dpath: quantitative assessment of progenitor and committed states in single cell transcriptomics

Wuming Gong

Last revised 09 March 2016

The dpath package aims to decompose the expression matrix of single cell RNA-seq, with the awareness of the dropout events, quantitatively assess the cellular state and prioritize important genes for both progenitor and committed cellular states.

1. Installing dpath

Under the Linux terminal, run:

```
%  R CMD INSTALL dpath_1.1.2.tar.gz
```

2. Loading gene expression matrix of single cell RNA-seq

In this tutorial, we first load an example single cell expression matrix for 548 genes and in 100 cells in R. The expression data need to be represented as log transformed transcript per million (TPM): $\log(TPM + 1)$.

```
> data(etv2)
> dim(etv2)
[1] 548 100
```

The row and column names of the matrix are gene symbols and cell identifiers. The expression matrix can be produced by popular RNA-seq analysis tools such as TopHat/Cufflinks, STAR, Sailfish and etc[1-3]. There is no requirement for the row and column names of the input matrix. However, we recommend to use gene symbols for row names and to use cell group identifiers for column names.

3. Decomposing expression matrix into metagenes and metacells

Next, we use the dpath() function (1) to decompose the expression matrix into metagenes by Poisson non-negative matrix factorization, (2) to cluster cells into metacells by self-organizing map (SOM), and (3) to construct the hierarchical relationships of progenitor-committed states among metacells through the metagene entropy.

```
> d <- dpath(etv2, K = 4, xdim = 10, ydim = 10)
--------------------------------------------------------------
dpath: the single cell RNA-seq analysis pipeline
```

```
-----------------------------------------------------------
number of metagenes(K)=4
number of metacells(xdim x ydim)=10x10=100
lambda for dropout event(lambda0)=0.100
weight for zero TPM(w0)=0.100
minimum number of expressed cells(expressed.min)=2
maximum percent of expressed cells(expressed.max)=100.0%
sparsity of expression matrix=92.3%
number of input cells=100
number of input genes=548
number of genes that are expressed in at least 2 cells and
at most 100.0% of all cells=548
number of repeated matrix factorization(repeat.mf)=10
number of repeated bootstrapping(repeat.bootstrap)=100
bootstrapping size(n.bootstrap)=1000
number of cores for optimization(mc.cores)=8
-----------------------------------------------------------
updating basis(U) and coefficient(V):
...................++++....++++++++++++++++++++++++++++++
++++++++++++++$+++..+++.++$...++++$++++++++$++++++++++$+$++
$+$++++$+++++$
building 100 10x10 SOM by using 1000 bootstraped cells
  |*******************| 100%
```

The arguments for `dpath()` are summarized in Table 1.

Table 1. Arguments for `dpath()`.

Arguments	Definition	Type	Range	Default
`X`	A $N \times M$ expression matrix ($\boldsymbol{X}$)	matrix	$[0, +\infty)$	NA
`K`	Number of metagenes (K)	integer	$(0, M)$	5
`xdim`	Number of metacells at x-axis of SOM	integer	$(1, +\infty)$	10
`ydim`	Number of metacells at y-axis of SOM	integer	$(1, +\infty)$	10
`lambda0`	Mean expression of the dropout events (λ_0)	real	$[0, +\infty)$	0.1
`w0`	Weight for zero entries in weighted NMF (w_0)	real	$[0, +\infty)$	0.1
`min.expressed`	Minimum number of cells that a gene is expressed	integer	$[0, M]$	2
`max.expressed`	Maximum percent of cells that a gene is expressed	real	$(0, 1]$	1
`max.iter.wnmf`	Maximum rounds of iterations of weighted NMF	integer	$(0, +\infty]$	200

`max.iter.pmnmf`	Maximum rounds of iterations of Poisson weighted NMF	integer	$(0,+\infty]$	1000
`repeat.mf`	Repetitive runs of Poisson mixture NMF (r_{mf})	integer	$(0,+\infty]$	10
`repeat.bootstrap`	Repetitive runs of bootstrapping in SOM (r_{som})	integer	$(0,+\infty]$	100
`n.bootstrap`	Bootstrapping size	integer	$(0,+\infty]$	1000
`mc.cores`	Number of utilized CPU cores	integer	$(0,+\infty]$	8

The only required argument is $\boldsymbol{X}$, the input expression matrix. The running time of `dpath()` depends on the size of the expression matrix, the number of metagenes K, and the maximum rounds of interactions of weighted and Poisson mixture NMF. `dpath()` uses R `parallel` package to speed up the optimization process, where the number of utilized CPU cores can be specified by the argument `mc.cores`.

