# Harvest: Even More Simulations for the "Thresher" Paper

Kevin R. Coombes

7 January 2014

## Contents



## List of Figures

# 1 Executive Summary

## 1.1 Introduction

This report describes the (second) analysis of simulated data sets to test the behavior of our proposed methods for analyzing continuous pathway data.

### 1.1.1 Aims/Objectives

We want to see whether the methods can identify the correct number of protein clusters (which should be between 1 and 4 in our simulated datasets).

## 1.2 Methods

### 1.2.1 Description of the Data

In the previous report, we simulated and saved 2500 datasets with a few proteins (around 10–20) and many samples (median: 304, range: 126–506). Each dataset exhibits either one or two independent correlated signals. Each signal can be unsigned (all proteins are positively correlated, so a reasonable summary would be a simple average of all proteins) or signed (including both positively and negatively correlated proteins, so a reasonable summary requires looking at a difference between two group averages). Each dataset also contains two "noise" genes that are not correlated with any of the simulated signals.

### 1.2.2 Statistical Methods

We use the "Thresher" algorithm described in the previous report, with a cutoff $\Delta \leq 0.3$, to detect outliers or "noise" proteins. We use the Auer-Gervini approach to estimate the number $K$ of signfiicant principal components. We fit a mixture of von Mises - Fisher distributions to cluster the protein directions (on a unit sphere) into $N = K, K+1, \ldots, 2K+1$ protein groups. To select the optimal number of protein groups, we compute the Bayesian Information Criterion (BIC) for each $N$; the best number corresponds to the minimum BIC.

## 1.3 Results

- The estimated number of principal components is (a) always correct if the true dimension equals 1 and (b) is correct 94% of the time when the true dimension equals 2 (**Section 4.1**).
- When clustering in the space of principal component loadings, the estimated number of protein groups is correct 73% of the time (**Section 4.3**). If you only consider situations where the PC dimenion was correctly estimated, then the number of protein groups is correct 75% or the time.
- When clustering in the complete protein-sample space, the estimated number of protein groups is correct 86% of the time (**Section 4.4**). If you only consider situations where the PC dimenion was correctly estimated, then the number of protein groups is correct 89% or the time.
- After removing outliers and estimating the number of protien groups, the plots give a clearer idea of the true underlying structure. (For loadings, compare **Figure ??** to **Figure ??**. For heatmaps, compare **Figure ??** to **Figure ??**. For samples in principal component space, compare **Figure ??** to **Figure ??**.)

### 1.4 Conclusions

The Thresher-Reaper methods provide effective tools for removing outliers and detemineing the correct number of protein groups in (simulated) data sets containing about 10–20 proteins.

## 2 Preliminaries / Methods

### 2.1 Library Packages

We start by loading all of the R library packages that we need for this analysis.

```
> library(Thresher)
> library(RColorBrewer)   # for sensible color schemes
```

