =_{\text{df}} \left[x_i, \boldsymbol{f}_i^\top\right]^\top.$$

Define the second moment of this as

$$\Theta =_{\text{df}} \mathrm{E}\left[\tilde{\boldsymbol{x}}\tilde{\boldsymbol{x}}^\top\right].$$

First note that

$$\Theta = \begin{bmatrix} \sigma^2 + \beta^\top \Gamma_f \beta & \beta^\top \Gamma_f \\ \Gamma_f \beta & \Gamma_f \end{bmatrix}, \tag{4}$$

where Γ_f is the uncentered second moment of $\boldsymbol{f}$. Simple matrix multiplication confirms that the inverse of Θ is

$$\Theta^{-1} = \begin{bmatrix} \sigma^{-2} & -\beta^\top \sigma^{-2} \\ -\beta\sigma^{-2} & \Gamma_f^{-1} + \sigma^{-2}\beta\beta^\top \end{bmatrix}, \tag{5}$$

and the Cholesky factor of that inverse is

$$\Theta^{-1/2} = \begin{bmatrix} \sigma^{-1} & 0 \\ -\beta\sigma^{-1} & \Gamma_f^{-1/2} \end{bmatrix}. \tag{6}$$

The ex-factor signal-noise ratio (*cf.* Equation 3) can thus be expressed as

$$\zeta_g = \frac{\boldsymbol{\beta}^\top \boldsymbol{v} - r_0}{\sigma} = -\left[r_0, \boldsymbol{v}^\top\right] \Theta^{-1/2} \boldsymbol{e}_1. \tag{7}$$

Up to scaling by some factor of n and l, which becomes immaterial for large n, the sample ex-factor Sharpe ratio takes the same form in the sample analogue.

We demonstrate this computation by grabbing the weekly simple returns of `AAPL` and `IBM`, then attributing them to the Fama French three factor weekly returns. We compute the ex-factor Sharpe ratio to test for idiosyncratic alpha by computing the intercept term divided by the volatility. Because we estimate the variance-covariance of the combined vector of returns, we can estimate the variance-covariance of our estimates of the ex-factor signal-noise ratios together. Again we stress that the hard work is in gathering the data together, putting them in the right form, and sanely computing the estimate. The `madness` class automatically computes the derivatives and the marginal Wald statistics are trivial to compute. Here we apply this analysis to the weekly returns of `AAPL` and of `IBM`, collected over 1930 weeks from 1981-01-09 to 2017-12-29, as downloaded from Quandl. [15] We will perform attribution against the Fama-French factor weekly returns considered earlier. The tail of the data looks as follows:

```
data(wff3)
data(stock_returns)
allweekly <- stock_returns %>% mutate(AAPL = 100 *
    AAPL, IBM = 100 * IBM) %>% left_join(wff3, by = "Date") %>%
    mutate(Mkt_RF = Mkt - RF) %>% dplyr::select(-Mkt)

tail(allweekly, 6) %>% dplyr::select(-RF) %>% knitr::kable(row.names = FALSE)
```

Date	AAPL	IBM	SMB	HML	Mkt_RF
2017-11-24	2.79	1.91	1.11	-0.84	1.04
2017-12-01	-2.27	1.90	-0.90	2.22	1.61
2017-12-08	-0.99	0.03	-1.44	0.16	0.30
2017-12-15	2.68	-1.50	-0.24	-0.70	0.81
2017-12-22	0.60	0.00	0.79	0.48	0.54
2017-12-29	-3.36	0.60	0.01	-0.20	-0.37

We now perform the attributions and test them for significance:

```
tht <- theta(allweekly %>% dplyr::select(AAPL, IBM,
    Mkt_RF, SMB, HML) %>% mutate(one = 1), xtag = "stocks")

thinv_aapl <- chol(solve(tht[c(1, 3, 4, 5, 6), c(1,
    3, 4, 5, 6)]))
thinv_ibm <- chol(solve(tht[c(2, 3, 4, 5, 6), c(2,
    3, 4, 5, 6)]))
r0 <- 1e-04
v <- c(0, 0, 0, 1)
r0v <- array(c(r0, v), dim = c(5, 1))

exfacsr_aapl <- -(t(r0v) %*% t(thinv_aapl))[1, 1]
exfacsr_ibm <- -(t(r0v) %*% t(thinv_ibm))[1, 1]
exfacsr <- c(exfacsr_aapl, exfacsr_ibm)

show(cov2cor(vcov(exfacsr)))

##      [,1] [,2]
## [1,] 1.00 0.12
## [2,] 0.12 1.00

waldboth <- val(exfacsr)/sqrt(diag(vcov(exfacsr)))
show(waldboth)

##      [,1]
## [1,] 1.36
## [2,] 0.63
```

