flashtpx: Simulation Run 4

Wei Wang, Kushal K Dey

April 1, 2016

Contents

Overview

This is the fourth simulation run of *flashtpx* without the non-negative constraint. The main idea is to try and replicate the results for the `FitGoM()` in CountClust or `topics()` model in the maptpx package due to Matt Taddy.

Without the non-negative constraint, *flashtpx* is basically applying *flash* with the covariance matrix for the data estimated from the GoM or topic model fitting on the counts data. Here we apply *flashtpx* on the counts data generated from a chosen simulation design and then interpret the results and compare the results to the PMA model fitting.

Simulation Design

We load the packages and the functions we need to perform the model.

```
library(ashr)
library(irlba)
library(PMA)
source("../R/flash.R")
```

We go back to $K = 2$.

Next we determine the `omega` matrix determined in batches of size 1000, the first batch of 1000 samples coming from one cluster completely, the next 1000 coming from a second cluster and the final 1000 being proportionally assigned to the two clusters. (Check barplot for more clarity).

```
n.out <- 1000
omega_sim <- rbind(cbind(rep(1, n.out), rep(0,
    n.out)), cbind(rep(0, n.out), rep(1, n.out)),
    cbind(seq(0.6, 0.4, length.out = n.out), 1 -
        seq(0.6, 0.4, length.out = n.out)))
dim(omega_sim)
```

```
## [1] 3000    2
```

```
K <- dim(omega_sim)[2]
```

Assume there are 1000 genes.

```
freq <- rbind(c(0.1, 0.2, rep(0.7/998, 998)),
    c(rep(0.7/998, 998), 0.1, 0.2))
str(freq)
```

```
##  num [1:2, 1:1000] 0.1 0.000701 0.2 0.000701 0.000701 ...
```

The counts data generated as follows

```
counts <- t(do.call(cbind, lapply(1:dim(omega_sim)[1],
    function(x) rmultinom(1, 1000, prob = omega_sim[x,
        ] %*% freq))))
dim(counts)
```

```
## [1] 3000 1000
```

We next fit a standard topic model with $K = 2$.

```
topic.fit <- maptpx::topics(counts, K = 2)
```

```
##
## Estimating on a 3000 document collection.
## Fitting the 2 topic model.
## log posterior increase: 33960.2, 6100.8, 48305.1, 0.5, done.
```

```
omega <- topic.fit$omega
theta <- topic.fit$theta
```

Brief methods overview

Under the standard topic model, we have

$$c_{ng} \sim Poi(c_{n+} \sum_{k=1}^{K} \omega_{nk} \theta_{kg})$$

Let us define

$$\lambda_{ng} = c_{n+} \sum_{k=1}^{K} \omega_{nk} \theta_{kg}$$

Under the normal model, if λ_{ng} is large, we can assume

$$c_{ng} \sim N(\lambda_{ng}, \lambda_{ng})$$

which is equivalent to saying

$$c_{ng} = \lambda_{ng} + e_{ng} \qquad e_{ng} \sim N(0, \lambda_{ng})$$

For applying *flash*, we first estimate the λ in the variance using topic model estimate (we call this λ to be $\lambda^{\star}$).

to guard against the odd possibility that for some n and g, this estimate $\lambda^{\star}$ could be 0, we replace such cases as of now with a small value 0.0001.

In this example model we have considered $c_{n+} = 1000$.

```
lambda_star <- 1000 * (omega %*% t(theta))
lambda_star[lambda_star == 0] <- 1e-04
dim(lambda_star)
```

```
## [1] 3000 1000
```