### 2.2 The Data Sets

Next, we load the simulated datasets from the first report.

```
> nSimSets <- 500
> f <- "moreSavedSims.rda"
> if (file.exists(f)) {
+   load (f)
+ } else {
+   set.seed(981079)
+   moreSavedSims <- list()
+   bsdim <- rep(NA, nSimSets)
+   sigProteins <- rep(NA, nSimSets)
+   for (idx in 1:nSimSets) {
+     cat(idx, "\n", file=stderr())
+ # parameters
+     NGROUPS <- 5
+     nProtein <- 2*sample(40:70,1)
+     split <- sample(1:NGROUPS, nProtein, replace=TRUE)
+     nNoise <- max(10, round(rnorm(1, nProtein/3, 8)))
+     positive <- sample(nProtein, nProtein/2)
+     negative <- (1:nProtein)[!((1:nProtein) %in% positive)]
+     signed <- rep(-1, length=nProtein)
+     signed[positive] <- 1
+ # unsigned
+     sigma1 <- matrix(0, ncol=nProtein, nrow=nProtein)
+     for (i in 1:NGROUPS) {
+       rho <- max(0.1, rnorm(1, 0.4, 0.15))
+       who <- split==i
+       sigma1[who,who] <- rho
```

```
+     }
+     diag(sigma1) <- 1
+ # signed
+     sigma3 <- sigma1
+     sigma3[positive, negative] <- -sigma3[positive, negative]
+     sigma3[negative, positive] <- -sigma3[negative, positive]
+ # reordered
+     os <- order(split, signed)
+     sigma4 <- sigma3[os,os]
+
+ # number of samples
+     nSample <- round(rnorm(1, 300, 60))
+ # add noise
+     ss <- matrix(0, nProtein+nNoise, nProtein+nNoise)
+     diag(ss) <- 1
+     ss[1:nProtein, 1:nProtein] <- sigma4
+ #    image(ss, col=blueyellow(64), zlim=c(-1,1))
+ # basic analysis
+     value <- SimThresher(ss, nSample,
+                          paste("newsim", idx, sep="."),
+                          method='auer.gervini')
+     moreSavedSims[[idx]] <- value
+     bsdim[idx] <- bsDimension(value@spca)
+     sigProteins[idx] <- nProtein
+   }
+   save(moreSavedSims, bsdim, sigProteins, file=f)
+ }
> rm(f)
>
```

now we test whether the broken-stick model or the Auer-Gervini approach does a better job of recovering the true number of principal components.

```
> pcdim <- unlist(lapply(moreSavedSims, function(x) x@pcdim))
> table(pcdim)

pcdim
  2   4   5   6
  1   2 496   1

> table(bsdim)

bsdim
  3   4   5
 21 102 377
```

Clearly, theh Auer-Gervini method works better: it gets the correct answer more than 99% of the time, while the broken stick model underestimates the number of components almost 25% of the time.

## 3 Three Examples

We run the following loop of code to create the five standard figures for several different sample datasets.

```
> if (!file.exists("SimFigs")) {
+   dir.create("SimFigs")
+   for (idx in 1:40) { # really, do not do 2500 of these ...
+     makeFigures(moreSavedSims[[idx]], DIR="SimFigs")
+   }
+ }
```

# 4 Finding Protein Groups

We first apply the `reaper` algorithm to the directions in PC space.

```
> f <- "moreVmfMixturesLoaded.rda"
> if(file.exists(f)) {
+   load(f)
+ } else {
+   set.seed(473643)
+   vmfMixturesLoaded <- lapply(moreSavedSims, Reaper, useLoadings=TRUE,
+                               method="auer.gervini")
+   save(vmfMixturesLoaded, file=f)
+ }
> rm(f)
```

Next, we apply the algorithm in the full protein-sample space.

```
> f <- "moreVmfMixtures.rda"
> if(file.exists(f)) {
+   load(f)
+ } else {
+   set.seed(521143)
+   vmfMixtures <- lapply(moreSavedSims, Reaper, useLoadings=FALSE,
+                         method="auer.gervini")
+   save(vmfMixtures, file=f)
+ }
> rm(f)
```

## 4.1 Number of Principal Components

Since both applications use the same code to determine the correct PC dimension, $K$, we want to see how this compares both to the value before removing outliers. and to the true value.

```
> pcDimension <- sapply(vmfMixtures, function(x) x@pcdim)
> table(pcDimension, pcdim)
```

```
            pcdim
pcDimension   2   4   5   6
          2   1   0   0   0
          4   0   2   0   0
          5   0   0 496   0
          6   0   0   0   1
```

None of the 2500 simulated datasets have the estimated dimension changed when removing outliers. This finding is not terribly surprising, since we saw in the previous report that the main explanation of the failure to find the correct dimension was attributable to few signal proteins or few samples, neither of which has anything to do with the outliers.