For the example dataset (548 genes and in 100 cells), it took less than one minute to finish on an 8-core Intel Xeon 2.80 GHz computer. For a typical genome-wise single cell RNA-seq dataset that have ~10,000 genes and 200 cells, it took ~30 minutes on the same computer.

4. Visualizing the results

We provide a function `plot()` to visualize various information stored in a `dpath` object, returned by the `dpath()` function.

4.1 Metagene coefficients, metagene basis and observed expression levels for selected marker genes.

```
> dev.new(width = 15, height = 15)
> plot(d, type = 'markers', gene = c("T", "Kdr", "Gata4", "Gata5",
"Gata6", "Tbx5", "Hand1", "Hand2", "Alcam", "Smarcd3", "Myl7", "Tnnt2",
"Mef2c", "Isl1", "Wt1", "Nrg1", "Nfatc1", "Cgnl1", "Dok4", "Etv2",
"Plvap", "Pecam1", "Cdh5", "Tal1", "Tie1", "Tek", "Gata2", "Flt1",
"Emcn", "Lmo2", "Cd34", "Rasip1", "Gata1", "Hbb-y", "Itga2b", "Ikzf1",
"Runx1", "Ighmbp2", "Smo", "Bmp2", "Bmp4", "Notch1", "Wnt5a", "Vegfa",
"Ephb4", "Nr2f2", "Pdgfra", "Sox7", "Cldn6", "Krt19"))
48 of 50 supplied genes are included
```

