pwOmics - Pathway-based integration of time-series omics data using public database knowledge

Astrid Wachter
Medical Statistics, University Medical Center Göttingen, Germany

April 24, 2017

Contents

1 Introduction

Characterization of biological processes can be performed in great detail with the increased generation of omics data on different functional levels of the cell. Especially interpretation of time-series omics data measured in parallel with different platforms is a complex but promising task, needing consideration of

time-independent combination of omics data and additionally time-dependent signaling analysis. As each measurement technique shows a certain bias and has natural limitations in identifying full signaling responses (Yeger-Lotem, E et al., 2009), such cross-platform analysis is an up-to-date approach in order to connect biological implications on different signaling levels. Using diverse data types is expected to provide a deeper understanding of global biological functions and the underlying complex processes (Kholodenko B et al., 2012). This is why computational data analysis tools for interpretation of data from proteomics and transcriptomics measurements in parallel are needed.
pwOmics is a tool for pathway-based level-specific data comparison and analysis of single time point or time-series omics data measured in parallel. It provides individual analysis workflows for the different omics data sets (see Figure 1) and in addition enables consensus analysis of omics data.

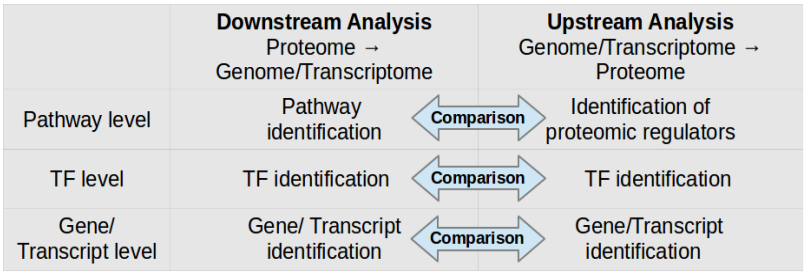

Figure 1: *pwOmics* downstream and upstream analysis.

Up to this point analysis is restricted to human species. In future an expansion of the package is possible dependent on available online open access database information.

2 Databases

As *pwOmics* is a package for data integration based on prior pathway and transcription knowledge data, it is necessary to define the databases to work with. Three different kinds of databases are necessary to do all analyses steps:

1. Pathway databases:
   The user can choose from Biocarta (Nishimura, D, 2001), Reactome (Milacic, M et al., 2012; Croft, David et al., 2014), PID (Schaefer, CF et al., 2009) from the National Cancer Institute (NCI) and KEGG (Kanehisa, M et al.,2014, Kanehisa, M and Goto, S, 2000).

2. Protein-protein interaction (PPI) database:
   STRING (Franceschini, A et al., 2013).

3. Transcription factor (TF) - target gene databases:
   The user can choose from ChEA (Lachmann, A et al., 2010), Pazar (Portales-Casamar, E et al., 2009; Portales-Casamar, E et al., 2007) and/or decide to specify an own file e.g. based on a commercial database.

The pathway database information is used to identify the pathways of the differentially abundant phosphoproteins in the downstream analysis as well as upstream protein regulators of TFs in the upstream analysis. The PPI database STRING (Franceschini, A et al., 2013) was chosen to define the protein net for the consensus analysis. TF - target gene database information is necessary for the TF identification in pathways in the downstream analysis. Additionally the upstream TFs of differentially expressed genes/transcripts are identified in the upstream analysis based on this information.
In downstream analysis the pathway gene set information is used, whereas in the upstream analysis also the pathway topology information is exploited.