### 4.2 Outlier Detection

```
> temp <- as.data.frame(t(sapply(1:length(vmfMixtures), function(idx) {
+   found <- vmfMixtures[[idx]]@keep
+   truth <- rep(FALSE, length(found))
+   truth[1:sigProteins[idx]] <- TRUE
+   c(TP=sum(truth&found),
+     FP=sum(!truth&found),
+     FN=sum(truth&!found),
+     TN=sum(!truth&!found),
+     sens=sum(truth&found)/sum(truth),
+     spec=sum(!truth&!found)/sum(!truth))
+ })))
> summary(temp)
```

```
       TP             FP               FN               TN             sens
 Min.   : 50   Min.   : 0.000   Min.   : 0.000   Min.   : 9.0   Min.   :0.4545
 1st Qu.: 94   1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.:30.0   1st Qu.:1.0000
 Median :110   Median : 0.000   Median : 0.000   Median :36.0   Median :1.0000
 Mean   :110   Mean   : 0.206   Mean   : 0.598   Mean   :36.5   Mean   :0.9947
 3rd Qu.:126   3rd Qu.: 0.000   3rd Qu.: 0.000   3rd Qu.:44.0   3rd Qu.:1.0000
 Max.   :140   Max.   :11.000   Max.   :60.000   Max.   :63.0   Max.   :1.0000
      spec
 Min.   :0.5769
 1st Qu.:1.0000
 Median :1.0000
 Mean   :0.9938
 3rd Qu.:1.0000
 Max.   :1.0000
```

```
> plot(sort(temp$sens))
> plot(sort(temp$spec))
> plot(1-temp$spec, temp$sens)
> which(temp$sens < 0.9)

[1] 188 263 475

> mean(temp$spec == 1 & temp$sens==1)

[1] 0.766

> mean(temp$sens < 0.9 | temp$spec < 0.9)

[1] 0.014
```

```
> odd <- which(temp$sens < 0.9 | temp$spec < 0.9)
> temp[odd,]

     TP FP FN TN      sens      spec
7   127  5  5 41 0.9621212 0.8913043
18   92 11  0 15 1.0000000 0.5769231
83  102  2  2 17 0.9807692 0.8947368
188  50  0 60 33 0.4545455 1.0000000
263  84  0 12 39 0.8750000 1.0000000
369  92  2  0 17 1.0000000 0.8947368
475  74  0 10 32 0.8809524 1.0000000

> nrho <- sapply(moreSavedSims, function(x) length(x@rho))
> bod <- which(nrho<5)
> rose <- t(sapply(moreSavedSims, function(x) {
+   fog <- x@rho
+   if (length(fog) < 5) fog <- c(0.1, fog)
+   fog
+ }))
> rose[bod,]

     [,1] [,2]      [,3]      [,4]      [,5]
[1,]  0.1  0.1 0.2379719 0.5138677 0.6213471
[2,]  0.1  0.1 0.3017474 0.4464947 0.5366649
[3,]  0.1  0.1 0.3121899 0.3180293 0.4677051
[4,]  0.1  0.1 0.1988509 0.2235231 0.4948387
[5,]  0.1  0.1 0.3705988 0.4676935 0.6511945
[6,]  0.1  0.1 0.3197944 0.4215384 0.5039410

> rose[odd,]

          [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 0.1000000 0.1294872 0.3760799 0.3896534 0.6296539
[2,] 0.2864932 0.3219822 0.3788453 0.5009299 0.5574004
[3,] 0.1287035 0.1645752 0.3088101 0.3252078 0.3979862
[4,] 0.1000000 0.1621900 0.2086104 0.2870825 0.6588241
[5,] 0.1000000 0.1928217 0.2747828 0.6031629 0.6160979
[6,] 0.1667407 0.2360938 0.5334244 0.5592960 0.6783369
[7,] 0.1000000 0.2164886 0.3838144 0.3851067 0.4325819

>
>
```

### 4.3 Number of Protein Groups: PC Loadings

Now we explore how often clustering the proteins (using a mixture of von Mises - Fisher distributions) in principal component space gets the correct number of protein groups.

```
> ngL <- sapply(vmfMixturesLoaded, function(x) x@nGroups)
> table(ngL)
```

```
ngL
  5   8   9  10  11  12
  1   2   1 403  92   1
```

