The oposSOM Package

Henry Löffler-Wirth, Martin Kalcher

October 17, 2016

High-throughput technologies such as whole genome transcriptional profiling revolutionized molecular biology and provide an incredible amount of data. On the other hand, these techniques pose elementary methodological challenges simply by the huge and ever increasing amount of data produced: researchers need adequate tools to extract the information content of the data in an effective and intelligent way. This includes algorithmic tasks such as data compression and filtering, feature selection, linkage with the functional context, and proper visualization. Especially, the latter task is very important because an intuitive visualization of massive data clearly promotes quality control, the discovery of their intrinsic structure, functional data mining and finally the generation of hypotheses. We aim at adapting a holistic view on the gene activation patterns as seen by expression studies rather than to consider single genes or single pathways. This view requires methods which support an integrative and reductionist approach to disentangle the complex gene-phenotype interactions related to cancer genesis and progression. With this motivation we implemented an analysis pipeline based on data processing by a Self-Organizing Map (SOM) (**?**)(**?**). This approach simultaneously searches for features which are differentially expressed and correlated in their profiles in the set of samples studied. We include functional information about such co-expressed genes to extract distinct functional modules inherent in the data and attribute them to particular types of cellular and biological processes

such as inflammation, cell division, etc. This modular view facilitates the understanding of the gene expression patterns characterizing different cancer subtypes on the molecular level. Importantly, SOMs preserve the information richness of the original data allowing the detailed study of the samples after SOM clustering. A central role in our analysis is played by the so-called expression portraits which serve as intuitive and easy-to-interpret fingerprints of the transcriptional activity of the samples. Their analysis provides a holistic view on the expression patterns activated in a particular sample. Importantly, they also allow identification and interpretation of outlier samples and, thus, improve data quality (**?**)(**?**).

1 Example data: transctiptome of healthy human tissue samples

The data was downloaded from Gene Expression Omnibus repository (GEO accession no. GSE7307). About 20,000 genes in more than 650 tissue samples were measured using the Affymetrix HGU133-Plus2 microarray. A subset of 12 selected tissues from different categories is used here as example data set for the oposSOM-package.

2 Setting up the environment

In order to set the analysis parameters and to create the enclosing environment it is obligatory to use **opossom.new**. If any parameter is not explicitly defined, default values will be used (see also Parameters section):

```
> library(oposSOM)
> env <- opossom.new(list(dataset.name="Tissues",
+                         dim.1stLvlSom=20))
```

The oposSOM package requires input of the expression data. Usually the raw microarray intensity data is preprocessed using appropiate calibration and summarization algorithms (e.g. MAS5, VSN or RMA), and transformed into logarithmic scale prior to utilizing them in the pipeline.
The package then accepts two formats: Firstly a simple two-dimensional numerical matrix, where the columns and rows represent the samples and genes, respectively:

```
> data(opossom.tissues)
> str(opossom.tissues, vec.len=3)

 num [1:20957, 1:12] 0.299 2.492 2.293 2.041 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:20957] "ENSG00000115415" "ENSG00000252095" "ENSG00000111640" ...
  ..$ : chr [1:12] "liver" "kidney cortex" "thyroid gland" ...

> env$indata <- opossom.tissues
```

Secondly the input data can also be given as *Biobase::ExpressionSet* object:

```
> data(opossom.tissues)
> library(Biobase)
> opossom.tissues.eset = ExpressionSet(assayData=opossom.tissues)
> opossom.tissues.eset

ExpressionSet (storageMode: lockedEnvironment)
assayData: 20957 features, 12 samples
  element names: exprs
protocolData: none
phenoData: none
featureData: none
experimentData: use 'experimentData(object)'
Annotation:

> env$indata <- opossom.tissues.eset
```

Each sample may be assigned to a distinct group and a corresponding color to improve data visualization and result presentations. *group.labels* can also be set to *"auto"* to apply unsupervised grouping of samples according to their expression module activation patterns. Otherwise, samples will be collected within one group and colored using a standard scheme.

```
> env$group.labels <- c(rep("Homeostasis", 2),
+                           "Endocrine",
+                           "Digestion",
+                           "Exocrine",
+                           "Epithelium",
+                           "Reproduction",
+                           "Muscle",
+                       rep("Immune System", 2),
+                       rep("Nervous System", 2) )

> env$group.colors <- c(rep("gold", 2),
+                           "red2",
+                           "brown",
+                           "purple",
+                           "cyan",
+                           "pink",
+                           "green2",
+                       rep("blue2", 2),
+                       rep("gray", 2) )
```