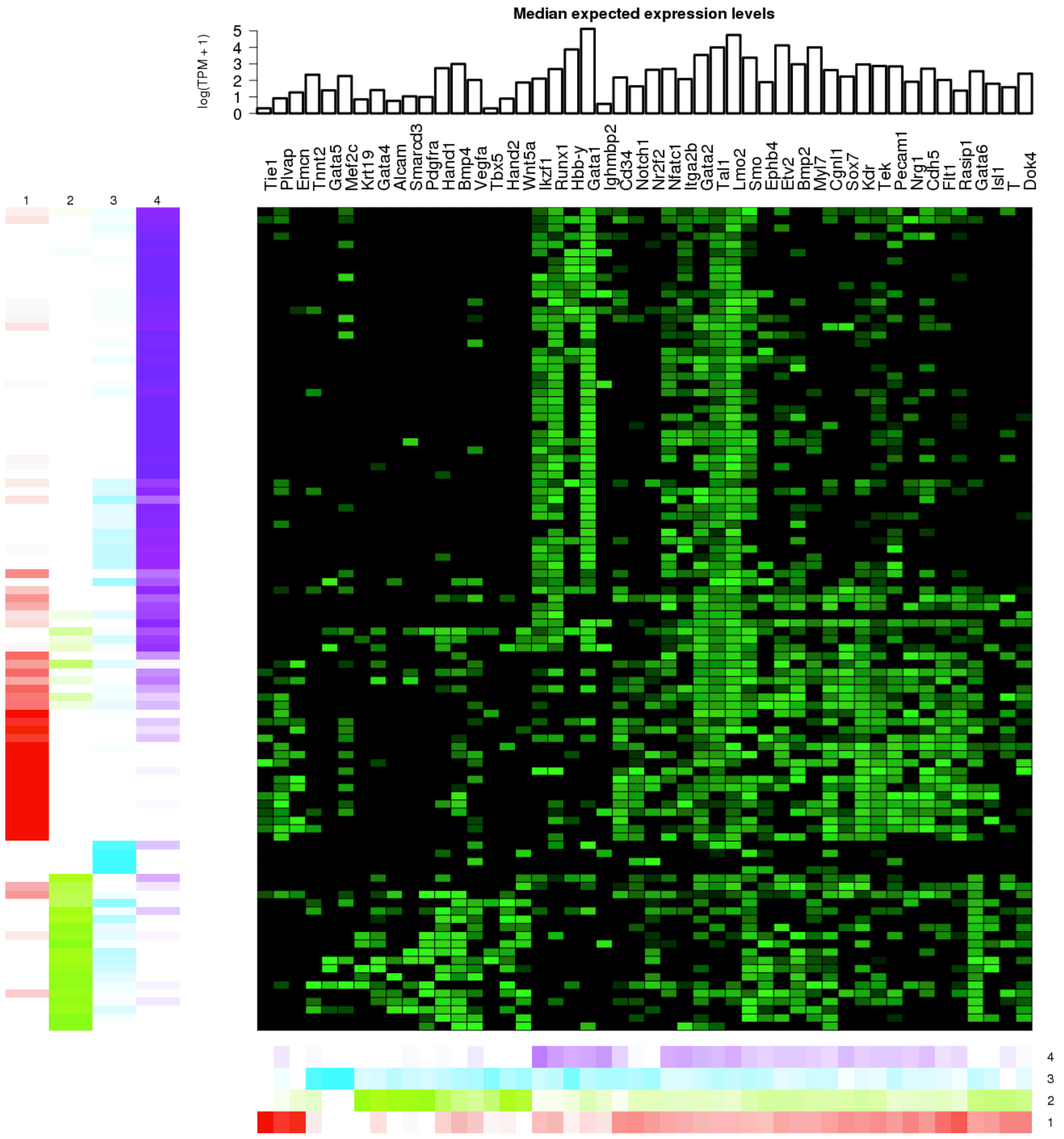

Figure 1. Visualizing metagene coefficients, metagene basis and expression levels for selected marker genes by `plot(..., type = 'markers', ...)`.

The `gene` argument accepts a list of marker gene names. If the gene names do not match the row names in the input matrix, they will be automatically excluded.

In the output plot (Figure 1), the left panel shows the mean estimated metagene coefficients ($\boldsymbol{V}$) for all cells where each column represents one metagene, and the color indicates the expression intensities of the metagene in each cell and the cell-wise metagene coefficients is summed up to one. The top right panel shows the median expected expression level for each marker gene. The middle right panel shows the observed expression level of each gene in each cell. The bottom right panel shows the metagene basis ($\boldsymbol{U}$), that is, the contribution of each gene to each metagene.

4.2 Metagene entropy and metagene expression on the SOM

We provide a way to visualize the distribution metagene entropy on the SOM (Figure 2).

```
> dev.new(width = 15, height = 5)
> plot(d, type = 'metagene.entropy')
```

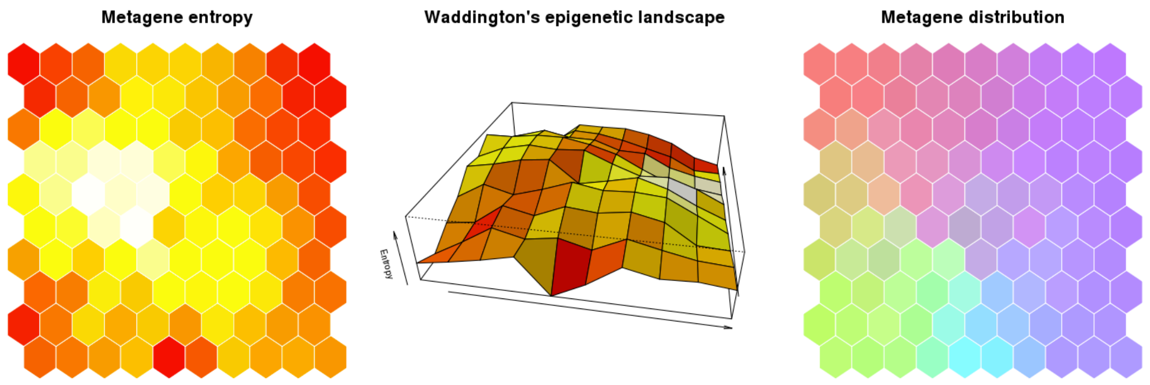

Figure 2. Visualizing metagene entropy on the SOM by `plot(..., type = 'metagene.entropy', ...)`.

