WaveCrest: a statistical approach to reconstruct gene expression trajectory in ordered single cell RNA-seq experiments

Ning Leng , Li-Fang Chu, James Thomson, and Ron Stewart

December 17, 2015

Contents

1 Introduction

WaveCrest (as detailed in Chu* and Leng* *et al.*, 2015 (1)) is a statistical approach to reconstruct gene expression trajectory in single cell RNA-seq experiments with ordered conditions. WaveCrest contains two modules - the first module implements an extended nearest insertion (ENI) algorithm that searches for optimal cell orders, and the second module implements a spline fitting module that can be used to identify additional dynamic genes.

2 Run WaveCrest

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

```
> library(WaveCrest)
```

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 `WaveCrestExData` is a simulated data matrix containing 200 rows of genes and 120 columns of cells. The genes are named `g1, g2, ...` and the cells are named `s1, s2, ...`

```
> data(WaveCrestExData)
> str(WaveCrestExData)
```

```
 num [1:200, 1:120] 99.7 134.6 105.5 83 103.3 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:200] "g1" "g2" "g3" "g4" ...
  ..$ : chr [1:120] "s1" "s2" "s3" "s4" ...
```

Here we simulated 4 time points (conditions), each has 30 cells. To specify which condition each cell belongs, we define:

```
> CondVector <- rep(paste("t",1:4,sep=""),each=30)
> str(CondVector)
```

```
 chr [1:120] "t1" "t1" "t1" "t1" "t1" "t1" "t1" "t1" "t1" "t1" ...
```

Downstream analysis by WaveCrest 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 t4, we define:

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

```
 Factor w/ 4 levels "t1","t2","t3",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
> levels(Conditions)
```

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

2.2 Normalization

WaveCrest 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(WaveCrestExData)
> str(Sizes)
```

```
 Named num [1:120] 1 1 1 1 1 ...
 - attr(*, "names")= chr [1:120] "s1" "s2" "s3" "s4" ...
```

Note that 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(WaveCrestExData,alternative = TRUE)
> str(Sizes)
```

```
 num [1:120] 1 1 1 1 1 ...
```

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

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

Note the WaveCrest analysis requires normalized expression estimates.

2.3 Recover the cell orders

WaveCrest reconstructs gene expression trajectory in ordered single cell RNA-seq experiments using an ENI algorithm. A list of key markers should be provided for the reconstruction. Such markers may be defined from differential expression analysis and/or prior knowledge (1). As an example, we define the first 8 genes in the example data set as the key markers:

```
> Markers <- rownames(DataNorm)[1:8]
> print(Markers)
```

```
[1] "g1" "g2" "g3" "g4" "g5" "g6" "g7" "g8"
```

Take a look at the key markers following the original cell order:

```
> par(mfrow=c(3,3))
> frame()
> legend("top", levels(Conditions),col=1:4,pch=1,ncol=1)
> for(i in 1:8) {
+         plot(DataNorm[Markers[i],],col=as.numeric(Conditions),
+                             ylab="Normalized expression", xlab="Original order",main=Markers[i])}
```