Here we conclude that the ex-factor signal-noise ratio of `AAPL` is greater than the hurdle rate of 1 bp per week, but that of `IBM` is not. The correlation of the errors of our estimates is estimated to be fairly small. We can also perform a paired test for whether the ex-factor signal-noise ratio of `AAPL` is greater than that of `IBM` by taking the difference in our estimates, and trivially computing the Wald statistic. In this case, the evidence does not strongly support that `AAPL` has higher idiosyncratic alpha than `IBM`:

```
isbigger <- array(c(1, -1), dim = c(1, 2)) %*% exfacsr
show(val(isbigger)/sqrt(diag(vcov(isbigger))))

##      [,1]
## [1,] 0.56
```

3.3 The Markowitz portfolio

The Markowitz portfolio is the unconstrained portfolio that maximizes the signal-noise ratio. For a vector of returns of p assets, if the unconditional expected return is $\boldsymbol{\mu}$, and the covariance of returns is $\boldsymbol{\Sigma}$, then the Markowitz portfolio is

$$\boldsymbol{\nu}_* =_{\text{df}} \lambda \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}, \tag{8}$$

where λ is some positive constant chosen to respect a cap on portfolio volatility (or leverage).

Since the population parameters $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are unknown, they must be estimated from the data. The noisy estimates may be unreliable, and one may wish to check the standard error around the portfolio weights. This can be found under assumptions of normality, or by using the 'theta' matrix, but computing directly via a `madness` object. [3, 12] Here we compute the Markowitz portfolio on the 1930 weeks of weekly returns, from 1981-01-09 to 2017-12-29, of the Fama-French three factor data and of `AAPL` and `IBM` discussed above[2].

```
library(sandwich)
twom <- twomoments(allweekly %>% select(AAPL, IBM,
    Mkt_RF, SMB, HML), vcov = sandwich::vcovHAC, diag.only = FALSE)
the_mp <- solve(twom$Sigma, twom$mu)
show(val(t(the_mp)))

##        [,1]   [,2]  [,3]  [,4]  [,5]
## [1,] 0.0055 0.0039 0.021 0.016 0.065

show(vcov(the_mp))

##           [,1]     [,2]     [,3]     [,4]    [,5]
## [1,]  1.9e-05 -5.0e-06 -1.6e-05 -1.1e-06 1.2e-05
## [2,] -5.0e-06  7.4e-05 -5.7e-05  8.6e-06 1.5e-05
## [3,] -1.6e-05 -5.7e-05  2.1e-04  3.8e-05 5.4e-05
## [4,] -1.1e-06  8.6e-06  3.8e-05  3.7e-04 6.0e-05
## [5,]  1.2e-05  1.5e-05  5.4e-05  6.0e-05 4.5e-04

# let's normalize to unit gross leverage:
mp_norm <- outer(the_mp, norm(the_mp, "1"), "/")
dim(mp_norm) <- dim(the_mp)

show(val(t(mp_norm)))

##       [,1]  [,2] [,3] [,4] [,5]
## [1,] 0.049 0.035 0.19 0.14 0.58
```

[2]It should be recognized that one can *not* trade on the Fama French factors directly, that there is a selection bias in our choice of stocks, and so on. This is just an example.

```
show(cov2cor(vcov(mp_norm)))

##       [,1]    [,2]  [,3]   [,4]    [,5]
## [1,]  1.00 -0.1231 -0.27 -0.221  0.2390
## [2,] -0.12  1.0000 -0.56 -0.046  0.0057
## [3,] -0.27 -0.5597  1.00 -0.233 -0.2280
## [4,] -0.22 -0.0460 -0.23  1.000 -0.7949
## [5,]  0.24  0.0057 -0.23 -0.795  1.0000
```

More elaborate inference on the Markowitz portfolio is possible via the 'theta' matrix. Computation of theta requires one to choose 'features' for prediction of returns–either constant one for the unconditional model, or some time varying state variables for the linear conditional expectation model. [12] Using the `theta` method requires one to bind the features to the returns. Here we perform this computation on the two stocks and the Fama French weekly returns.

```
library(sandwich)
tht <- theta(allweekly %>% mutate(one = 1) %>% select(one,
    AAPL, IBM, Mkt_RF, SMB, HML), xtag = "all5", vcov = sandwich::vcovHAC)
```