The left and middle panels show the distribution of metagene entropy on a 10×10 SOM in 2-D and 3-D view, respectively. The right panel shows the distribution of expression levels of each metagene on the SOM.

4.3 Distribution of a specified group of cells on the SOM

Using `plot(..., type = 'cell.distribution', ...)`, we can visualize the distribution of a specified group of cells on the SOM. Here we check the distribution of the cells from E8.25 on the SOM (Figure 3):

```
> dev.new(width = 5, height = 5)
> plot(d, type = 'cell.distribution', cell.group = colnames(etv2) == 
'E8.25')
```

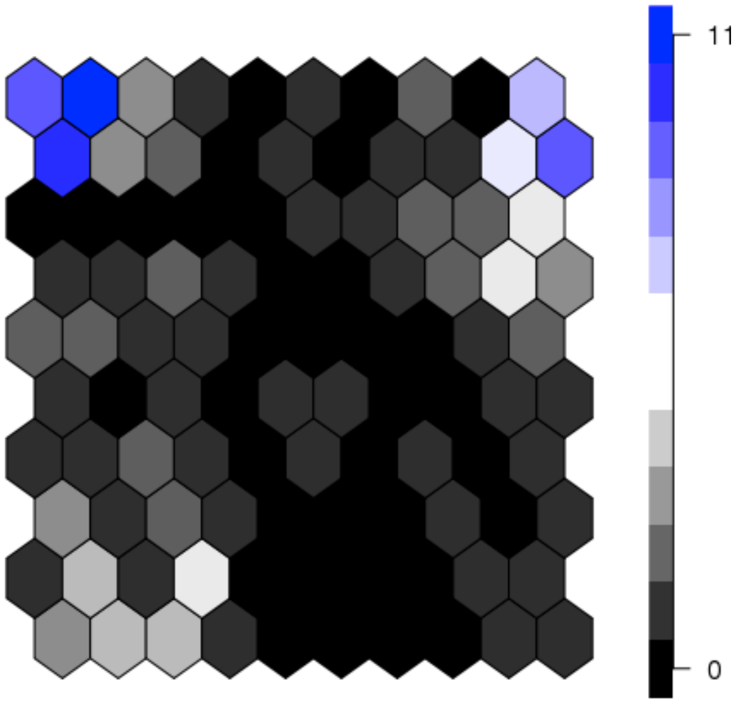

Figure 3. Distribution of a specified group of cells on the SOM by `plot(..., type = 'cell.distribution', ...)`.

As indicated in Figure 3, a majority of the cells from E8.25 located at the top left and right corner of the SOM, which corresponds to the 1st (red) and 4th (purple) metagene.

4.4 Examining expression pattern for individual genes on SOM.

We can use `plot(..., type = 'gene.expression', ...)` to visualize the expression pattern of individual gene on the SOM. Here is an example to explore the expression pattern of four genes Etv2, T, Hbb-y and Plvap.

```
> dev.new(width = 5, height = 5)
> plot(d, type = 'gene.expression', gene = 'Etv2')
> dev.new(width = 5, height = 5)
> plot(d, type = 'gene.expression', gene = 'T')
> dev.new(width = 5, height = 5)
> plot(d, type = 'gene.expression', gene = 'Hbb-y')
> dev.new(width = 5, height = 5)
> plot(d, type = 'gene.expression', gene = 'Plvap')
```

