SCPattern: a statistical approach to identify and classify expression changes in single cell RNA-seq experiments with ordered conditions

Ning Leng , Li-Fang Chu, Jeea Choi, Christina Kendziorski, James Thomson, and Ron Stewart

December 16, 2015

Contents

1 Introduction

SCPattern (as detailed in Leng* and Chu* *et al.*, 2015 (1)) is an empirical Bayes approach to characterize expression changes in single cell RNA-seq (scRNA-seq) experiments with ordered conditions, such as time points, spacial course, etc. SCPattern identifies genes with expression changes by considering zeros and non-zero cells collectively, and classifies them into directional expression patterns (e.g. Up-Up-Up-Up, Up-Up-Down-Down, etc). SCPattern tests distribution changes of a gene across each pair of adjacent conditions using directional Kolmogorov-Smirnov (K-S) statistic, and then classify the gene into expression patterns with probability estimates.

2 Run SCPattern

Before analysis can proceed, the SCPattern package must be loaded into the working space:

```
> library(SCPattern)
```

2.1 Required inputs

Data: The object `Data` should be a $G - by - S$ matrix containing the expression values for each gene and each cell, where G is the number of genes and S is the number of cells. These values should exhibit estimates of gene expression across cells. Counts of this nature may be obtained from RSEM (2), Cufflinks (3), or a similar approach. Cross-cell

library size normalization should be performed. A cross-cell library size normalization by median normalization are shown in section 2.2.

Conditions: The object `Conditions` should be a factor of length S that indicates to which condition each cell belongs. Note the order of levels in the factor should represent the order in the RNA-seq experiments.

The object `SCPatternExData` is a simulated data matrix containing 1000 rows of genes and 300 columns of cells. The genes are named `g1, g2, ...` and the cells are named `S1, S2, ...`

```
> data(SCPatternExData)
> str(SCPatternExData)
```

```
 num [1:1000, 1:300] 558 21 54 80 0 420 0 0 0 68 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:1000] "Gene_1" "Gene_2" "Gene_3" "Gene_4" ...
  ..$ : chr [1:300] "S1" "S2" "S3" "S4" ...
```

Here we simulated 60 cells for 5 time points (conditions). To specify which condition each cell belongs, we define:

```
> CondVector <- rep(paste("t",1:5,sep=""),each=60)
> str(CondVector)
```

```
 chr [1:300] "t1" "t1" "t1" "t1" "t1" "t1" "t1" "t1" "t1" "t1" ...
```

Downstream analysis by SCPattern requires the conditions to be specified as a factor. In particular, levels of the factor need to be sorted along the time/spatial course. For example, to generate a factor with ordered conditions from t1 to t5, we define:

```
> Conditions <- factor(CondVector, levels=c("t1","t2","t3","t4","t5"))
> str(Conditions)
```

```
 Factor w/ 5 levels "t1","t2","t3",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
> levels(Conditions)
```

```
[1] "t1" "t2" "t3" "t4" "t5"
```

2.2 Normalization

SCPattern requires cross-cell normalization to be applied to adjust for sequencing depth differences among different cells. Here, the library size factors may be obtained via the function `MedianNorm`, which implements the median-by-ratio normalization introduced in DESeq (4).

```
> Sizes <- MedianNorm(SCPatternExData)
> str(Sizes)
```

```
 Named num [1:300] NA NA NA NA NA NA NA NA NA NA ...
 - attr(*, "names")= chr [1:300] "S1" "S2" "S3" "S4" ...
```

As shown here, in a case that none of the genes are expressed in all cells (any gene has at least one zero counts) due to technical dropouts, the `MedianNorm()` function may return `NA` estimates. The option `alternative = TRUE` in `MedianNorm` function may be applied to address this issue. For example,

```
> Sizes <- MedianNorm(SCPatternExData,alternative = TRUE)
> str(Sizes)
```

```
 num [1:300] 0.999 1.001 0.992 0.993 0.988 ...
```

To obtain the normalized expression matrix for visualization purpose or other downstream analyses, we may use the `GetNormalizedMat()` function:

```
> DataNorm <- GetNormalizedMat(SCPatternExData, Sizes)
```

Note the SCPattern testing function requires raw expression estimates.

