Implementation Checks on tflashr

David Gerard

2016-03-25

Abstract.

I run some implementation checks to make sure everything is working ok.

Compare `flashr` to `tflashr`.

The following is the same as the `flash_h1` function in `flashr` except I changed the starting values and convergence criteria to be what I use in `tflash` and I optimize the rows first, then the columns:

```
flash_r1_modified = function(Y, tol=1e-6, maxiter_r1 = 100){
  #dealing with missing value
  Y[is.na(Y)] = 0
  # get initial value for l and f and sigmae
  svd_y <- svd(Y)
  El = svd_y$u[,1]
  Ef <- svd_y$v[,1]
  d1 <- svd_y$d[1]
  El <- El * d1  ## put all mass on first mode
  El2 = El^2
  Ef2 = Ef^2

  #start iteration
  sigmae2_v = mean( Y^2 - 2*Y*(El %*% t(Ef)) + (El2 %*% t(Ef2)) )

  epsilon = 1
  tau = 2 # since already did one iteration
  while(epsilon >= tol & tau < maxiter_r1 ){
    tau = tau + 1
    pre_sigmae2 = sigmae2_v

    sigmae2_v = mean( Y^2 - 2*Y*(El %*% t(Ef)) + (El2 %*% t(Ef2)) )

    ## make rows go first
    par_l = flashr:::ATM_l(Y,Ef,Ef2,sigmae2_v)
    El = par_l$El
    El2 = par_l$El2
    if(sum(El^2)==0){
      El = rep(0,length(El))
      Ef = rep(0,length(Ef))
      break
    }
    sigmae2_v = mean( Y^2 - 2*Y*(El %*% t(Ef)) + (El2 %*% t(Ef2)) )
    ## update sigma after every mode.

    par_f = flashr:::ATM_f(Y,El,El2,sigmae2_v)
    Ef = par_f$Ef
    Ef2 = par_f$Ef2
    if(sum(Ef^2)==0){
      El = rep(0,length(El))
      Ef = rep(0,length(Ef))
      break
    }
    sigmae2_v = mean( Y^2 - 2*Y*(El %*% t(Ef)) + (El2 %*% t(Ef2)) )

    epsilon = abs(pre_sigmae2/sigmae2_v - 1)
  }
  return(list(l = El, f = Ef, sigmae2 = sigmae2_v))
}
```

First, I generate some rank 1 matrix data.

```
set.seed(555)
library(flashr)
n <- 50
p <- 50
Theta <- rnorm(n) %*% t(rnorm(p)) * 1
E <- matrix(rnorm(n * p), nrow = n)
Y <- Theta + E
```

Now I run `flashr` and `tflashr` under the same settings.

```
flash_out_m <- flash_r1_modified(Y = Y, tol = 10^-5, maxiter_r1 = 100)

fstart <- proc.time()
flash_out <- flash_r1(Y = Y, tol = 10^-5, maxiter_r1 = 100)
ftot <- proc.time() - fstart

tstart <- proc.time()
tflash_out <- tflash(Y = Y, tol = 10^-5, itermax = 100)
ttot <- proc.time() - tstart
```

These should all be the same:

```
cat("    From FLASH:", flash_out$sigmae2, "\n",
    "From Modified:", flash_out_m$sigmae2, "\n",
    "From  T-FLASH:", 1 / tflash_out$sigma_est)
```

```
##     From FLASH: 1.04
##  From Modified: 1.04
##  From  T-FLASH: 1.04
```