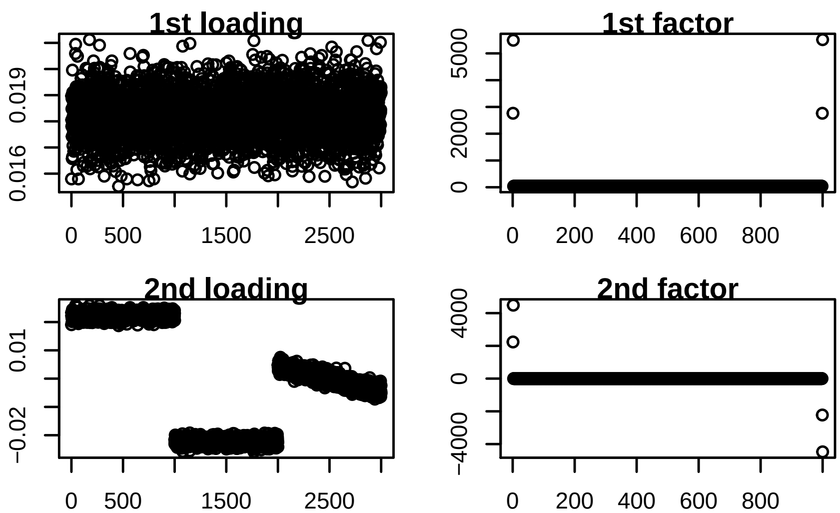
FLASH model fitting

flash then fits the model

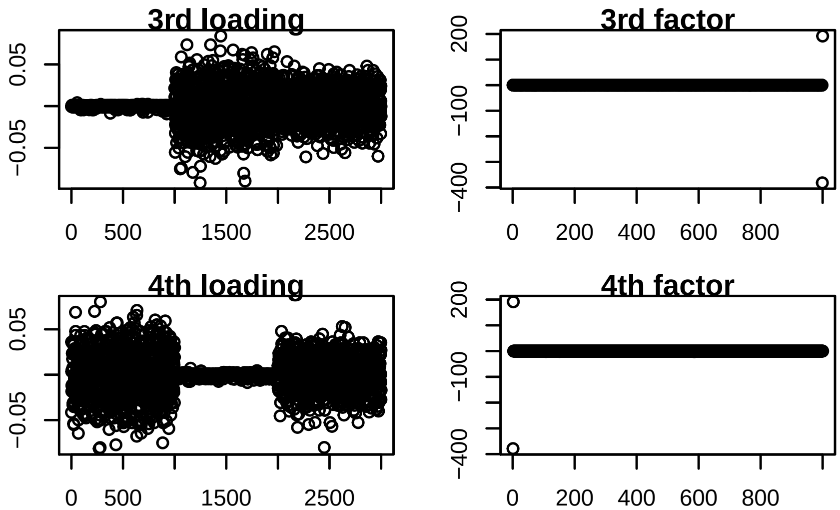
$$c_{ng} = \sum_{k=1}^{K} l_{nk} f_{kg} + e_{ng} \qquad e_{ng} \sim N(0, \lambda^{\star}_{ng})$$

```
g1 = suppressMessages(flash(counts, sigmae2_true = lambda_star))
f = g1$f
l = g1$l
res = counts - l %*% t(f)
# g_new = flash(res,nonnegative =
# TRUE,sigmae2_true = lambda)
g2 = suppressMessages(flash(res, sigmae2_true = lambda_star))
l = g2$l
f = g2$f
res = res - l %*% t(f)
g3 = suppressMessages(flash(res, sigmae2_true = lambda_star))
l = g3$l
f = g3$f
res = res - l %*% t(f)
```

