The oposSOM Package

Henry Löffler-Wirth, Martin Kalcher

April 25, 2023

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) (Wirth et al., 2011)(Wirth et al., 2012a)(Löffler-Wirth et al., 2015). 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 (Hopp et al., 2013a)(Hopp et al., 2013b).

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, for example preprocessed RNA microarray or sequencing data. It is recommended to transform data into logarithmic scale prior to utilizing them in the pipeline.
The workflow 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)
```

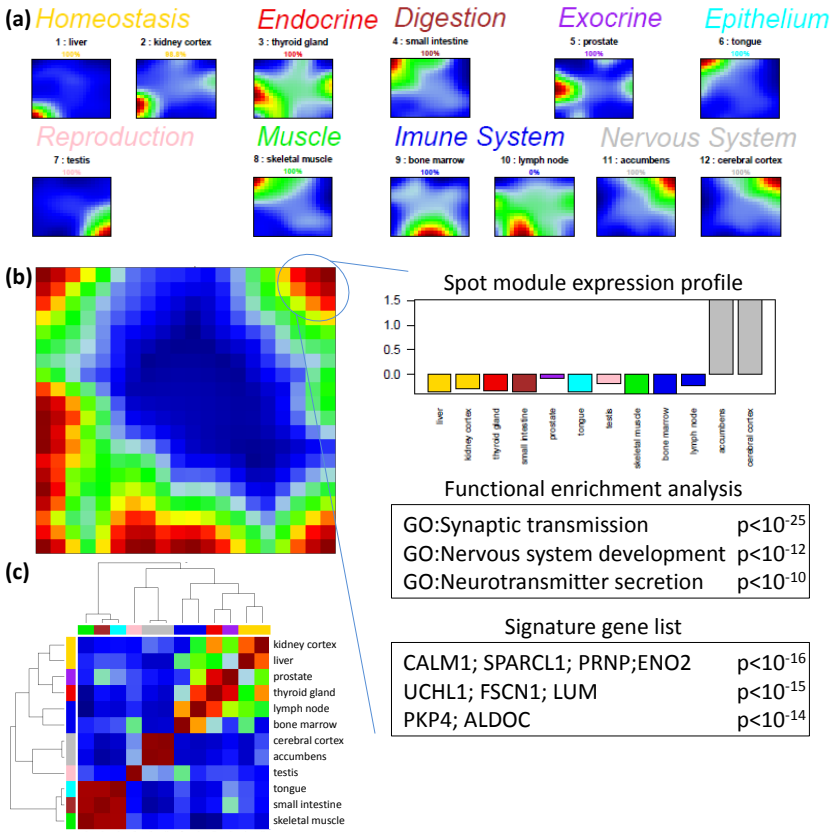

Figure 1: Few selected results provided by the oposSOM package: (a) Expression landscape portraits represent fingerprints of transcriptional activity. The *group.labels* and *group.colors* parameters are used to arrange and represent the samples throughout all analyses. (b) Functional expression modules 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 1 shows few selected outputs generated by the pipeline. The expression landscape portraits (Figure 1a) represent fingerprints of transcriptional activity. They are used to identify functional expression modules, which are further visualized and evaluated (Figure 1b). Sample similarity structure is analysed using different algorithms and distance metrics, for example by clustering the pairwise sample correlation matrix (Figure 1c).
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 (Wirth et al., 2011)(Wirth et al., 2012b)(Wirth et al., 2012a)(Wirth, 2012)(Steiner et al., 2012)(Binder et al., 2012)(Hopp et al., 2013a)(Hopp et al., 2013b).
HTML files are generated to provide straightforward access to this great amount of analysis results (see Figure 2). 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.

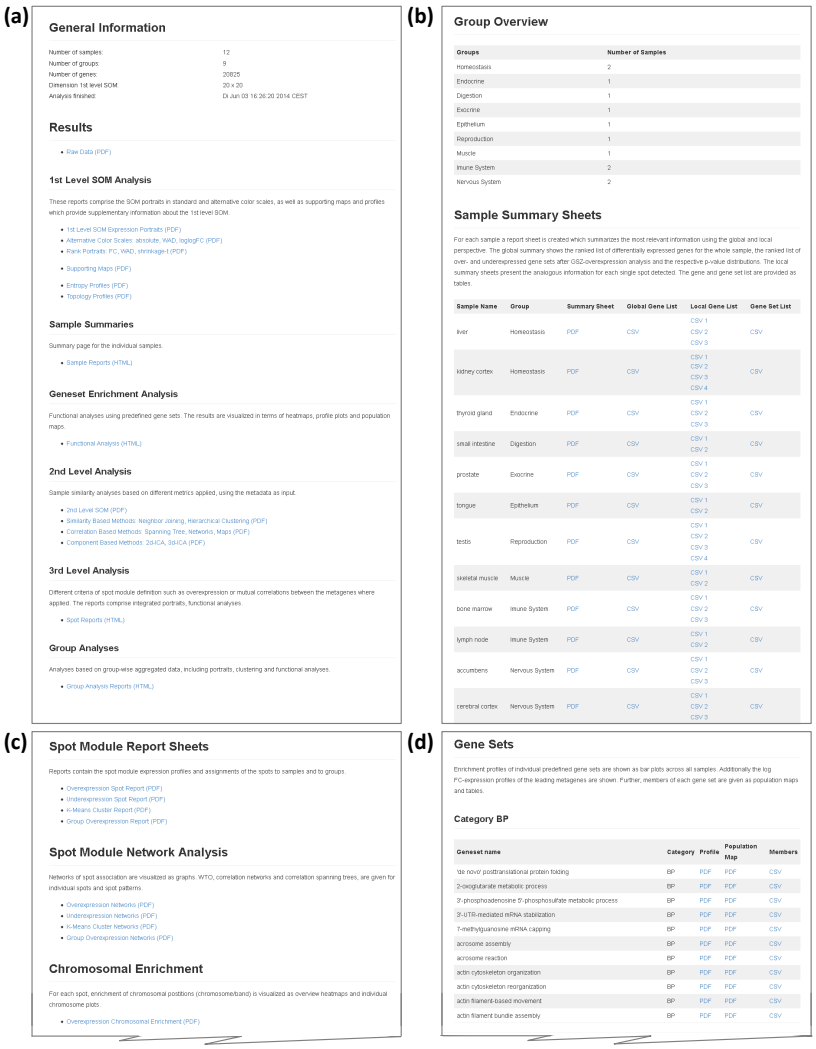

Figure 2: HTML files allow browsing all results provided by the oposSOM package: (a) The central *Summary.html* serves as starting point and contains general information and results, as well as links to other HTML files such as (b) the sample summary page, (c) the spot module summary page and (d) the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

- *note* (character): a short note shown in html summary file to give some keywords about the data or analysis parameters (default: "").
- *activated.modules* (list): activates/deactivates pipeline functionalities:
  - *reporting* (boolean): enables or disables output of pdf and csv results and html summaries (default: TRUE). When deactivated, only R workspace will be stored after analysis.
  - *primary.analysis* (boolean): enables or disables data preprocessing and SOM training (default: TRUE). When deactivated, prior SOM training results are required to be contained in the workspace environment.
  - *sample.similarity.analysis* (boolean): enables or disables diversity analyses such as clustering heatmaps, correlation networks and ICA (default: TRUE).

  - *geneset.analysis* (boolean): enables or disables geneset analysis (default: TRUE).
  - *psf.analysis* (boolean): enables or disables pathway signal flow (PSF) analysis (default: TRUE). Human gene expression data is required as input data.
  - *group.analysis* (boolean): enables or disables group centered analyses such as group portraits and functional mining (default: TRUE).
  - *difference.analysis* (boolean): enables or disables pairwise comparisons of the grous and of pairs provided by *pairwise.comparison.list* as described below (default: TRUE).

- *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.

- *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 3). Addionally, we give an estimation for runtime of the SOM training algorithm (upper limits on an Intel Core i7 system with 16GB RAM).

Size recommendation

	Number of samples				
Number of genes	< 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

	Number of samples				
Number of genes	< 100	100 - 500	500 - 1,000	1,000 - 5,000	> 5,000
< 1,000	< 1 min	< 5 min	< 5 min	< 1 h	> 1 h
1,000 - 10,000	< 5 min	< 30 min	< 2 h	< 12 h	> 12 h
10,000 - 50,000	< 30 min	< 4 h	< 10 h	< 3 d	> 3 d

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 Citing oposSOM

Please cite (Löffler-Wirth et al., 2015) when using the package.

8 Details

This document was written using:

```
> sessionInfo()

R version 4.3.0 RC (2023-04-13 r84269)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 22.04.2 LTS

Matrix products: default
BLAS:   /home/biocbuild/bbs-3.17-bioc/R/lib/libRblas.so
LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.10.0

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_GB              LC_COLLATE=C
 [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

time zone: America/New_York
tzcode source: system (glibc)

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

other attached packages:
[1] Biobase_2.60.0      BiocGenerics_0.46.0 oposSOM_2.18.0
[4] igraph_1.4.2

loaded via a namespace (and not attached):
 [1] KEGGREST_1.40.0         lattice_0.21-8          vctrs_0.6.2
 [4] tools_4.3.0             bitops_1.0-7            generics_0.1.3
 [7] stats4_4.3.0            curl_5.0.0              parallel_4.3.0
[10] tibble_3.2.1            fansi_1.0.4             AnnotationDbi_1.62.0
[13] RSQLite_2.3.1           blob_1.2.4              pkgconfig_2.0.3
[16] dbplyr_2.3.2            S4Vectors_0.38.0        RcppParallel_5.1.7
[19] scatterplot3d_0.3-43    graph_1.78.0            lifecycle_1.0.3
[22] GenomeInfoDbData_1.2.10 compiler_4.3.0          stringr_1.5.0
[25] Biostrings_2.68.0       progress_1.2.2          GenomeInfoDb_1.36.0
```