2.3 Classify genes into directional patterns

In a data set with multiple conditions, SCPattern calculates gene-specific posterior probability (PP) of being each expression pattern. For a given gene, higher $PP(\text{Pattern}_k)$ indicates that the gene is more likely to follow pattern k. The most likely pattern (MLP) of gene g is defined as $\text{argmax}_k PP(\text{Pattern}_k)$. To test for differentially expressed (DE) genes, function `SCPTest()` can be used:

```
> res.multi <- SCPTest(SCPatternExData, CondVector, Sizes)
```

The DE genes can be obtained from:

```
> head(res.multi$sortedlist)
```

```
         PP_marginal Path
Gene_977 "0.886"     "Up-Up-Down-Down"
Gene_143 "0.873"     "Up-Up-Down-Down"
Gene_379 "0.848"     "Up-Up-Down-Down"
Gene_233 "0.841"     "Down-Down-Up-Up"
Gene_740 "0.826"     "Up-Up-Down-Down"
Gene_853 "0.825"     "Down-Down-Up-Up"
```

```
> str(res.multi$sortedlist)
```

```
 chr [1:304, 1:2] "0.886" "0.873" "0.848" "0.841" "0.826" "0.825" ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:304] "Gene_977" "Gene_143" "Gene_379" "Gene_233" ...
  ..$ : chr [1:2] "PP_marginal" "Path"
```

The output `res.multi$sortedlist` is a matrix containing the DE genes. The DE genes are defined as the ones whose MLP is not constituted by EE (equally expressed, no change across any time points). The first column here shows PP(MLP) for each DE gene, and the second column shows genes' MLPs. The higher the PP(MLP) is, the more likely that this gene is following this pattern. Note the SCPTest() function requires the raw expression estimates.

In addition, estimates of PP(EE) and PP of all possible patterns can be found in `res.multi$EEPP` and `res.multi$PP.all`, respectively:

```
> head(res.multi$EEPP)
```

```
      Gene_1       Gene_2       Gene_3       Gene_4       Gene_5       Gene_6
2.413767e-06 5.253956e-01 2.028488e-01 2.455857e-01 5.638740e-02 6.259540e-01
```

```
> str(res.multi$PP.all)
```

```
 num [1:1000, 1:81] 2.41e-06 5.25e-01 2.03e-01 2.46e-01 5.64e-02 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:1000] "Gene_1" "Gene_2" "Gene_3" "Gene_4" ...
  ..$ : chr [1:81] "EE-EE-EE-EE" "Down-EE-EE-EE" "Up-EE-EE-EE" "EE-Down-EE-EE" ...
```

2.4 Visualization

Function `VioFun()` may be used to visualize the genes of interest. The `VioFun()` function produces side-by-side violin plots (where the curves represent a smoothed kernel density estimate). Each condition is represented by one viloin plot. For example, to visualize the top 6 DE identified genes from section 2.3:

```
> par(mfrow=c(2,3))
> for(i in 1:6) VioFun(rownames(res.multi$sortedlist)[i], DataNorm, Conditions)
```

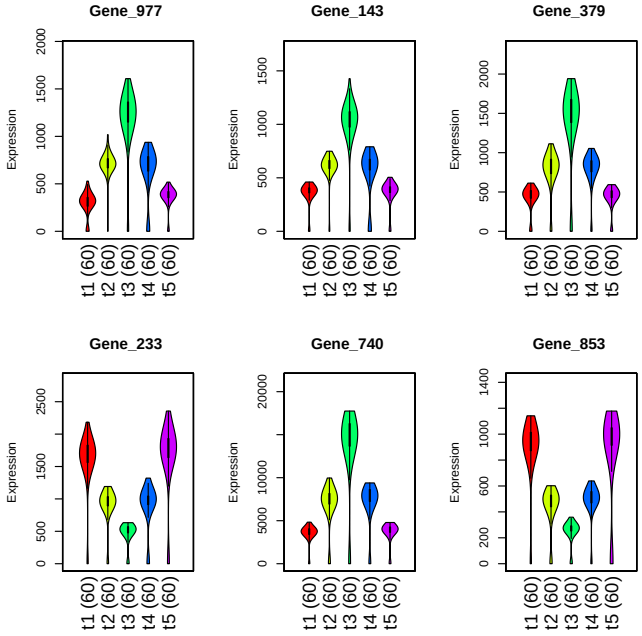

Figure 1: Top genes identified by SCPattern. The y axis shows normalized expression. The x axis shows conditions.

2.5 Include the 'Both' category

The SCPattern implementation also allows users to classify genes to category 'Both' in addition to Up, Down, and EE. For a given gene, the category 'Both' represents the case that some of the cells have increased expression from t to $t+1$ while the others decrease.