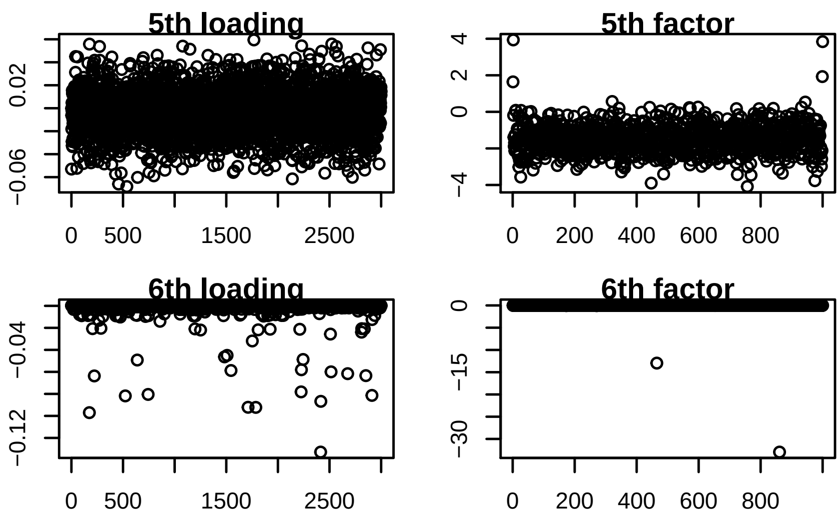
```r
g4 = suppressMessages(flash(res, sigmae2_true = lambda_star))
l = g4$l
f = g4$f
res = res - l %*% t(f)
g5 = suppressMessages(flash(res, sigmae2_true = lambda_star))
l = g5$l
f = g5$f
res = res - l %*% t(f)
g6 = suppressMessages(flash(res, sigmae2_true = lambda_star))
l = g6$l
f = g6$f
par(mfrow = c(2, 2))
par(cex = 0.6)
par(mar = c(3, 3, 0.8, 0.8), oma = c(1, 1, 1,
    1))
plot(g1$l, main = "1st loading")
plot(g1$f, main = "1st factor")
plot(g2$l, main = "2nd loading")
plot(g2$f, main = "2nd factor")
```




```r
plot(g3$l, main = "3rd loading")
plot(g3$f, main = "3rd factor")
plot(g4$l, main = "4th loading")
plot(g4$f, main = "4th factor")
```




```
plot(g5$l, main = "5th loading")
plot(g5$f, main = "5th factor")
plot(g6$l, main = "6th loading")
plot(g6$f, main = "6th factor")
```



PMA model fitting

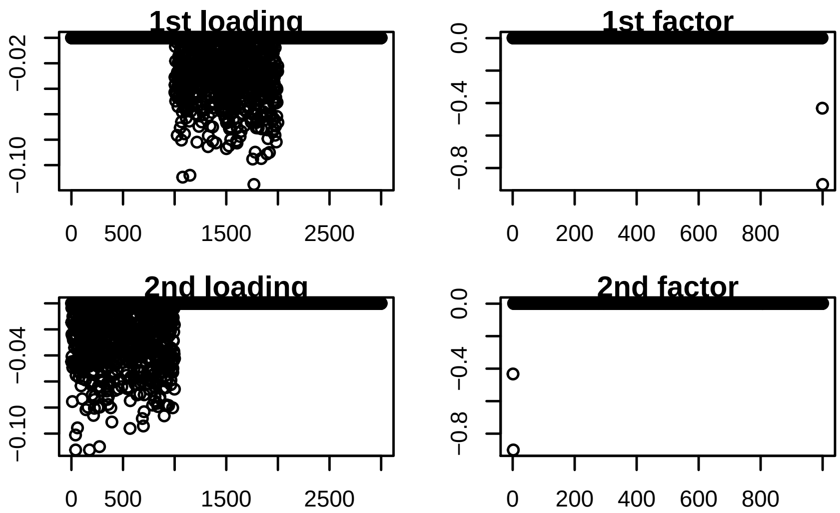
We compare the output of *flash* with that of the `PMD()` function of the package PMA. We use their default settings for shrinkage.

```
out <- PMD(counts, K = 6)
```

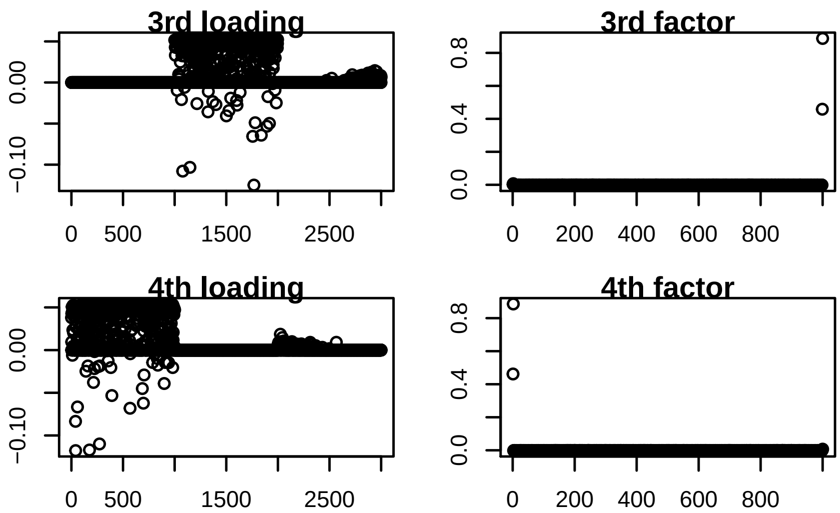
```
## 1234567
## 123456
## 1234567
## 12345678
```

```
## 12345678910111213
## 12345678910
```

