Factor analysis with Adaptive Shrinkage using *flashr*

Wei Wang, Kushal K Dey & Matthew Stephens

Stephens Lab, The University of Chicago

*Correspondending Email: mstephens@uchicago.edu

August 24, 2016

Abstract

The *R* package *flashr* provides tools to perform factor analysis with adaptive shrinkage on the factor loadings and the factors and also provides means to visualization of the factor analysis results. The adaptive shrinkage is performed using the **ashr** package due to Stephens (2016).

The package provides generic functions to visualize loadings data and post processing functions to analyze the factors estimated with focus on sparsity and the proportion of variance in the data explained by each factor. It also provides a list of features that play the key role in distinguishing the factors.

flashr version: 0.1.1 [1]

[1] This document used the vignette from *Bioconductor* package *DESeq2, CountClust* as *knitr* template

Contents

1 Introduction

FLASH (Factor Loadings with Adaptive Shrinkage) is an extension of the adaptive shrinkage methods in **ashr** package due to Stephens (2016) to the domain of factor analysis. An important consideration in any factor analysis scheme are shrinkage and sparsity. There are many algorithms that perform Sparse Factor Analysis (check Engelhardt and Stephens), however determining the level of shrinkage is a challenging task for the user. FLASH solves this problem by adaptively selecting the level of shrinkage for factor loadings and factors.

flashr offers 3 versions of FLASH (*normal*, *greedy* and *backfitting*) to perform factor analysis with adaptive shrinkage. Also, it provides generic visualization tools to view and analyze the factor loadings along with post processing tools to check the proportion of variance explained and sparsity level of the different factors. Finally, it offers functions to select a list of features that drive the factors or play the most key role in distinguishing the features.

2 flashr Installation

flashr requires the packages *ashr*, *ggplot2*, *irlba*, *CountClust*, *cowplot*, *RColorBrewer*, *grid*, *gridExtra*. Also one needs the packages *devtools*to install the developmental version of the *flashr* which will be the most up-to-date version.

We install the Github version of the package.

```
devtools::install_github("kkdey/flashr")
```

```
library(flashr)
```

3 Data preparation

We install a single cell RNA-seq data across mouse embryo developmental stages due to Deng *et al* 2014 [**?**]. We load it as an ExpressionSet object.`singleCellRNASeqMouseDeng2014` data package due to Deng *et al* is a processed version of the data publicly available at Gene Expression Omnibus (GEO:GSE45719: see http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE45719).

```
library(devtools)

read.data1 = function() {
  x = tempfile()
  download.file('https://cdn.rawgit.com/kkdey/singleCellRNASeqMouseDeng2014/master/data/De
```

```
  z = get(load((x)))
  return(z)
}

Deng2014MouseESC <- read.data1()

## Alternatively
# install_github('kkdey/singleCellRNASeqMouseDeng2014')
```

We load the data

```
deng.counts <- Biobase::exprs(Deng2014MouseESC)
deng.meta_data <- Biobase::pData(Deng2014MouseESC)
deng.gene_names <- rownames(deng.counts)

voom_out <- limma::voom(deng.counts);
voom_weights <- t(voom_out$weights);
voom_data <- t(voom_out$E);
```

We apply FLASH on the data. We show here example of two versions of FLASH - the normal pooled FLASH on successive residuals and the greedy FLASH. We assume the number of factors to be $K = 10$.

```
ll <- flash.greedy(voom_data, K=10, flash_para = list(tol=1e-3, maxiter_r1 = 50,
                partype="known", sigmae2_true = voom_weights,
                nonnegative=FALSE));
ll <- flashpool(voom_data, K=10, tol=1e-3, maxiter_r1 = 50,
                partype="known", sigmae2_true = voom_weights);
```

We load the FLASH output from the greedy implementation of FLASH and then we process the output.

```
library(flashr)
ll_deng <- get(data("flash_deng_ex"))
```

We postprocess the loadings and the factors from the FLASH output. We calculate the percentage of variance explained by each factor.

```
postprocess_ll <- flash_factor_postprocess(ll_deng$l,ll_deng$f, voom_data)
pve_percentage <- c("", paste0(":PVE-", round(postprocess_ll$PVE*100,0), "%"))
```

We represent the loadings by a stacked barchart of the loadings obtained from the FLASH output. We keep aside the first factor as that represents the mean factor.