Suppose that Θ is somehow known to be reduced rank. We can perform inference on the Markowitz portfolio by computing the pseudoinverse of $\hat{\Theta}$ and computing the sample Markowitz portfolio, performing inference on its elements. [12] Here we show this calculation assuming that Θ is of rank 2.

```
rnk <- 2
ev <- eigen(tht, symmetric = TRUE)
evals <- ev$values[, 1:rnk]
evecs <- ev$vectors[, 1:rnk]
thtinv <- evecs %*% todiag(evals^-1) %*% t(evecs)

the_mp2 <- -thtinv[2:nrow(thtinv), 1]
show(val(t(the_mp2)))

##         [,1]   [,2]     [,3]    [,4]    [,5]
## [1,] 3.8e-05 -6e-04 -0.00023 1.3e-05 2.5e-05

show(vcov(the_mp2))

##          [,1]     [,2]     [,3]     [,4]     [,5]
## [1,]  5.1e-08 -1.2e-07 -4.5e-08  3.9e-09  3.4e-09
## [2,] -1.2e-07  3.6e-07  1.3e-07 -1.1e-08 -1.1e-08
## [3,] -4.5e-08  1.3e-07  4.9e-08 -4.0e-09 -4.1e-09
## [4,]  3.9e-09 -1.1e-08 -4.0e-09  4.2e-10  3.1e-10
## [5,]  3.4e-09 -1.1e-08 -4.1e-09  3.1e-10  5.1e-10
```

Comparing the Markowitz portfolio computed here to the one computed previously, we see that the weights for the Fama French factors are much smaller in magnitude, while the weight for `AAPL` is relatively unchanged.

3.4 Correlation matrix

We can trivially use the covariance computed by `twomoments` to compute a correlation matrix. Here we demonstrate this use on the Fama French three factor weekly returns. We compute the Wald statistics of the three off-diagonal correlations, finding that the correlation of weekly returns between `MktRF` and `SMB`, and between returns between `MktRF` and `HML` is likely to be significantly non-zero, while the correlation between `SMB` and `HML` is apparently very close to zero:

```
library(sandwich)
data(wff3)
wff3$Mkt_RF <- wff3$Mkt - wff3$RF
ff3 <- wff3[, c("Mkt_RF", "SMB", "HML")]
# compute first and second moments:
two <- twomoments(ff3, vcov = sandwich::vcovHAC)
# basically cov2cor:
fcorr <- two$Sigma/tcrossprod(sqrt(diag(two$Sigma)))
show(val(fcorr))

##       [,1]   [,2]   [,3]
## [1,] 1.000  0.086  0.188
## [2,] 0.086  1.000 -0.028
## [3,] 0.188 -0.028  1.000

# compute the Wald statistic of the off-diagonal
# correlations:
odiag <- vech(fcorr, -1)
wald <- val(odiag)/sqrt(diag(vcov(odiag)))
show(wald)

##       [,1]
## [1,]  2.24
## [2,]  3.59
## [3,] -0.75
```

3.5 As an objective function

The `madness` class can be of some limited use when writing objective functions[3]. For this purpose, the `to_objective` method converts a `madness` object representing a scalar into a numerical value with a `gradient` attribute. Consider this artificial example of a matrix factorization objective with a penalty for highly negative elements:

```
fitfun <- function(R, L, Y, nu = -0.1) {
    Rmad <- madness(R)
    dim(Rmad) <- c(ncol(L), ncol(Y))
    Err <- Y - L %*% Rmad
    penalty <- sum(exp(nu * Rmad))
```

[3] Automatic computation of the Hessian matrix would improve this area of functionality, but it is not clear how this would interoperate with support for computing derivatives of multivariate-valued functions.

```
    fit <- norm(Err, "f") + penalty
    # convert to an objective:
    to_objective(fit)
}
set.seed(1234)
L <- array(runif(30 * 5), dim = c(30, 5))
Y <- array(runif(nrow(L) * 20), dim = c(nrow(L), 20))
R0 <- array(runif(ncol(L) * ncol(Y)), dim = c(ncol(L),
    ncol(Y)))
Rk <- nlm(fitfun, R0, L, Y, iterlim = 30)

show(c(fitfun(R0, L, Y)))

## [1] 116

show(c(fitfun(Rk$estimate, L, Y)))

## [1] 105
```

4 Future Directions

To make this package more useful for the computation of objective functions, the second derivative should also be computed and maintained during operations. Moreover, use of higher-order derivatives could also be useful for application of the delta method when the sample size is so small that estimators are seriously biased. It is challenging to add this feature while keeping the 'high-level' approach to automatic differentiation, since the second derivative of matrix-to-matrix operations like the Cholesky factorization are hard to code.