Alternatively, the *group.labels* and *group.colors* can also be defined within the phenotype information of the ExpressionSet:

```
> group.info <- data.frame(
+                   group.labels = c(rep("Homeostasis", 2),
+                                    "Endocrine",
+                                    "Digestion",
+                                    "Exocrine",
+                                    "Epithelium",
+                                    "Reproduction",
+                                    "Muscle",
+                                  rep("Immune System", 2),
+                                  rep("Nervous System", 2) ),
+
+                   group.colors = c(rep("gold", 2),
+                                    "red2",
+                                    "brown",
+                                    "purple",
+                                    "cyan",
+                                    "pink",
+                                    "green2",
+                                  rep("blue2", 2),
+                                  rep("gray", 2) ),
+
+                   row.names=colnames(opossom.tissues))

> opossom.tissues.eset = ExpressionSet(assayData=opossom.tissues,
+                                      phenoData=AnnotatedDataFrame(group.info) )
> opossom.tissues.eset

ExpressionSet (storageMode: lockedEnvironment)
assayData: 20957 features, 12 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: liver kidney cortex ... cerebral cortex (12 total)
  varLabels: group.labels group.colors
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
Annotation:

> env$indata <- opossom.tissues.eset
```

Finally the pipeline will run through all analysis modules without further input. Periodical status messages are given to inform about running and accomplished tasks. Please note that the tissue sample will take approx. 30min to finish, depending on the users' hardware:

```
> opossom.run(env)
```

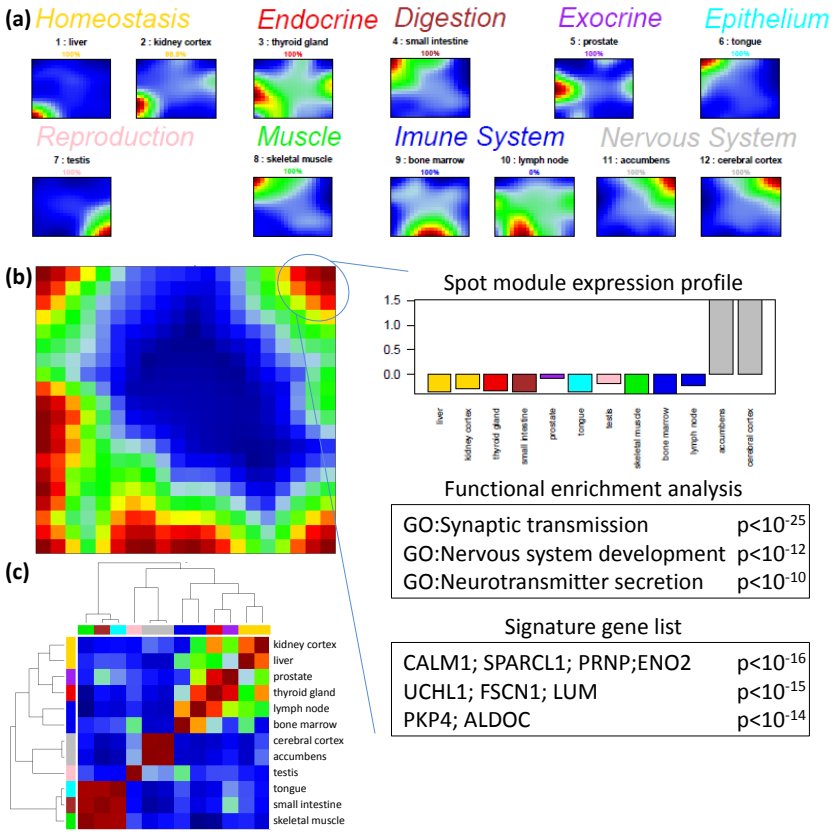

Figure 1: Few selected results provided by the oposSOM package: (a) Expression landscape portraits represent fingerprints of transcriptional activity. The *group.labels* and *group.colors* parameters are used to arrange and represent the samples throughout all analyses. (b) Functional expression modules are identified in the expression landscapes and described using appropriate summary portraits (left part), and expression profiles, enrichment analyses and differential gene lists (right part). (c) Sample similarity structure is analysed using different algorithms and distance metrics. Here a clustered pairwise correlation matrix is shown.

