It’s Madness!
(Multivariate Automatic Differentiation)

Steven E. Pav
steven@gilgamath.com

Gilgamath Consulting

May 20, 2016
What is it?

It’s **madness**:

- Package for ‘multivariate’ automatic differentiation.
- Available on [CRAN](https://CRAN.R-project.org) and [github](https://github.com).
- ‘Multivariate’: supports array to array operations. If $v \leftarrow f(x)$, then $\text{dim}(\text{dvdx}(f)) == \text{c(length(v),length(x))}$.
- Should act just like a numeric, with derivatives computed for you.
- Derivatives via forward accumulation and chain rule. No fancy tricks like NSE.
- S4 object with lots of overloaded methods.
Motivation

- Primary use case is multivariate delta method.

\[
\left\{ \sqrt{n} \left( \bar{X} - X \right) \rightsquigarrow \mathcal{N} \left( 0, \Omega \right) \right\} \Rightarrow \left\{ \sqrt{n} \left( f (\bar{X}) - f (X) \right) \rightsquigarrow \mathcal{N} \left( 0, \frac{df}{dX} \Omega \frac{df}{dX}^T \right) \right\}
\]

- Allow user to transform estimates via \( f \), automatically compute \( \frac{df}{dX} \).
- Maybe also useful for constructing objective functions.
blah blah derivatives

(w/ apologies to GARY LARSON)

What we say to users

madness computes derivatives
of multivariate operations automatically. Having derivatives is useful for inference via the delta method...

What they hear

blah blah derivatives
blah blah derivatives
blah blah derivatives
blah blah derivatives
blah blah blah
Basic use I

Initialize with independent variable:

```r
X_independent <- array(1:4, dim = c(1, 4))
X_mad <- madness(X_independent)
show(X_mad)
```

```r
# class: madness
## calc: -----------------
## d X_independent
## val: 1 2 3 4 ...
## dvdx: 1 0 0 0 ...
## varx: ...
```

```r
show(diag(dvdx(X_mad)))
```

```r
## [1] 1 1 1 1
```
Basic use II

Initialize with independent variable:

```r
X_independent <- array(1:4, dim = c(1, 4))
X_mad <- madness(X_independent)
show(X_mad^2)
```

```r
## class: madness
## d (X_independent ^ numeric)
## calc: -----------------------------
## d X_independent
## val: 1 4 9 16 ...
## dvdx: 2 0 0 0 ...
## varx: ...
```

```r
show(diag(dvdx(X_mad^2)))
```

```r
## [1] 2 4 6 8
```
Basic use III

```r
X_mad <- madness(array(1:4, dim = c(1, 4)))
show(diag(dvdx(abs(cos(X_mad)) ^ sqrt(X_mad))))

## [1] -1.01 0.80 0.24 -1.03

show(dvdx(prod(X_mad)))

## [1,] 24 12  8  6

show(dvdx(colSums(crossprod(X_mad))))

## [1,] 11  1  1  1
## [2,]  2 12  2  2
## [3,]  3  3 13  3
## [4,]  4  4  4 14
```
Intermediate use I

as.madness promotes objects with coef and optional vcov methods:

```r
set.seed(12345)  # mental note: change gmail password
adf <- data.frame(x = rnorm(1000), y = rnorm(1000))
alm <- lm(y ~ x, data = adf)
amad <- as.madness(alm)
show(cov2cor(vcov(amad^2)))
```

```
## [,1] [,2]
## [1,] 1.000 0.046
## [2,] 0.046 1.000
```
Intermediate use II

twomoments computes first and second moments from data as madness:

```r
set.seed(12345)  # mental note: change gmail password
X <- matrix(rnorm(1000 * 2), ncol = 2)
two <- twomoments(X, diag.only = FALSE)
# symmetric square root of covariance:
show(cov2cor(vcov(sqrtm(two$Sigma))))

## [1,] 1.000 0.029 0.029 0.02
## [2,] 0.029 1.000 1.000 0.11
## [3,] 0.029 1.000 1.000 0.11
## [4,] 0.020 0.110 0.110 1.00

# Hotelling's statistic, up to scaling:
show(vcov(t(two$mu) %*% solve(two$Sigma, two$mu))[1])

## [1] 1.3e-05
```
library(dplyr)

ff3 <- Quandl("KFRENCH/FACTORS_W", order = "asc") %>%
  select(starts_with("Mkt"), SMB, HML)

# compute first and second moments:

two <- twomoments(ff3, diag.only = FALSE)
the_mp <- solve(two$Sigma, two$mu)

show(val(t(the_mp)))

## [,1] [,2] [,3]
## [1,] 0.013 0.013 0.022

show(vcov(the_mp))

## [,1] [,2] [,3]
## [1,] 3.0e-05 -4.9e-06 -1.4e-05
## [2,] -4.9e-06 9.9e-05 6.9e-07
## [3,] -1.4e-05 6.9e-07 7.5e-05
What’s next?

To learn more:

- Check the [github project page](#).
- Read the vignette.
- Comb through the unit tests.
- Ask me questions.

Thank you.
## Common Questions

### Why another AD package?
I needed it for the multivariate delta method, and suspected that numerical approximations would be too slow.

### Can it compute Hessians or higher order derivatives?
Not yet. That would be very complicated. You might be better off using the `numDeriv` package.

### How are derivatives computed?
I took a ‘high level’ approach, overloading as many methods as possible. A ‘low level’ approach that overloads only addition, multiplication, exponentiation, and so on, would not work for computations performed in e.g., C or C++ code.
Common Questions II

I am getting an error.
Not everything works yet. File an issue on github. In the interim, you can use the `madness::numderiv` method for numerical approximation, or instead use the `numDeriv` package.

What about symmetry?
The `twomoments` and `theta` methods deal with symmetry correctly. Be wary about constructing your own `madness` objects where the matrix must be symmetric. To ‘force’ matrix symmetry, use e.g.,

```r
forcesym <- function(x) { 0.5 * (x+t(x)) }
```