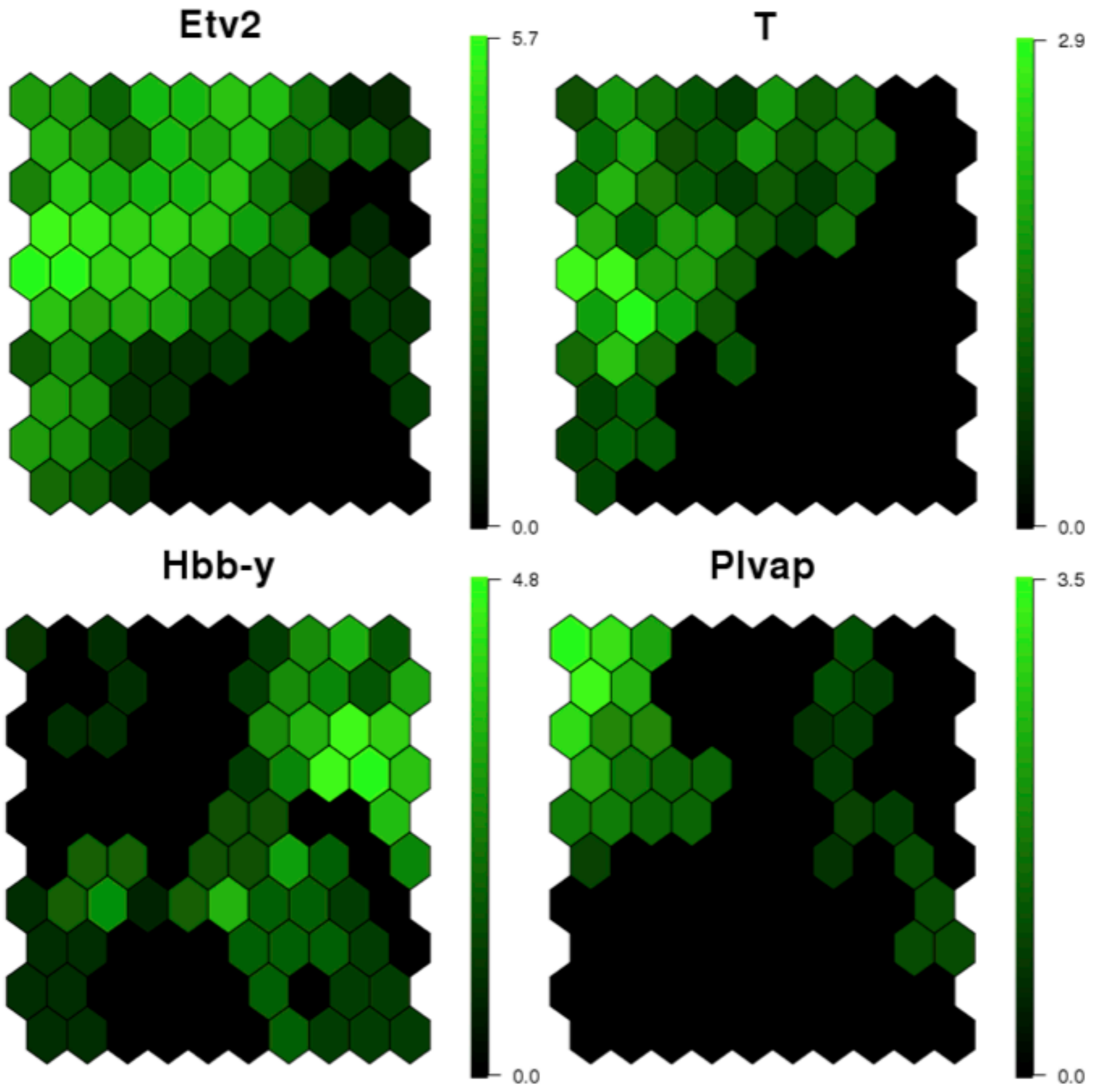

Figure 4. Visualizing gene expression pattern on the SOM by `plot(..., type = 'gene.expression', ...)`. Color represents the expression level in $\ln(TPM + 1)$

4.5 Prioritizing metacells with respect to specified cellular states (committed or progenitor) and ranking genes by enrichment scores.