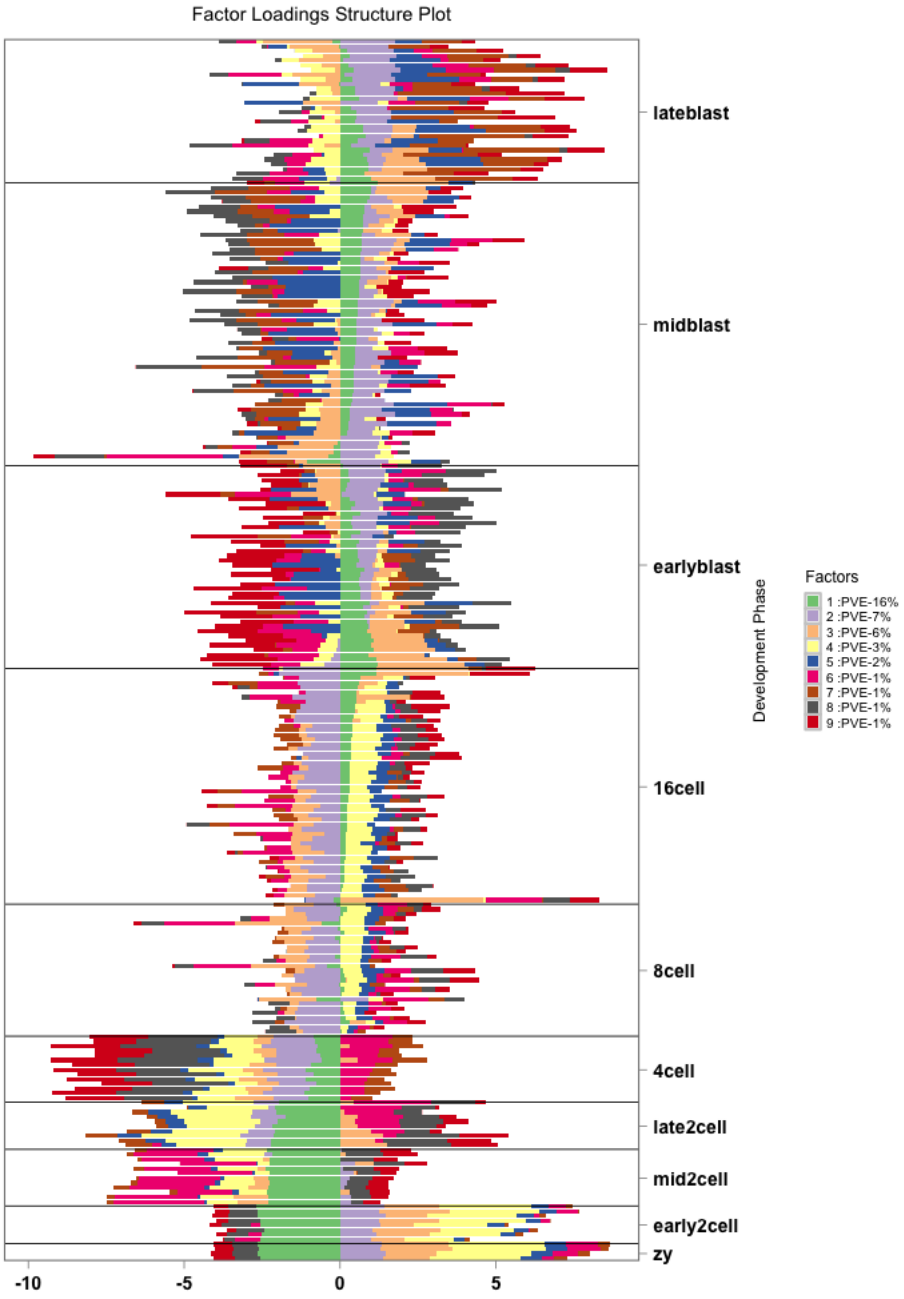
```
omega <- ll_deng$l

annotation <- data.frame(
  sample_id = paste0("X", c(1:NROW(omega))),
  label = factor(deng.meta_data$cell_type,
                        levels = c("zy", "early2cell",
```

```
"mid2cell", "late2cell","4cell", "8cell", "16cell","earlyblast","midblast","lateblast") ) )

rownames(omega) <- annotation$sample_id

FactorGGStack(loadings = omega[,-1],
                annotation = annotation,
                palette = c(RColorBrewer::brewer.pal(8, "Accent"),RColorBrewer::brewer.pal
                yaxis_label = "Development Phase",
                order_sample = TRUE,
                figure_title = "Factor Loadings Structure Plot",
                legend_labels = pve_percentage[-1],
                scale=TRUE,
                axis_tick = list(axis_ticks_length = .1,
                                 axis_ticks_lwd_y = .1,
                                 axis_ticks_lwd_x = .1,
                                 axis_label_size = 7,
                                 axis_label_face = "bold"))
```



We next look at a multi-grid bar chart representation of the factor loadings.

```
FactorGGBar(loadings = omega,
            annotation = annotation,
            palette = list("mid"="white",
                           "low"="red",
                           "high"="blue",
                           "midpoint"=0),
            yaxis_label = "Population Type",
            figure_title = " ",
```

```
            axis_tick = list(axis_ticks_length = .1,
                             axis_ticks_lwd_y = .1,
                             axis_ticks_lwd_x = .1,
                             axis_label_size = 7,
                             axis_label_face = "bold"),
            legend_labels=pve_percentage,
            scale=TRUE,
            panel=list(panel_rows=2,
                       panel_title="Factor Loadings Bar plot",
                       panel_title_fontsize=10,
                       panel_title_font=3))
```