To include the 'Both' category, we may specify `NumPat = 4` in the `SCPTest()` function:

```
> res.multi.4pat <- SCPTest(SCPatternExData, CondVector, Sizes, NumPat=4)
```

Then the DE genes can be obtained from:

```
> head(res.multi.4pat$sortedlist)
```

```
         PP_marginal Path
Gene_233 "0.718"     "Down-Down-Up-Up"
Gene_715 "0.704"     "Up-Up-Down-Down"
Gene_379 "0.691"     "Up-Up-Down-Down"
Gene_267 "0.687"     "Up-Up-Down-Down"
Gene_391 "0.672"     "Up-Up-Down-Down"
Gene_740 "0.664"     "Up-Up-Down-Down"
```

```
> str(res.multi.4pat$sortedlist)
```

```
 chr [1:304, 1:2] "0.718" "0.704" "0.691" "0.687" "0.672" "0.664" ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:304] "Gene_233" "Gene_715" "Gene_379" "Gene_267" ...
  ..$ : chr [1:2] "PP_marginal" "Path"
```

```
> DE.4pat <- res.multi.4pat$sortedlist
```

Similarly as in section 2.3, PP(EE) and the posterior probability matrix could be obtained from:

```
> head(res.multi.4pat$EEPP)
```

```
      Gene_1       Gene_2       Gene_3       Gene_4       Gene_5       Gene_6
1.989691e-06 5.047266e-01 1.744149e-01 1.724865e-01 5.159997e-02 6.071591e-01
```

```
> str(res.multi.4pat$PP.all)
```

```
 num [1:1000, 1:256] 1.99e-06 5.05e-01 1.74e-01 1.72e-01 5.16e-02 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:1000] "Gene_1" "Gene_2" "Gene_3" "Gene_4" ...
  ..$ : chr [1:256] "EE-EE-EE-EE" "Down-EE-EE-EE" "Up-EE-EE-EE" "Both-EE-EE-EE" ...
```

Take a look at the top genes that have been classified into a pattern with 'Both':

```
> both.index <- grep("Both",DE.4pat[,2])
> DE.4pat.top6 <- DE.4pat[both.index[1:6],]
> print(DE.4pat.top6)
         PP_marginal Path
Gene_350 "0.367"     "Down-Down-Both-Down"
Gene_261 "0.354"     "Down-Up-Down-Both"
Gene_141 "0.352"     "Both-Up-Down-Down"
Gene_258 "0.337"     "Down-Down-Up-Both"
Gene_711 "0.321"     "Up-Up-Down-Both"
Gene_528 "0.3"       "Down-Down-Both-Down"
> par(mfrow=c(2,3))
> for(i in 1:6) VioFun(rownames(DE.4pat.top6)[i], DataNorm, Conditions)
```

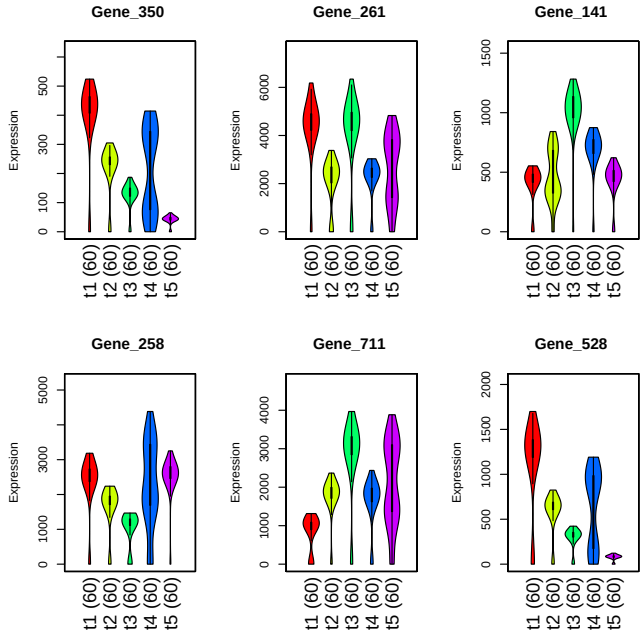

Figure 2: Top genes that have been classified into a pattern with 'Both'. The y axis shows normalized expression. The x axis shows conditions.