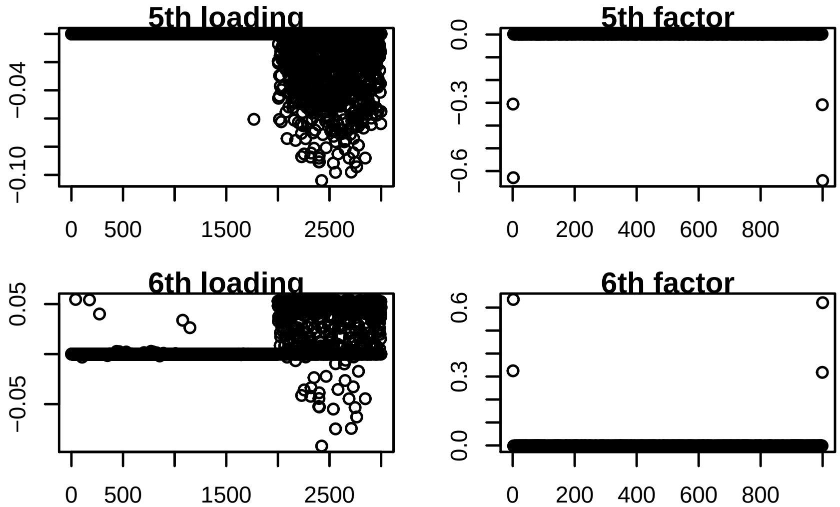
```r
par(mfrow = c(2, 2))
par(cex = 0.6)
par(mar = c(3, 3, 0.8, 0.8), oma = c(1, 1, 1,
    1))
plot(out$u[, 1], main = "1st loading")
plot(out$v[, 1], main = "1st factor")
plot(out$u[, 2], main = "2nd loading")
plot(out$v[, 2], main = "2nd factor")
```




```r
plot(out$u[, 3], main = "3rd loading")
plot(out$v[, 3], main = "3rd factor")
plot(out$u[, 4], main = "4th loading")
plot(out$v[, 4], main = "4th factor")
```



```r
plot(out$u[, 5], main = "5th loading")
plot(out$v[, 5], main = "5th factor")
plot(out$u[, 6], main = "6th loading")
plot(out$v[, 6], main = "6th factor")
```




```r
sessionInfo()
```

```
## R version 3.2.4 (2016-03-10)
## Platform: x86_64-apple-darwin13.4.0 (64-bit)
## Running under: OS X 10.10.5 (Yosemite)
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils
## [5] datasets  methods   base
##
## other attached packages:
## [1] Rcpp_0.12.4   PMA_1.0.9     impute_1.44.0
## [4] plyr_1.8.3    irlba_2.0.0   Matrix_1.2-4
## [7] ashr_1.0.8
##
## loaded via a namespace (and not attached):
##  [1] knitr_1.12.3      magrittr_1.5
##  [3] MASS_7.3-43       doParallel_1.0.10
##  [5] pscl_1.4.9        SQUAREM_2014.8-1
##  [7] lattice_0.20-33   foreach_1.4.3
##  [9] stringr_1.0.0     tools_3.2.4
## [11] parallel_3.2.4    grid_3.2.4
```

```
## [13] maptpx_1.9-2      htmltools_0.3
## [15] iterators_1.0.8   assertthat_0.1
## [17] yaml_2.1.13       digest_0.6.9
## [19] formatR_1.2.1     codetools_0.2-14
## [21] slam_0.1-32       evaluate_0.8
## [23] rmarkdown_0.9.2   stringi_1.0-1
## [25] truncnorm_1.0-7   tufte_0.2
```