References

[1] Chidambaram Annamalai. *radx: Automatic Differentation in R*, 2010. URL https://github.com/quantumelixir/radx. R package version 0.1.

[2] Chidambaram Annamalai. *tada: templated automatic differentiation*, 2010. URL https://github.com/quantumelixir/tada.

[3] Mark Britten-Jones. The sampling error in estimates of mean-variance efficient portfolio weights. *The Journal of Finance*, 54(2):655–671, 1999. URL http://www.jstor.org/stable/2697722.

[4] A. Griewank and A. Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation, Second Edition*. SIAM e-books. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 2008. ISBN 9780898717761. URL https://books.google.com/books?id=xoiiLaRxcbEC.

[5] P. R. Hansen. A test for superior predictive ability. *Journal of Business and Economic Statistics*, 23(4), 2005. doi: 10.1198/073500105000000063. URL http://pubs.amstat.org/doi/abs/10.1198/073500105000000063.

[6] Po-Hsuan Hsu, Yu-Chin Hsu, and Chung-Ming Kuan. Testing the predictive ability of technical analysis using a new stepwise test without data snooping bias. *Journal of Empirical Finance*, 17(3):471 – 484, 2010. ISSN 0927-5398. doi: 10.1016/j.jempfin.2010.01.001. URL http://ssrn.com/abstract=1087044.

[7] J. D. Jobson and Bob M. Korkie. Performance hypothesis testing with the Sharpe and Treynor measures. *The Journal of Finance*, 36(4):pp. 889–908, 1981. ISSN 00221082. URL http://www.jstor.org/stable/2327554.

[8] N. L. Johnson and B. L. Welch. Applications of the non-central t-distribution. *Biometrika*, 31(3-4):362–389, mar 1940. doi: 10.1093/biomet/31.3-4.362. URL http://dx.doi.org/10.1093/biomet/31.3-4.362.

[9] Pui-Lam Leung and Wing-Keung Wong. On testing the equality of multiple Sharpe ratios, with application on the evaluation of iShares. *Journal of Risk*, 10(3):15–30, 2008. URL http://www.risk.net/digital_assets/4760/v10n3a2.pdf.

[10] Andrew W. Lo. The Statistics of Sharpe Ratios. *Financial Analysts Journal*, 58(4), July/August 2002. URL http://ssrn.com/paper=377260.

[11] Jan R. Magnus and H. Neudecker. *Matrix Differential Calculus with Applications in Statistics and Econometrics*. Wiley Series in Probability and Statistics: Texts and References Section. Wiley, 3rd edition, 2007. ISBN 9780471986331. URL http://www.janmagnus.nl/misc/mdc2007-3rdedition.

[12] Steven E. Pav. Asymptotic distribution of the Markowitz portfolio. Privately Published, 2013. URL http://arxiv.org/abs/1312.0557.

[13] Steven E. Pav. *madness: Automatic Differentiation of Multivariate Operations*, 2018. URL https://CRAN.R-project.org/package=madness. R package version 0.2.5.

[14] L. B. Rall. *Automatic Differentiation: Techniques and Applications*. Lecture Notes in Computer Science. Springer, 1981. ISBN 9783540108610. URL http://books.google.com/books?id=5QxRAAAAMAAJ.

[15] Raymond McTaggart, Gergely Daroczi, and Clement Leung. *Quandl: API Wrapper for Quandl.com*, 2015. URL https://CRAN.R-project.org/package=Quandl. R package version 2.7.0.

[16] Stephen A. Ross. The arbitrage theory of capital asset pricing. *Journal of Economic Theory*, 13(3):341–360, 1976. URL http://linkinghub.elsevier.com/retrieve/pii/0022053176900466.

[17] Halbert White. A reality check for data snooping. *Econometrica*, 68:1097–1127, 2000. URL http://www.econ.ucsd.edu/~mbacci/white/pub_files/hwcv-077.pdf.

[18] John Alexander Wright, Sheung Chi Phillip Yam, and Siu Pang Yung. A test for the equality of multiple Sharpe ratios. *Journal of Risk*, 16(4), 2014. URL http://www.risk.net/journal-of-risk/journal/2340044/latest-issue-of-the-journal-of-risk-volume-16-number-4-2014.