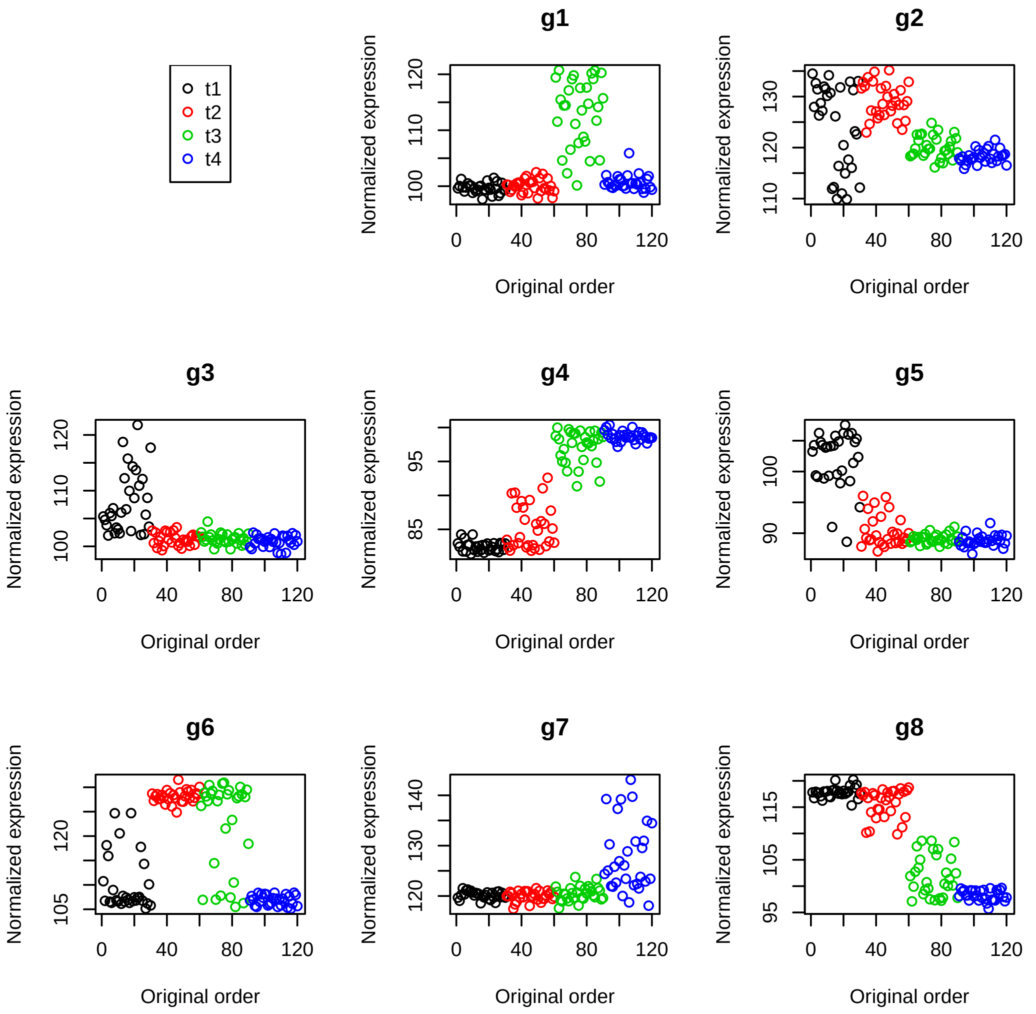

Figure 1: Key markers used in the ENI step. The y axis shows normalized expression. The x axis shows cells following original order.

Once the marker list is provided, the `WaveCrestENI()` function can be used to reconstruct the cell order. A 2-opt algorithm is also implemented in the `WaveCrestENI()` function to avoid finding local maxima. To run the ENI and 2-opt algorithm on our example data set:

```
> ENIRes <- WaveCrestENI(Markers, WaveCrestExData, Conditions, N=1000)
> str(ENIRes)
```

```
 int [1:120] 22 19 14 13 30 16 23 25 20 27 ...
```

Here `N` is used to specify the number of iterations to run in the 2-opt algorithm. The default value for `N` is 20000. In this example we set it to 1000 to reduce the run-time. In empirical analysis we recommend users to increase `N` to improve performance. The output of the `WaveCrestENI()` function is a numerical vector that indicates order of the cells. To visualize the 8 key markers following recovered cell order:

```
> par(mfrow=c(3,3))
> frame()
> legend("top", levels(Conditions),col=1:4,pch=1,ncol=1)
> for(i in 1:8) {
+         plot(DataNorm[Markers[i],ENIRes],col=as.numeric(Conditions),
+                           ylab="Normalized expression", xlab="Recovered order",
+                           main=Markers[i])}
```