The database information is downloaded internally via *STRINGdb* and *AnnotationHub* (Morgan M et al.) package. In case the author is interested also in the metadata of the pathway database and TF - target database it can be received by

```
> library(pwOmics)
> library(AnnotationHub)
> ah = AnnotationHub()
> #pathway databases
> pw = query(ah, "NIH Pathway Interaction Database")
> pw[1]

AnnotationHub with 1 record
# snapshotDate(): 2016-08-15
# names(): AH22329
# $dataprovider: NIH Pathway Interaction Database
# $species: Homo sapiens
# $rdataclass: biopax
# $title: BioCarta.owl.gz
# $description: BioCarta BioPax file from NCI Pathway Interaction Database
# $taxonomyid: 9606
# $genome: hg19
# $sourcetype: BioPax
# $sourceurl: ftp://ftp1.nci.nih.gov/pub/PID/BioPAX/BioCarta.owl.gz
# $sourcelastmodifieddate: 2009-09-09
# $sourcesize: 338343
# $tags: BioCarta, BioPax, Pathway Interaction Database
# retrieve record with 'object[["AH22329"]]'

> #TF-target databases
> chea = query(ah, "ChEA")
> chea[1]

AnnotationHub with 1 record
# snapshotDate(): 2016-08-15
# names(): AH22237
# $dataprovider: ChEA
# $species: NA
# $rdataclass: data.frame
# $title: chea-background.zip
```

```
# $description: ChEA background file, containing transcription factor data t...
# $taxonomyid: NA
# $genome: NA
# $sourcetype: Zip
# $sourceurl: http://amp.pharm.mssm.edu/result/kea/chea-background.zip
# $sourcelastmodifieddate: 2015-03-09
# $sourcesize: 3655103
# $tags: ChEA, Transcription Factors
# retrieve record with 'object[["AH22237"]]'
```

```
> pazar = query(ah, "Pazar")
> pazar[1]
```

```
AnnotationHub with 1 record
# snapshotDate(): 2016-08-15
# names(): AH22238
# $dataprovider: Pazar
# $species: NA
# $rdataclass: GRanges
# $title: pazar_ABS_20120522.csv
# $description: TF - Target Gene file from pazar_ABS_20120522
# $taxonomyid: NA
# $genome: NA
# $sourcetype: CSV
# $sourceurl: http://www.pazar.info/tftargets/pazar_ABS_20120522.csv
# $sourcelastmodifieddate: 2012-06-04
# $sourcesize: 120202
# $tags: Pazar, Transcription Factors
# retrieve record with 'object[["AH22238"]]'
```

In case you want to use TF - target gene information which is not part of the mentioned databases but e.g. part of a commercial database, a user-specified file can be used for the analysis. This file should be a '.txt' file with first column transcription factors and second column target gene symbols without a header, e.g.:

GATA-4	HAMP
c-Jun	IL18
NF-kappaB	TLR2
MYB	LTB
FOXO1A	TGFBR1
...	...

The STRING PPI-information is downloaded automatically while processing and analyzing the data: The *STRINGdb* package (Franceschini, A et al., 2013) is used here.

3 Example dataset

The example dataset used here for demonstration purposes comprises whole genome time course microarray data at time points 0, 1, 4, 8, 13, 18 and 24 hr

after stimulation. The complementary phosphoproteomics data was measured at time points 0.25, 1, 4, 8, 13, 18 and 24 hours after stimulation. Preprocessing of data is presupposed with given lists of significant genes and phosphoproteins for each time point as logarithmized expression ratios relative to the time 0 hr controls.

4 Data pre-processing

pwOmics is a package for secondary data analysis, i.e. it needs already pre-processed data as input for the analysis. The input required is

1. a list of all phosphoprotein IDs measured,

2. a list of all gene/transcript IDs measured,

3. a list of differentially abundant phosphoproteins + log fold changes,

4. a list of differentially expressed genes/transcripts + log fold changes.

The IDs need to be gene symbols, both for phosphoprotein and gene/transcript data. In case time-series data is analyzed inputs 3. and 4. needs to be specified for each time point. It is absolutely necessary, that all phosphoproteins and genes/transcript in inputs 3. and 4. are part of the lists of all phosphoprotein IDs and all gene/transcript IDs, respectively.

The OmicsData object is the format used for data analysis in *pwOmics* package. It contains a list of four main elements:

1. OmicsD - here the omics data set, its description and the results are stored

2. PathwayD - here the chosen pathway databases and the generated Biopax model is stored

3. TFtargetsD - here the chosen TF-target gene databases and the combined TF-target gene information is stored

4. Status - The status variable equals '1' in case not all information needed for the analysis is read in yet and '2' after identification of the first upstream/downstream signaling levels. As the enrichment step is not necessarily part of the analysis and dependent on the pathway database and the TF-target gene database the identification of signaling molecules in further levels might not be successful, the status variable is not used in the further analysis.