One unique advantage of `dpath` pipeline is able to prioritize the metacells with respect to specified cellular states and rank genes based on the correlation between the their expression pattern and the cellular states of the metacells. We prioritize metacells with committed states of the 1st metagene (red) by the `prioritize()` function:

```
> pr2c <- prioritize(d, metagene = c(1,0,0,0), direction = 'committed')
Prioritizing committed metacells for metagene signature:
MG1:1.000; MG2:0.000; MG3:0.000; MG4:0.000
```

```
  |********************| 100%
Computing the gene enrichment score:
  |******************* | 100%
```

The `prioritize` function accept the `dpath` object returned by the `dpath()` function as the first argument. The second argument `metagene` indicates the desired weight for each metagene with respect to the cellular states. The third argument `direction` is either "committed" or "progenitor", depending on the types of the cellular states. The returned object of the `prioritize` function is a named list with two slots, where the first slot "metacell" is the metacell prioritization scores, while the second slot "gene" is the gene ranking scores.

Next, we plot the prioritization score of metacells with respect to the committed state of the 1st metagene, as well as the top 20 ranking genes which expression pattern is most correlated with the prioritization score.

```
> dev.new(width = 15, height = 15)
> plot(d, type = 'prioritization', score.gene = pr2c[['gene']],
score.metacell = pr2c[['metacell']], top = 20)
```