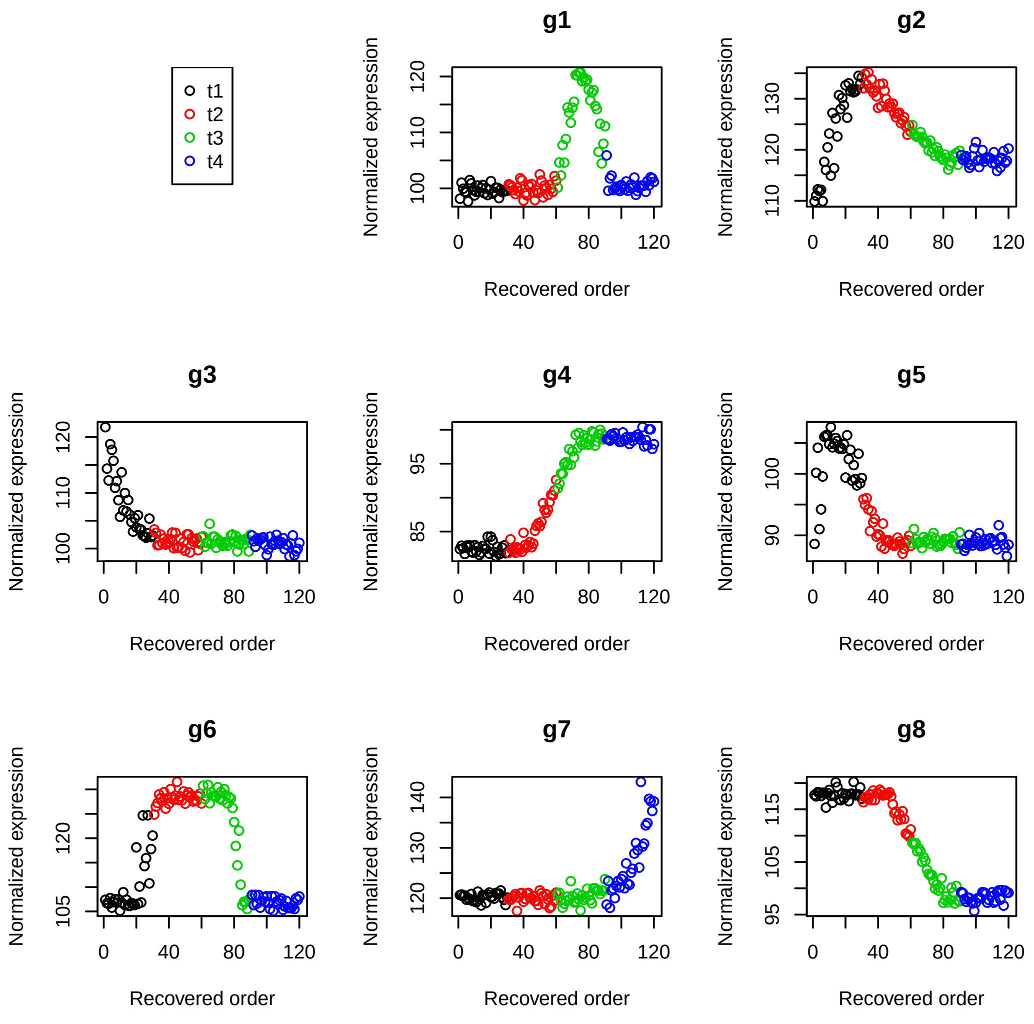

Figure 2: Key markers used in the ENI step. The y axis shows normalized expression. The x axis shows cells following WaveCrest recovered cell order.

2.4 Identify addtional dynamic genes based on the recovered order

WaveCrest also provides a function `WaveCrestIden()` to detect additional dynamic genes based on the recovered order. This allows users to 'rescue' genes that were not considered by prior studies (if the markers were picked based on prior knowledge) or were missed by prior analysis (if the markers were picked based on differential expression results, etc.). To identify additional dynamic genes, WaveCrest fits a polynomial regression on each gene's rescaled expression (z-score), following the recovered cell order. Genes with low fitting errors are considered to be top dynamic genes.