2.6 Non-directional tests

While the direction of change is not of interest, a user may classify genes into DE vs. EE for each pairwise comparison. To do so, we may specify `Directional = FALSE` in the `SCPTest()` function:

```
> res.multi.nd <- SCPTest(SCPatternExData, CondVector, Sizes, Directional=FALSE)
```

The DE genes can be obtained from:

```
> head(res.multi.nd$sortedlist)
```

```
         PP_marginal Path
Gene_977 "0.604"     "DE-DE-DE-DE"
Gene_143 "0.582"     "DE-DE-DE-DE"
Gene_379 "0.551"     "DE-DE-DE-DE"
Gene_233 "0.544"     "DE-DE-DE-DE"
Gene_715 "0.537"     "DE-DE-DE-DE"
Gene_45  "0.527"     "DE-DE-DE-DE"
```

```
> str(res.multi.nd$sortedlist)
```

```
 chr [1:301, 1:2] "0.604" "0.582" "0.551" "0.544" "0.537" "0.527" ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:301] "Gene_977" "Gene_143" "Gene_379" "Gene_233" ...
  ..$ : chr [1:2] "PP_marginal" "Path"
```

Similarly as in section 2.3, PP(EE) and the posterior probability matrix could be obtained from:

```
> head(res.multi.nd$EEPP)
```

```
      Gene_1       Gene_2       Gene_3       Gene_4       Gene_5       Gene_6
0.0005612605 0.3429461036 0.1664478519 0.2539749342 0.0663275171 0.4248766066
```

```
> str(res.multi.nd$PP.all)
```

```
 num [1:1000, 1:16] 0.000561 0.342946 0.166448 0.253975 0.066328 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:1000] "Gene_1" "Gene_2" "Gene_3" "Gene_4" ...
  ..$ : chr [1:16] "EE-EE-EE-EE" "DE-EE-EE-EE" "EE-DE-EE-EE" "DE-DE-EE-EE" ...
```

2.7 Only consider directional patterns

If a user is interested in classifying all the genes into dynamic patterns (patterns that are constituted by 'Up' and 'Down'), SCPattern also provides such an option.

To do so, we may specify `Directional = TRUE` and `NumPat = 2` in the `SCPTest()` function:

```
> res.multi.2d <- SCPTest(SCPatternExData, CondVector, Sizes, Directional=TRUE, NumPat=2)
```

The DE genes can be obtained from:

```
> head(res.multi.2d$sortedlist)
```

```
         PP_marginal Path
Gene_233 "0.969"     "Down-Down-Up-Up"
Gene_715 "0.968"     "Up-Up-Down-Down"
Gene_267 "0.968"     "Up-Up-Down-Down"
Gene_391 "0.968"     "Up-Up-Down-Down"
Gene_102 "0.968"     "Up-Up-Down-Up"
Gene_829 "0.968"     "Down-Down-Down-Up"
```

```
> str(res.multi.2d$sortedlist)
```

```
 chr [1:1000, 1:2] "0.969" "0.968" "0.968" "0.968" "0.968" "0.968" ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:1000] "Gene_233" "Gene_715" "Gene_267" "Gene_391" ...
  ..$ : chr [1:2] "PP_marginal" "Path"
```

Since the 'EE' category is not considered here, PP(EE) is not available. The posterior probability matrix could be obtained from:

```
> str(res.multi.2d$PP.all)
```

```
 num [1:1000, 1:16] 5.84e-05 3.64e-03 9.02e-02 7.65e-02 1.98e-04 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:1000] "Gene_1" "Gene_2" "Gene_3" "Gene_4" ...
  ..$ : chr [1:16] "Down-Down-Down-Down" "Up-Down-Down-Down" "Down-Up-Down-Down" "Up-Up-Down-Down" ...
```

2.8 SCPattern-nonzero

SCPattern-nonzero is implemented in a similar way to SCPattern. In SCPattern-nonzero, for each gene, we remove cells whose expression value is zero prior to the analyses. To run SCPattern-nonzero, we may specify `Dropout.remove = TRUE` in the `SCPTest()` function:

```
> res.multi.nz <- SCPTest(SCPatternExData, CondVector, Sizes, Dropout.remove = TRUE)
```