Each of the following should be constants.

```
flash_out$l / tflash_out$postMean[[1]]
```

```
##  [1] -0.1388 -0.1388 -0.1388 -0.1388 -0.1388 -0.1388 -0.1388 -0.1388
##  [9] -0.1388 -0.1388 -0.1388 -0.1388 -0.1388 -0.1388 -0.1388 -0.1388
## [17] -0.1388 -0.1388 -0.1388 -0.1388 -0.1388 -0.1388 -0.1388 -0.1388
## [25] -0.1388 -0.1388 -0.1388 -0.1388 -0.1388 -0.1388 -0.1388 -0.1388
## [33] -0.1388 -0.1388 -0.1388 -0.1388 -0.1388 -0.1388 -0.1388 -0.1388
## [41] -0.1388 -0.1388 -0.1388 -0.1388 -0.1388 -0.1388 -0.1388 -0.1388
## [49] -0.1388 -0.1388
```

```
flash_out$f / tflash_out$postMean[[2]]
```

```
##  [1] -7.179 -7.179 -7.179 -7.179 -7.179 -7.179 -7.179 -7.179 -7.179 -7.179
## [11] -7.179 -7.179 -7.179 -7.179 -7.179 -7.179 -7.179 -7.179 -7.179 -7.179
## [21] -7.179 -7.179 -7.179 -7.179 -7.179 -7.179 -7.179 -7.179 -7.179 -7.179
## [31] -7.179 -7.179 -7.179 -7.179 -7.179 -7.179 -7.179 -7.179 -7.179 -7.179
## [41] -7.179 -7.179 -7.179 -7.179 -7.179 -7.179 -7.179 -7.179 -7.179 -7.179
```

```
flash_out_m$l / tflash_out$postMean[[1]]
```

```
##  [1] -7.175 -7.175 -7.175 -7.175 -7.175 -7.175 -7.175 -7.175 -7.175 -7.175
## [11] -7.175 -7.175 -7.175 -7.175 -7.175 -7.175 -7.175 -7.175 -7.175 -7.175
## [21] -7.175 -7.175 -7.175 -7.175 -7.175 -7.175 -7.175 -7.175 -7.175 -7.175
## [31] -7.175 -7.175 -7.175 -7.175 -7.175 -7.175 -7.175 -7.175 -7.175 -7.175
## [41] -7.175 -7.175 -7.175 -7.175 -7.175 -7.175 -7.175 -7.175 -7.175 -7.175
```

```
flash_out_m$f / tflash_out$postMean[[2]]
```

```
##  [1] -0.1394 -0.1394 -0.1394 -0.1394 -0.1394 -0.1394 -0.1394 -0.1394
##  [9] -0.1394 -0.1394 -0.1394 -0.1394 -0.1394 -0.1394 -0.1394 -0.1394
## [17] -0.1394 -0.1394 -0.1394 -0.1394 -0.1394 -0.1394 -0.1394 -0.1394
## [25] -0.1394 -0.1394 -0.1394 -0.1394 -0.1394 -0.1394 -0.1394 -0.1394
## [33] -0.1394 -0.1394 -0.1394 -0.1394 -0.1394 -0.1394 -0.1394 -0.1394
## [41] -0.1394 -0.1394 -0.1394 -0.1394 -0.1394 -0.1394 -0.1394 -0.1394
## [49] -0.1394 -0.1394
```

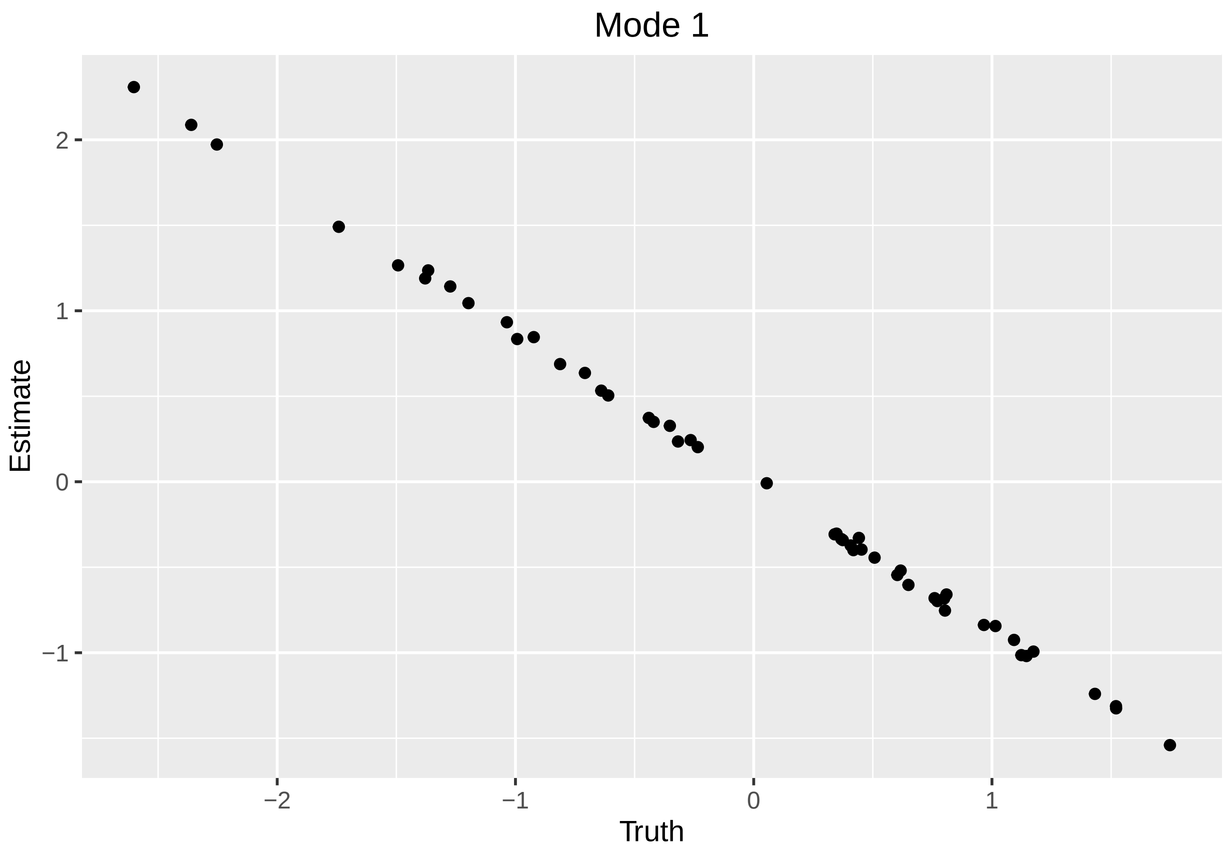
See performance on a tensor dataset.