3 Browsing the results

The pipeline will store the results in a defined folder structure. These results comprise a variety of PDF documents with plots and images addressing the input data, supplementary descriptions of the SOM generated, the metadata obtained by the SOM algorithm, the sample similarity structures and also functional annotations. The PDF reports are accompanied by detailed CSV spreadsheets which render the complete information richness accessible.
Figure **??** shows few selected outputs generated by the pipeline. The expression landscape portraits (Figure **??**a) represent fingerprints of transcriptional activity. They are used to identify functional expression modules, which are further visualized and evaluated (Figure **??**b). Sample similarity structure is analysed using different algorithms and distance metrics, for example by clustering the pairwise sample correlation matrix (Figure **??**c).
Detailed description of the respective algorithms and visualizations would exceed the scope of this outline. We therefore refer to our publications aiming at methodical issues and application of the pipeline (**?**)(**?**)(**?**)(**?**)(**?**)(**?**)(**?**)(**?**).
HTML files are generated to provide straightforward access to this great amount of analysis results (see Figure **??**). They guide the user in terms of giving the most prominent links at a glance and leading from one analsis module to another. The **Summary.html** is the starting point of this browsing and can be found in the results folder created by the oposSOM pipeline.

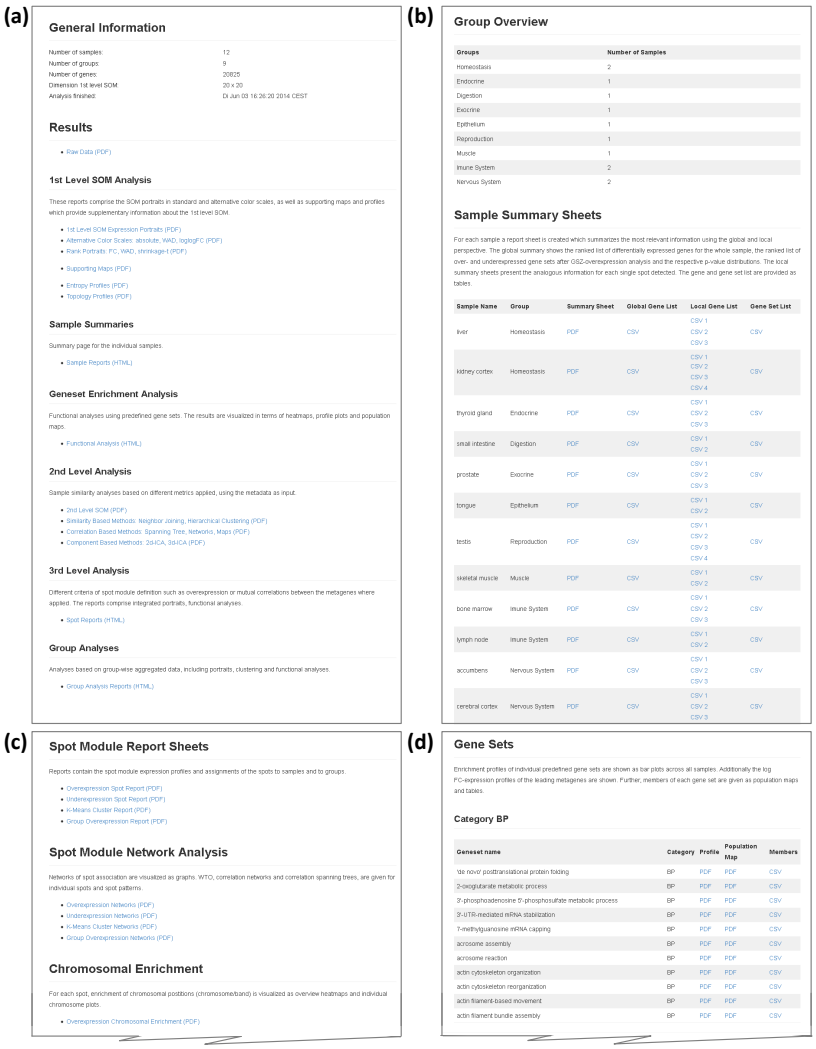

Figure 2: HTML files allow browsing all results provided by the oposSOM package: (a) The central *Summary.html* serves as starting point and contains general information and results, as well as links to other HTML files such as (b) the sample summary page, (c) the spot module summary page and (d) the functional analyses page.