Thus *pwOmics* reads in the omics data set provided by the user to the first element of the OmicsData object and further on stores all the results in this part as well.

This is why the user has to provide the omics data set in a special format: A list

needs to be generated with a phosphoprotein list named 'P' as first element and a gene/transcript list named 'G' as second element. These lists contain as first element a data frame with all (unique) phosphoprotein IDs and gene/transcript IDs in the first column, respectively, and as second element a list with data frames for each time point of measurement. The data frames have two columns with the first one containing the differentially abundant/expressed phosphoproteins or genes/transcripts as gene symbols and the second column containing the corresponding log fold changes, e.g.:

```
> data(OmicsExampleData)
> OmicsExampleData
```

Generated as in the following example:

```
OmicsExampleData = list(P = list(allPIDs,
                   list(PIDstp0.25, PIDstp1, PIDstp4, PIDstp8,
                        PIDstp13, PIDstp18, PIDstp24)),
                        G = list(allGIDs,
                   list(GIDstp1, GIDstp4, GIDstp8, GIDstp13,
                        GIDstp18, GIDstp24)))
```

```
> head(OmicsExampleData$P[[2]][[1]])
```

```
  GeneSymbol     X15min
1     MRPS17  0.6976049
2      RPS12 -1.0297977
3     SLC3A2 -1.2623327
4       RPL8  0.8304820
5       ACTB -2.4914461
6      ALDOA  0.8637013
```

In case the user only wants to analyze omics data from a single time point just one data frame has to be specified.
The time points do not have to be the same for phosphoprotein and gene/transcript data and need to be specified when reading in the omics data set separately via the 'tp_prots' and 'tp_genes' parameters of the 'readOmics' function.

```
> data_omics = readOmics(tp_prots = c(0.25, 1, 4, 8, 13, 18, 24),
+                        tp_genes = c(1, 4, 8, 13, 18, 24),
+                        OmicsExampleData,
+                        PWdatabase = c("biocarta", "kegg", "nci",
+                                       "reactome"),
+                        TFtargetdatabase = c("userspec"))
```

If data from a single timepoint measurement should be analyzed the user simply assigns the experiment number '1' for these parameters:

```
#for single time point data set:
omics = list(P = list(allPIDs, list(PIDs_1)),
             G = list(allGIDs, list(GIDs_1)))
```

```
data_omics = readOmics(tp_prots = c(1),
                    tp_genes = c(1),
                    OmicsExampleData,
                    PWdatabase = c("biocarta", "kegg", "nci",
                                   "reactome"),
                    TFtargetdatabase = c("userspec"))
```

Additionally the selected databases have to be specified.

The stored information can be easily accessed via the following functions:

```
> getOmicsTimepoints(data_omics)
> head(getOmicsallProteinIDs(data_omics))
> head(getOmicsallGeneIDs(data_omics))
> head(getOmicsDataset(data_omics, writeData = FALSE)[[1]])
```

5 Individual analysis

As shown in Figure 1 the analysis is based on an individual analysis of the phosphoproteomic and the genomic/transcriptomic data. The downstream analysis and upstream analysis are described in the following subsections.
Prior to that the database information has to be read in. In a first step the phosphoprotein information about downstream activating/inactivating effects should be read in.

```
data_omics = readPhosphodata(data_omics, phosphoreg)
```

The parameter 'phosphoreg' should be a tab-separated .txt file with two columns, but no header. The first column should include the HUGO gene symbols of the phosphoproteins with known downstream effects, the second column should include either activating (1) or inactivating (-1) downstream effect coding upon phosphorylation of the corresponding phosphoprotein. So far, there is no possibility to include multiple phosphorylation events individually. However, if no information for a phosphoprotein is available in this file, any up/downregulation match is allowed for the molecules identified in the intersection analyses.
In a second step the TF- target information can be made accessible to the 'OmicsData' object by:

```
data_omics = readTFdata(data_omics, TF_target_path)
```