This is just to see if I get any weird results from a tensor dataset. But everything looks good

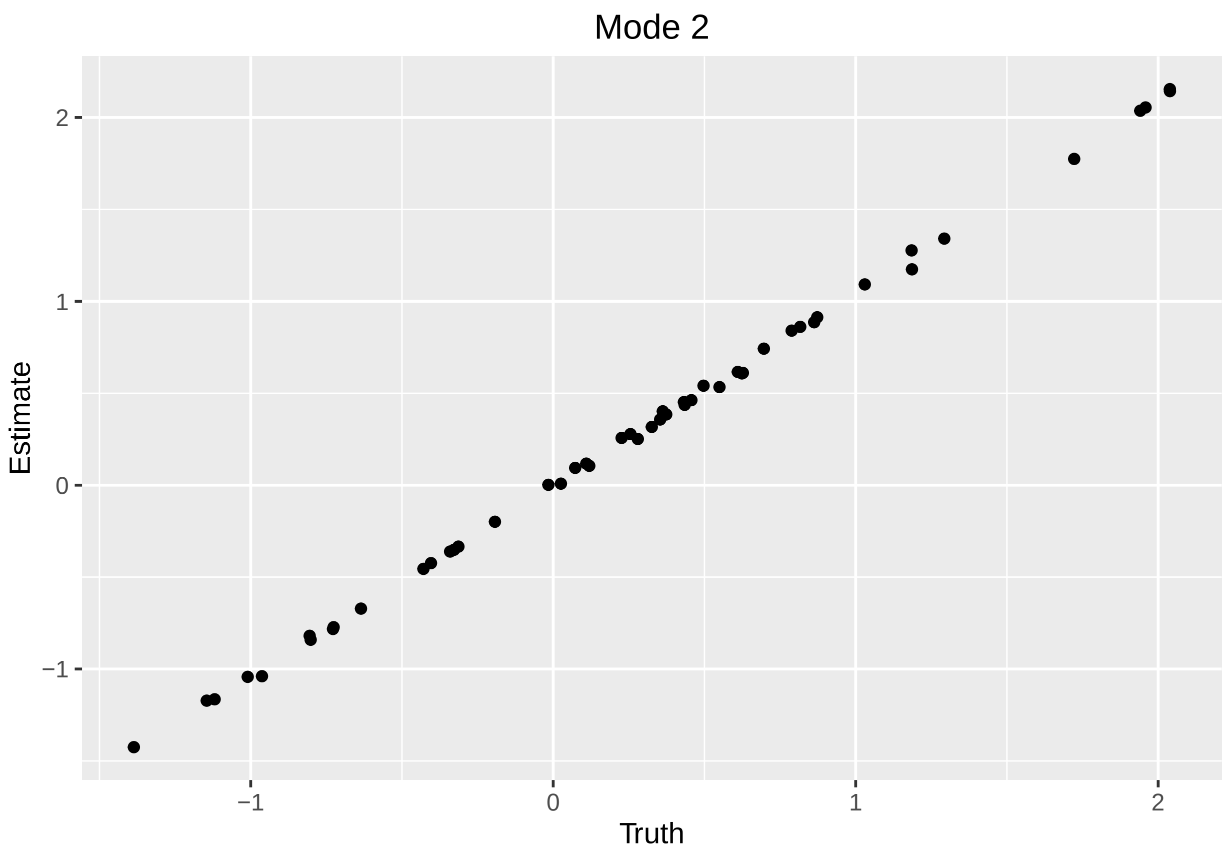
```
rm(list = ls())
p <- c(50, 50, 50)
u <- list()
u[[1]] <- rnorm(p[1])
u[[2]] <- rnorm(p[2])
u[[3]] <- rnorm(p[3])
Theta <- form_outer(u)
E <- array(rnorm(prod(p)), dim = p)
Y <- Theta + E
tflash_out <- tflash(Y)

cat("Estimate of Variance:", 1 / tflash_out$sigma_est)
```

```
## Estimate of Variance: 0.996
```