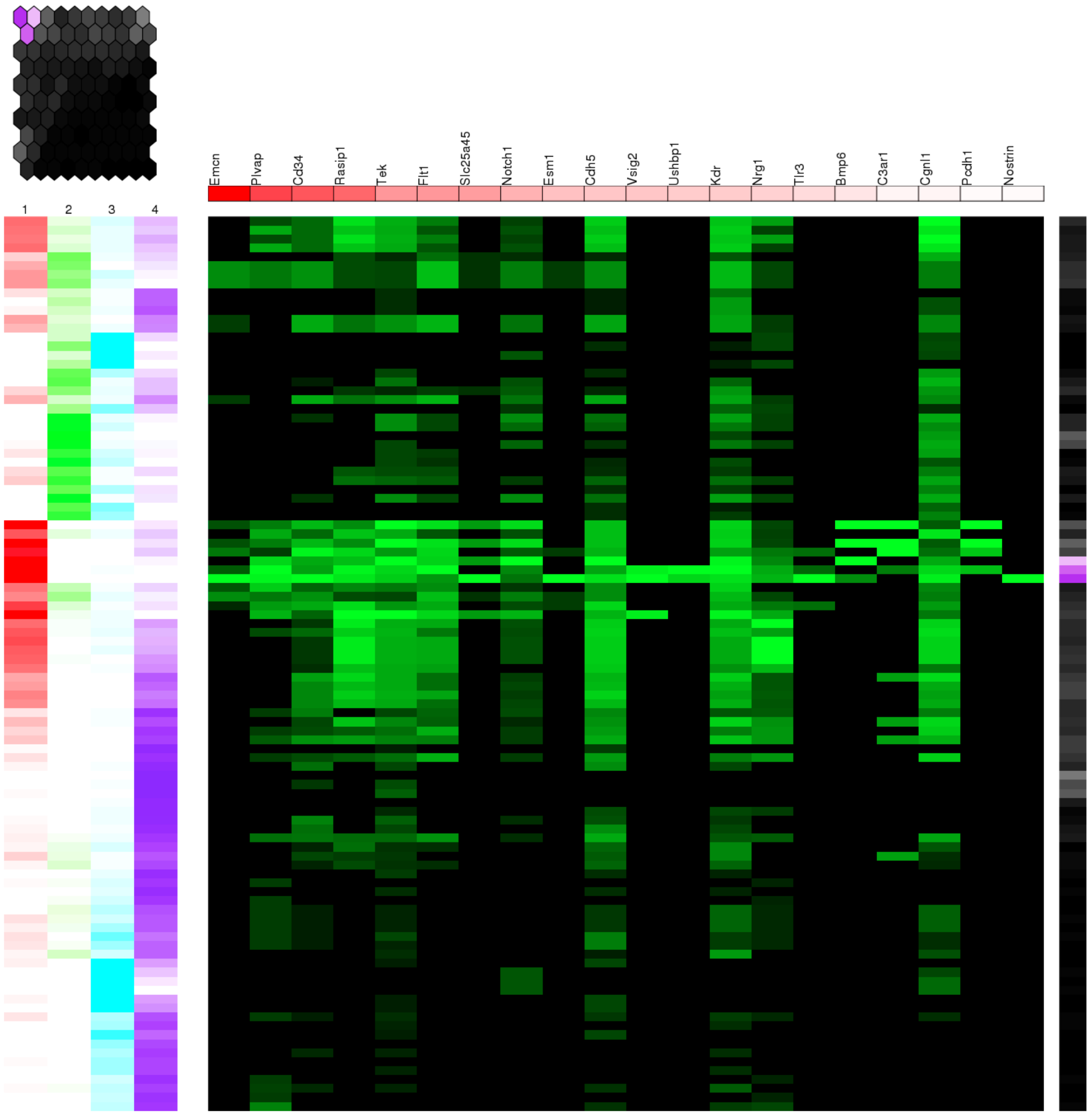

Figure 5. Visualizing prioritization score of metacells with respect to the committed cellular states and top ranking genes which expression pattern is most correlated with the prioritization scores, by `plot(..., type = 'prioritization', ...)`.

In the output plot (Figure 5), the top left panel is the SOM where the color indicates the metacell prioritization score. The bottom left panel indicates the metagene expression profile of different metacells. The middle right panel is the observed expression pattern of top genes in metacells. The top right panel indicates the list of top genes. The right panel also represents the metacell-wise prioritization score.

Similarly, we prioritize the metacells for progenitor states for the 1st and the 4th metagene and visualize the top ranking genes:

```
> pr2p <- prioritize(d, metagene = c(1,0,0,1), direction ='progenitor')
```

```
Prioritizing progenitor metacells for metagene signature:
MG1:0.500; MG2:0.000; MG3:0.000; MG4:0.500
  |********************| 100%
Computing the gene enrichment score:
  |******************* |  95%
> dev.new(width = 15, height = 15)
> plot(d, type = 'prioritization', score.gene = pr2p[['gene']],
score.metacell = pr2p[['metacell']], top = 20)
```