The DE genes can be obtained from:

```
> head(res.multi.nz$sortedlist)
```

```
         PP_marginal Path
Gene_740 "1"         "Up-Up-Down-Down"
Gene_514 "1"         "Up-Up-Down-Down"
Gene_233 "1"         "Down-Down-Up-Up"
Gene_876 "1"         "Down-Down-Up-Up"
Gene_854 "1"         "Up-Up-Down-Down"
Gene_143 "1"         "Up-Up-Down-Down"
```

```
> str(res.multi.nz$sortedlist)
```

```
 chr [1:240, 1:2] "1" "1" "1" "1" "1" "1" "1" "1" "1" "1" "1" "1" ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:240] "Gene_740" "Gene_514" "Gene_233" "Gene_876" ...
  ..$ : chr [1:2] "PP_marginal" "Path"
```

To generate violin plots excluding the zeros, we may specify `Dropout.remove = TRUE` in the `VioFun` function:

SCPattern-nonzero supports different classification setups as in section 2.5, 2.6, and 2.7 as well.

```
> par(mfrow=c(2,3))
> for(i in 1:6)
+         VioFun(rownames(res.multi.nz$sortedlist)[i], DataNorm, Conditions, Dropout.remove = TRUE)
```

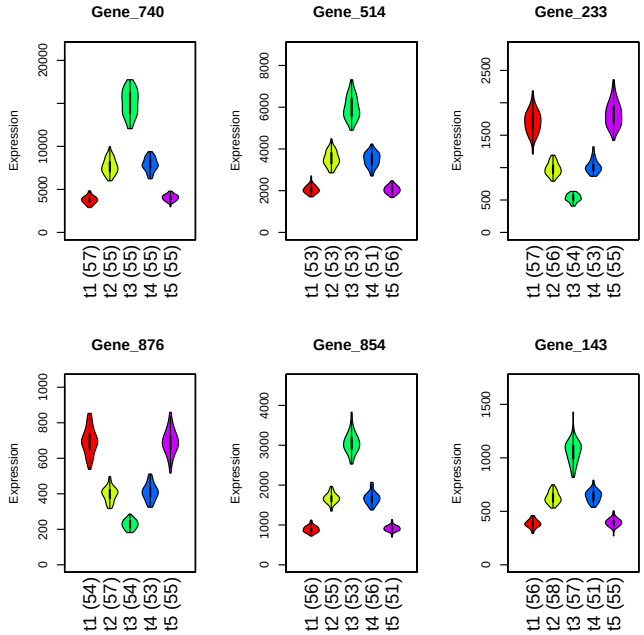

Figure 3: Top genes identified by SCPattern-nonzero. The y axis shows normalized expression. The x axis shows conditions. Cells with zero value are not shown in this plot. Number of cells considered in each condition is shown in the parentheses.

3 Session info

```
> print(sessionInfo())

R version 3.2.1 (2015-06-18)
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     datasets  methods   base

other attached packages:
 [1] SCPattern_0.0.4     DirichletReg_0.6-3  rgl_0.95.1247       Formula_1.2-1
 [5] gtools_3.5.0        vioplot_0.2         sm_2.2-5.4          EBSeq_1.11.1
 [9] testthat_0.11.0     gplots_2.17.0       blockmodeling_0.1.8

loaded via a namespace (and not attached):
 [1] splines_3.2.1       lattice_0.20-31     caTools_1.17.1      tools_3.2.1
 [5] grid_3.2.1          KernSmooth_2.23-14 maxLik_1.3-4        miscTools_0.6-16
 [9] digest_0.6.8        crayon_1.3.1        bitops_1.0-6        memoise_0.2.1
[13] VGAM_0.9-8          sandwich_2.3-4      gdata_2.17.0        stats4_3.2.1
[17] BiocStyle_1.6.0     zoo_1.7-12
```

References

[1] Ning Leng, Li-Fang Chu, Jeea Choi, Christina Kendziorski, James A Thomson, and Ron Stewart. Scpattern: A statistical approach to identify and classify expression changes in single cell rna-seq experiments with ordered conditions. *Submitted*, 2015.

[2] B Li and C N Dewey. Rsem: accurate transcript quantification from rna-seq data with or without a reference genome. *BMC Bioinformatics*, 12:323, 2011.

[3] C Trapnell, A Roberts, L Goff, G Pertea, D Kim, D R Kelley, H Pimentel, S L Salzberg, J L Rinn, and L Pachter. Differential gene and transcript expression analysis of rna-seq experiments with tophat and cufflinks. *Nature Protocols*, 7(3):562–578, 2012.

[4] S Anders and W Huber. Differential expression analysis for sequence count data. *Genome Biology*, 11:R106, 2010.