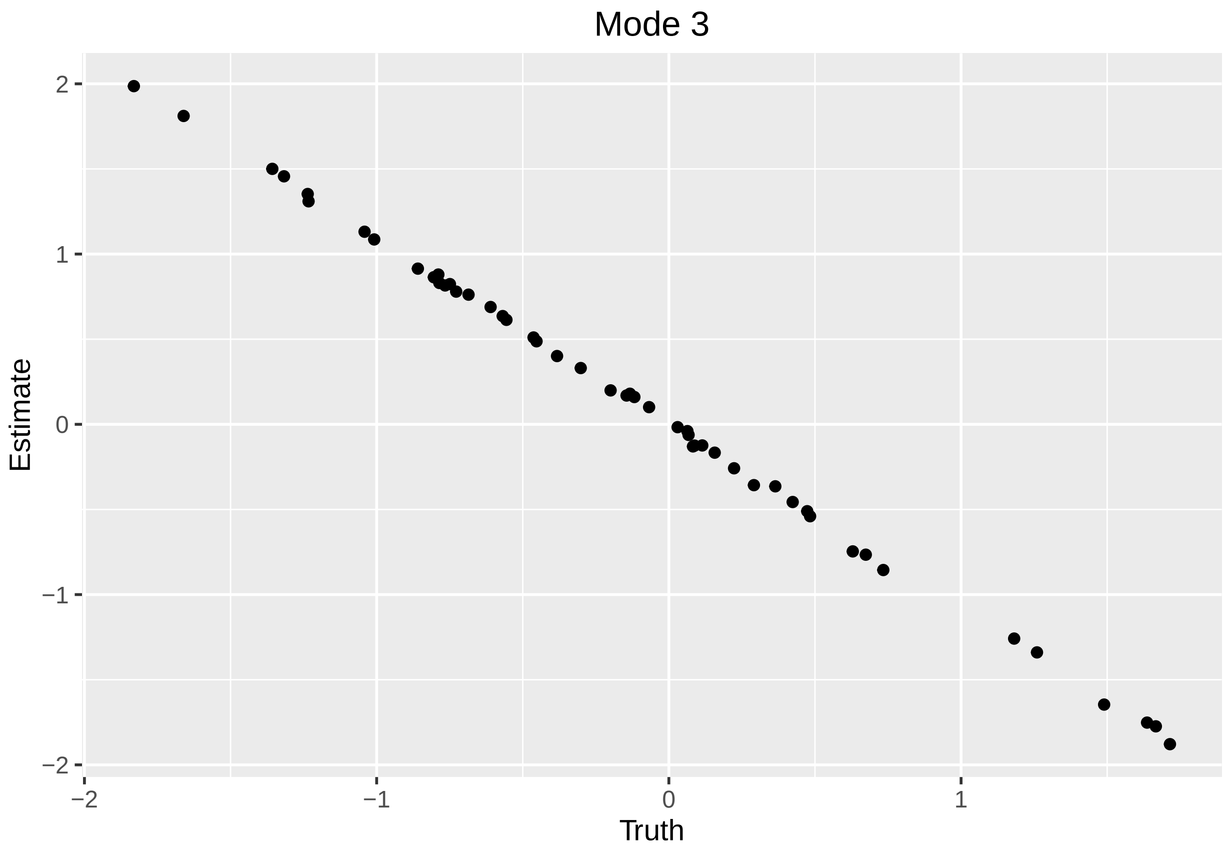
```r
ggplot2::qplot(u[[1]], tflash_out$postMean[[1]], xlab = "Truth", ylab = "Estimate",
               main = "Mode 1")
```



```r
ggplot2::qplot(u[[2]], tflash_out$postMean[[2]], xlab = "Truth", ylab = "Estimate",
               main = "Mode 2")
```



```r
ggplot2::qplot(u[[3]], tflash_out$postMean[[3]], xlab = "Truth", ylab = "Estimate",
               main = "Mode 3")
```



It’s OK if the slope is negative because of identifiability.