4 Parameter settings

All parameters are optional and will be set to default values if missing. However we recommend to adapt the following parameters according to the respective analysis:

- *dataset.name* (character): name of the dataset. Used to name results folder and environment image (default: "Unnamed").
- *dim.1stLvlSom* (integer): dimension of primary SOM (default: "auto"). Given as a single value defining the size of the square SOM grid. Use "auto" to set SOM size to recommendation (see below).
- *feature.centralization* (boolean): enables or disables centralization of the features (default: TRUE).
- *sample.quantile.normalization* (boolean): enables quantile normalization of the samples (default: TRUE).

Database parameters are required to enable gene annotations and functional analyses (details are given below):

- *database.dataset* (character): type of ensemble dataset queried using biomaRt interface (default: "auto"). Use "auto" to detect database parameters automatically.
- *database.id.type* (character): type of rowname identifier in biomaRt database (default: ""). Obsolete if *database.dataset="auto"*.

The parameters below are secondary and may be left unattended by the user:

- *dim.2ndLvlSom* (integer): dimension of the second level SOM (default: 20). Given as a single value defining the size of the square SOM grid.
- *training.extension* (numerical, >0): factor extending the number of iterations in SOM training (default: 1).
- *rotate.SOM.portraits* (integer {0,1,2,3}): number of roations of the primary SOM in counter-clockwise fashion (default: 0). This solely influences the orientation of the portraits.
- *flip.SOM.portraits* (boolean): mirroring the primary SOM along the bottom-left to top-right diagonal (default: FALSE). This solely influences the orientation of the portraits.
- *geneset.analysis* (boolean): enables or disables geneset analysis (default: TRUE).

- *geneset.analysis.exact* (boolean): enables or disables p-value and fdr calculation in geneset analysis (default: TRUE). Obsolete if *geneset.analysis=F*.

- *standard.spot.modules* (character, one of {"overexpression", "underexpression", "kmeans", "correlation", "group.overexpression", "dmap"}): spot modules utilized in diverse downstream analyses and visualizations, e.g. PAT detection and module correlation map (default: "dmap").

- *spot.threshold.modules* (numerical, between 0 and 1): spot detection in summary maps, expression threshold (default: 0.95).

- *spot.coresize.modules* (integer, >0): spot detection in summary maps, minimum spot size (default: 3).

- *spot.threshold.groupmap* (numerical, between 0 and 1): spot detection in group-specific summary maps, expression threshold (default: 0.75).

- *spot.coresize.groupmap* (integer, >0): spot detection in group-specific summary maps, minimum spot size (default: 5).

- *pairwise.comparison.list* (list of group lists): group list for pairwise analyses (default: NULL). Each element is a list of two character vectors containing the sample names to be analysed in pairwise comparison. The sample names must be contained in the column names of the input data matrix. For example, the following setting will compare the homeostasis (liver, kidney) to the nervous system samples (accumbens, cortex), and also tongue and intestine to the nervous system:

```
> env$preferences$pairwise.comparison.list <-
+     list(list(c("liver","kidney cortex"),
+               c("accumbens","cerebral cortex")),
+          list(c("tongue","small intestine"),
+               c("accumbens","cerebral cortex")))
```

5 Recommended SOM size and runtime estimation

The size of the SOM required to resolve main expression modules depends on both the number of features (e.g. genes measured) and the number of samples. Here we give a recommendation based on previous analyses of a multitude of different data sets (see Figure **??**). Addionally, we give a rough estimation for runtime of the SOM training algorithm.