```
[28] fdrtool_1.2.17          RCurl_1.98-1.12         pillar_1.9.0
[31] crayon_1.5.2            cachem_1.0.7            pixmap_0.4-12
[34] nlme_3.1-162            tidyselect_1.2.0        digest_0.6.31
[37] stringi_1.7.12          dplyr_1.1.2             biomaRt_2.56.0
[40] fastmap_1.1.1           grid_4.3.0              cli_3.6.1
[43] magrittr_2.0.3          XML_3.99-0.14           utf8_1.2.3
[46] tsne_0.1-3.1            ape_5.7-1               prettyunits_1.1.1
[49] filelock_1.0.2          rappdirs_0.3.3          bit64_4.0.5
[52] XVector_0.40.0          httr_1.4.5              bit_4.0.5
[55] png_0.1-8               hms_1.1.3               memoise_2.0.1
[58] IRanges_2.34.0          BiocFileCache_2.8.0     fastICA_1.2-3
[61] rlang_1.1.0             Rcpp_1.0.10             glue_1.6.2
[64] DBI_1.1.3               xml2_1.3.3              R6_2.5.1
[67] zlibbioc_1.46.0
```

References

Hans Binder, Lydia Hopp, Volkan Cakir, Mario Fasold, Martin von Bergen, and Henry Wirth. Molecular phenotypic portraits - Exploring the 'OMES' with individual resolution. In Jens Allmer, editor, *Health Informatics and Bioinformatics (HIBIT), 2011 6th International Symposium*, pages 99–107. IEEE Xplore, 2012. ISBN 978-2-4673-4394-4. doi: 10.1109/HIBIT.2011.6450817.

Lydia Hopp, Kathrin Lembcke, Hans Binder, and Henry Wirth. Portraying the Expression Landscapes of B-Cell Lymphoma - Intuitive Detection of Outlier Samples and of Molecular Subtypes. *Biology*, 2(4):1411–1437, 2013a. doi: 10.3390/biology2041411.

Lydia Hopp, Henry Wirth, Mario Fasold, and Hans Binder. Portraying the expression landscapes of cancer subtypes: A glioblastoma multiforme and prostate cancer case study. *Systems Biomedicine*, 1 (2):1–23, 2013b. URL `http://www.landesbioscience.com/journals/systemsbiomedicine/toc/volume/1/issue/2/`.

Henry Löffler-Wirth, Martin Kalcher, and Hans Binder. oposSOM: R-package for high-dimensional portraying of genome-wide expression landscapes on Bioconductor. *Bioinformatics (Oxford, England)*, June 2015. ISSN 1367-4811. doi: 10.1093/bioinformatics/btv342. URL `http://www.ncbi.nlm.nih.gov/pubmed/26063839`.

Lydia Steiner, Lydia Hopp, Henry Wirth, Jörg Galle, Hans Binder, Sonja Prohaska, and Thimo Rohlf. A global genome segmentation method for exploration of epigenetic patterns. *PloS one*, 7(10), 2012.

Henry Wirth. *Analysis of large-scale molecular biological data using self-organizing maps*. Dissertation, Leipzig University, 2012. URL `http://www.qucosa.de/fileadmin/data/qucosa/documents/10129/DissertationHenryWirth.pdf`.

Henry Wirth, Markus Loffler, Martin von Bergen, and Hans Binder. Expression cartography of human tissues using self organizing maps. *BMC Bioinformatics*, 12(1):306, 2011. ISSN 1471-2105. doi: 10.1186/1471-2105-12-306. URL `http://www.biomedcentral.com/1471-2105/12/306`.

Henry Wirth, Martin von Bergen, and Hans Binder. Mining SOM expression portraits: feature selection and integrating concepts of molecular function. *BioData Mining*, 5(1):18, October 2012a. ISSN 1756-0381. doi: 10.1186/1756-0381-5-18. URL `http://www.ncbi.nlm.nih.gov/pubmed/23043905`.

Henry Wirth, Martin von Bergen, Jayaseelan Murugaiyan, Uwe Rösler, Tomasz Stokowy, and Hans Binder. MALDI-typing of infectious algae of the genus Prototheca using SOM portraits. *Journal of microbiological methods*, 88(1): 83–97, January 2012b. ISSN 1872-8359. doi: 10.1016/j.mimet.2011.10.013. URL `http://www.ncbi.nlm.nih.gov/pubmed/22062088`.