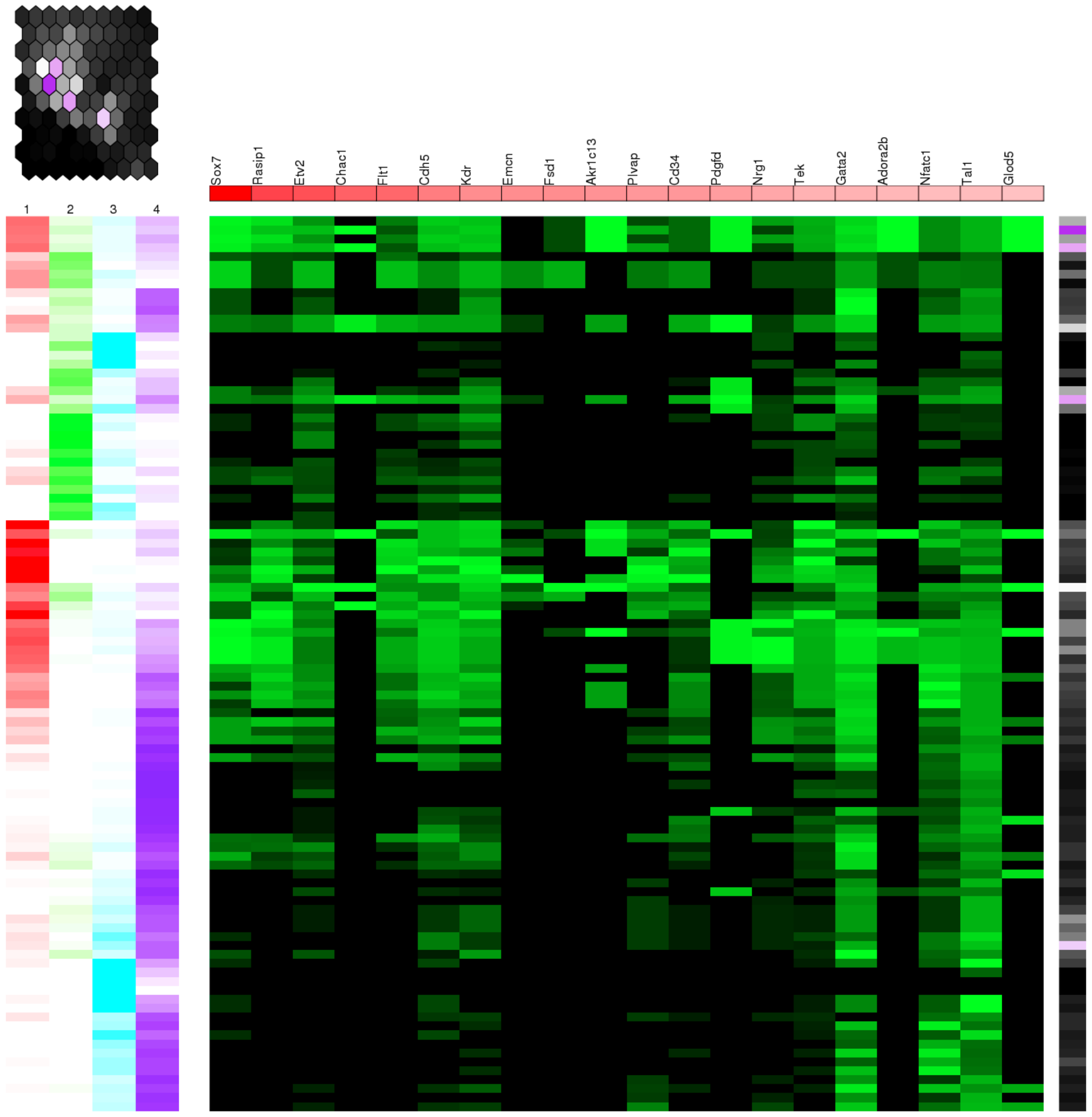

Figure 6. Visualizing prioritization score of metacells with respect to the progenitor cellular states and top ranking genes which expression pattern is most correlated with the prioritization scores, by `plot(..., type = 'prioritization', ...)`.

Note that distribution of prioritized committed and progenitor metacells on the metacell landscape are different: the metacells with committed state locate at the corner of the SOM (Figure 5, top left panel), while the metacells with progenitor state locate closer to the center of the SOM (Figure 6, top left panel), which have higher metagene entropy.

5. Conclusion

These case studies have demonstrated a few of the most important features of the `dpath` package for analyzing single cell RNA-seq data. Please see the package documentation for more details.

The following is the session information that generated this vignette:

```
> sessionInfo()
R version 3.1.1 (2014-07-10)
Platform: x86_64-unknown-linux-gnu (64-bit)

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

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

other attached packages:
[1] hopach_2.26.0       Biobase_2.26.0      BiocGenerics_0.12.1
[4] cluster_2.0.1       dpath_1.1.2

loaded via a namespace (and not attached):
 [1] bitops_1.0-6       caTools_1.17.1     class_7.3-12
colorspace_1.2-6
 [5] fields_8.2-1       FNN_1.1            gdata_2.13.3
gplots_2.16.0
 [9] grid_3.1.1         gtools_3.4.2       igraph_0.7.1
irlba_1.0.3
[13] KernSmooth_2.23-14 kohonen_2.0.18     lattice_0.20-31    maps_2.3-
9
[17] MASS_7.3-40        Matrix_1.2-0       spam_1.0-1
tools_3.1.1
```

References

1. Trapnell, C. *et al.* Differential gene and transcript expression analysis of RNA-seq experiments with TopHat and Cufflinks. *Nat Protoc* **7,** 562–578 (2012).
2. Patro, R., Mount, S. M. & Kingsford, C. Sailfish enables alignment-free isoform quantification from RNA-seq reads using lightweight algorithms. *Nat.*

Biotechnol. **32,** 462–464 (2014).
3. Dobin, A. *et al.* STAR: ultrafast universal RNA-seq aligner. *Bioinformatics* **29,** 15–21 (2013).