To run `WaveCrestIden()` on all remaining genes in our example data set:

```
> DataNormRemain <- DataNorm[setdiff(rownames(DataNorm),Markers),]
> IdenRes <- WaveCrestIden(DataNormRemain, ENIRes)
> IdenRes[1:5] # top 5 genes
```

```
       g16        g12        g20        g14        g13 
0.04877703 0.05635579 0.08616845 0.12269853 0.15784240 
```

The `WaveCrestIden()` function outputs mean square errors of the gene specific fittings. Genes are sorted increasingly by their mean square errors. To visualize the top 5 genes identified by the `WaveCrestIden()` function:

```
> par(mfrow=c(3,3))
> frame()
> legend("top", levels(Conditions),col=1:4,pch=1,ncol=1)
> for(i in 1:5) {
+         plot(DataNorm[names(IdenRes)[i],ENIRes],col=as.numeric(Conditions),
+                         ylab="Normalized expression", xlab="Recovered order",
+                         main=names(IdenRes)[i])}
```

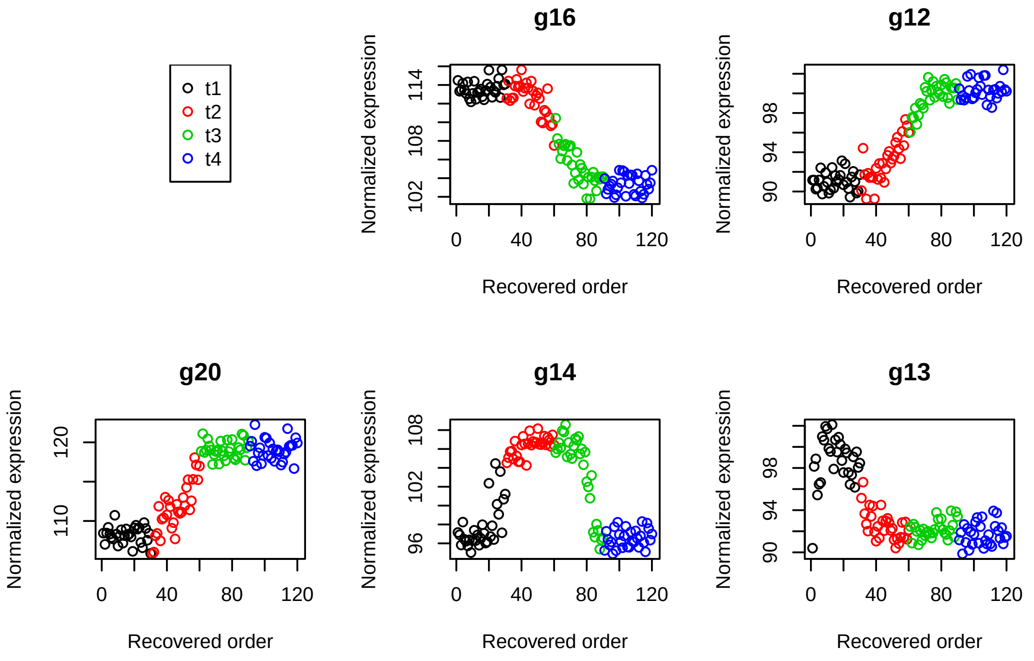

Figure 3: Top dynamic markers identified by WaveCrest. The y axis shows normalized expression. The x axis shows cells following WaveCrest recovered order.

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] WaveCrest_0.0.1     Oscope_0.99.1       BiocParallel_1.2.14 cluster_2.0.1
[5] EBSeq_1.11.1        testthat_0.11.0     gplots_2.17.0       blockmodeling_0.1.8

loaded via a namespace (and not attached):
 [1] gtools_3.5.0         digest_0.6.8         crayon_1.3.1         bitops_1.0-6
 [5] futile.options_1.0.0 KernSmooth_2.23-14   gdata_2.17.0         futile.logger_1.4.1
 [9] BiocStyle_1.6.0      lambda.r_1.1.7       tools_3.2.1          parallel_3.2.1
[13] caTools_1.17.1       memoise_0.2.1
```

References

[1] Li-Fang Chu, Ning Leng, Jue Zhang, Zhonggang Hou, Danny Mamott, David T Vereide, Christina Kendziorski, Ron Stewart, and James A Thomson. Single cell rna-seq reveals the molecular dynamics and uncovers novel regulators of human embryonic stem cell differentiation to definitive endoderm. *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.