Size recommendation

Number of genes \ Number of cells	< 100	100 - 500	500 - 1,000	1,000 - 5,000	> 5,000
< 1,000	20 x 20	25 x 25	30 x 30	35 x 35	40 x 40
1,000 - 10,000	30 x 30	35 x 35	40 x 40	45 x 45	50 x 50
10,000 - 50,000	40 x 40	45 x 45	50 x 50	55 x 55	60 x 60

Approx. runtime

(Intel Core i7) Number of genes \ Number of cells	< 100	100 - 500	500 - 1,000	1,000 - 5,000	> 5,000
< 1,000	< 1 min	~ 10 min	~ 25 min	~ 3 h	> 5 h
1,000 - 10,000	~ 20 min	~ 2 h	~ 7 h	~ 2 d	> 3 d
10,000 - 50,000	~ 3 h	~ 25 h	~ 2 d	~ 25 d	> 1 month

Figure 3: Recommended size of the SOM and estimated runtime of the SOM training on an Intel Core i7 system (16GB RAM).

6 Biomart database settings

Two parameters are required to access gene annotations and functional information via biomaRt interface:

database.dataset defines the Ensembl data set to be queried, e.g. "hsapiens_gene_ensembl", "mmusculus_gene_ensembl" or "rnorvegicus_gene_ensembl". A complete list of possible entries can be obtained by

```
> library(biomaRt)
> mart<-useMart("ensembl")
> listDatasets(mart)
```

The default setting "auto" will cause oposSOM to test frequently used settings of *database.dataset* and *database.id.type*. If this automatic download of annotation data fails, a warning will be given and manual definition of the parameters will be necessary to enable functional analyses.

database.id.type provides information about the identifier type constituted by the rownames of the expression matrix, e.g. "ensembl_gene_id", "refseq_mrna" or "affy_hg_u133_plus_2". A complete list of possible entries can be obtained by

```
> library(biomaRt)
> mart<-useMart(biomart="ensembl", dataset="hsapiens_gene_ensembl")
> listFilters(mart)
```

7 New functionalities introduced with oposSOM 1.0 on Bioconductor

The oposSOM-package release on Bioconductor is highly superior to the version released on CRAN in 2011:

- Structure of the source code was thoroughly revised to meet the requirements of Bioconductor.
- Organization and presentation of the results output was improved, accompanied with an extended HTML interface to access all results.
- A package vignette was introduced.
- New analysis modules were implemented:
  - Metagene entropy and portrait topology analyses
  - Neighbor-joining clustering of the samples
  - Correlation Network analysis of the samples
  - GSZ-profiles for the individual gene sets
  - Overview heatmaps summarizing enrichment of a large number of gene sets
  - Cancer hallmark enrichment analyses
  - Enrichment analyses for genes sets relating to chromosomal positions
  - Spot report sheets and spot correlation (wTO) networks
  - Expression portraits, differential expression analyses and functional characteristics summarized for the groups defined
  - Stability analyses of the groups using correlation silhouette methods
  - Differential expression analyses for pairs of samples or groups of samples, including differential expression portraits and functional characterization
- Primary input data can be given as Bioconductor 'ExpressionSet' object.

8 Citing oposSOM

Please cite (**?**) when using the package.

9 Details

This document was written using:

```
> sessionInfo()

R version 3.3.1 (2016-06-21)
Platform: x86_64-apple-darwin13.4.0 (64-bit)
Running under: OS X 10.9.5 (Mavericks)

locale:
[1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

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

other attached packages:
[1] Biobase_2.34.0      BiocGenerics_0.20.0 oposSOM_1.12.0
[4] igraph_1.0.1

loaded via a namespace (and not attached):
 [1] lattice_0.20-34      ape_3.5              IRanges_2.8.0
 [4] XML_3.98-1.4         bitops_1.0-6         grid_3.3.1
 [7] nlme_3.1-128         DBI_0.5-1            stats4_3.3.1
[10] magrittr_1.5         RSQLite_1.0.0        KernSmooth_2.23-15
[13] som_0.3-5.1          pixmap_0.4-11        fdrtool_1.2.15
[16] scatterplot3d_0.3-37 S4Vectors_0.12.0     fastICA_1.2-0
[19] tools_3.3.1          biomaRt_2.30.0       RCurl_1.95-4.8
[22] AnnotationDbi_1.36.0
```