### 4.4 Number of Protein Groups: Protein-Sample Space

The alternative method performs the clustering in the full protein-sample space, not just in the truncated principal component space. The overall perfomance clearly looks better:

```
> ng <- sapply(vmfMixtures, function(x) x@nGroups)
> table(ng)
```

```
ng
  4   5   6   7   8   9  10  11
  1  43  26  43  55  78 217  37
```

```
> table(ng, ngL)
```

```
    ngL
ng    5   8   9  10  11  12
  4   1   0   0   0   0   0
  5   0   0   0  39   4   0
  6   0   0   0  21   5   0
  7   0   2   0  34   7   0
  8   0   0   0  42  13   0
  9   0   0   0  54  23   1
  10  0   0   1 183  33   0
  11  0   0   0  30   7   0
```

```
> bodkins <- which(ng==5 & ngL==10)
> summary(rose[bodkins,])
```

```
       V1                V2                V3                V4                V5
 Min.   :0.1000   Min.   :0.1000   Min.   :0.1932   Min.   :0.2235   Min.   :0.3213
 1st Qu.:0.1010   1st Qu.:0.2038   1st Qu.:0.2759   1st Qu.:0.3285   1st Qu.:0.4012
 Median :0.1397   Median :0.2277   Median :0.3108   Median :0.3840   Median :0.4877
 Mean   :0.1528   Mean   :0.2349   Mean   :0.3113   Mean   :0.3769   Mean   :0.4885
 3rd Qu.:0.2024   3rd Qu.:0.2753   3rd Qu.:0.3535   3rd Qu.:0.4119   3rd Qu.:0.5561
 Max.   :0.2467   Max.   :0.3390   Max.   :0.4074   Max.   :0.5405   Max.   :0.7424
```

```
> summary(temp[bodkins,])
```

```
       TP               FP               FN              TN             sens
 Min.   : 79.00   Min.   :0.0000   Min.   :0.000   Min.   :13.00   Min.   :0.9397
 1st Qu.: 86.00   1st Qu.:0.0000   1st Qu.:0.000   1st Qu.:28.00   1st Qu.:0.9852
 Median : 95.00   Median :0.0000   Median :0.000   Median :32.00   Median :1.0000
 Mean   : 98.23   Mean   :0.2564   Mean   :1.103   Mean   :33.33   Mean   :0.9900
 3rd Qu.:103.50   3rd Qu.:0.0000   3rd Qu.:1.500   3rd Qu.:37.00   3rd Qu.:1.0000
 Max.   :140.00   Max.   :3.0000   Max.   :7.000   Max.   :56.00   Max.   :1.0000
      spec
 Min.   :0.8947
 1st Qu.:1.0000
 Median :1.0000
 Mean   :0.9910
 3rd Qu.:1.0000
 Max.   :1.0000
```

```
> b <- bodkins[1]
```

```
> if (!file.exists("MoreSimFigs")) {
+   dir.create("MoreSimFigs")
+   for (idx in 1:50) {
+     makeFigures(vmfMixturesLoaded[[idx]], DIR="MoreSimFigs")
+   }
+ }
```

# 5 Appendix

This analysis was run in the following directory:

```
> getwd()
```

```
[1] "d:/Work/Reaper/Manuscript"
```

This analysis was run in the following software environment:

```
> sessionInfo()
```

```
R version 3.0.0 (2013-04-03)
Platform: x86_64-w64-mingw32/x64 (64-bit)

locale:
[1] LC_COLLATE=English_United States.1252  LC_CTYPE=English_United States.1252
[3] LC_MONETARY=English_United States.1252 LC_NUMERIC=C
```

```
[5] LC_TIME=English_United States.1252

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
 [1] RColorBrewer_1.0-5   Thresher_0.9.3       ClassDiscovery_3.0.0 oompaBase_3.0.1
 [5] mclust_4.0           cluster_1.14.4       ade4_1.5-2           movMF_0.1-2
 [9] colorspace_1.2-4     MASS_7.3-26

loaded via a namespace (and not attached):
[1] tools_3.0.0
```