Via the 'TF_target_path' parameter a path of a user-specified file specifying transcription factors and corresponding target genes can be given. This should be a tab-separated .txt file with two columns, but no header, giving HUGO gene symbols of transcription factors in the first column and gene symbols of corresponding target genes in a second column. This information can be used additionally to the selected database content.
Thirdly, the 'readPWdata' function takes the 'OmicsData' object with the provided information about the omics data set and the path of the prepared '.RData' genelists from the pathway databases (see Section 2) or automatically generates the corresponding genelists of the pathway data if 'loadgenelists =

FALSE'. In this step the automatic definition of internal IDs for different pathway databases is necessary, which are stored in a new biopax model in the 'OmicsData' object.

```
data_omicsPW = readPWdata(data_omics,
               loadgenelists = FALSE)
```

As the process of generating genelists with these IDs can take some time - especially for rather big databases such as Reactome (Milacic, M et al., 2012; Croft, D et al., 2014) - the genelists for the different databases are automatically stored in the working directory and can be reused in another analysis when the corresponding path containing these files is given to the 'readPWdata' function as loadgenelists parameter.

```
data_omics = readPWdata(data_omics,
             loadgenelists = "Genelist_reactome.RData")
```

5.1 Downstream analysis

The downstream analysis is starting with the provided phosphoproteomic data (either single time point data or time-series data). The first step is the identification of downstream regulation influence of phosphoproteins based on the file read in by 'readPhosphodata' function.

```
data_omics = identifyPR(data_omics)
```

The second step is the identification of the pathways in which the differentially abundant phosphoproteins play a role. *pwOmics* performs this searching step on the basis of the provided phosphoproteomic data set and the selected pathway database(s). As this function exploits pathway knowledge, it is necessary to be in the working directory of the genelists that were generate by function 'readPWdata'.

```
data_omics = identifyPWs(data_omics)
```

Following the workflow the next step is the identification of the transcription factors in these pathways, which is done with the information provided by the chosen TF-target gene database.

```
data_omics = identifyPWTFTGs(data_omics)
```

For use of this function the working directory should still contain the previously generated genelists.

The results of the downstream analysis can be easily accessed by the following functions:

```
getDS_PWs(data_omics)
getDS_TFs(data_omics)
getDS_TGs(data_omics)

#Access biopax model generated newly on basis of selected
#pathway databases:
getBiopaxModel(data_omics)
```

5.2 Upstream analysis

The upstream analysis is starting with the provided gene/transcript data (either single time point data or time-series data). It first of all identifies the upstream TFs of the differentially expressed genes/transcripts. This step is done with the provided/selected information of TF-target gene pairs.
Given this information, the identification of upstream TFs can be done:

```
data_omics = identifyTFs(data_omics)
```

Upstream of the TFs the potential regulator proteins can be identified with the following function:

```
data_omics = identifyRsofTFs(data_omics,
                        noTFs_inPW = 1, order_neighbors = 10)
```

The identification of potential upstream regulators is done in the following way:

1. Identification of the pathways the previously identified TFs are part of.

2. Selection of pathways according to the user-specified parameter 'noTFs_inPW': Only those pathways are considered in further analysis with at least this number of TFs present in the pathway.

3. Upstream regulators are identified for these TFs. This is done by finding first for each TF the pathway neighborhood according to the user-specified parameter 'order_neighbors'. This parameter specifies the order of the identified pathway neighborhood. Then the intersection of all identified neighborhoods for all TFs in a pathway is determined. The resulting pathway node set is defined here as the set of potential regulator proteins.

In case the pathways under consideration do not have pathway components in the downloaded biopax model, this will be indicated with a warning. This warning can be ignored by the user in regard to the following analysis steps.
The results of the upstream analysis can be accessed with the following functions:

```
getUS_TFs(data_omics)
getUS_PWs(data_omics)
getUS_regulators(data_omics)
```

6 Consensus analysis

The consensus analysis combines the results from upstream and downstream analysis by constituting in particular the comparative analysis of the results of the two different data sets on each cellular level individually. The intersection analysis thus simply compares the results of the separate upstream and downstream analyses. The static consensus analysis allows to determine static consensus graphs for each time point measured in parallel. Finally, the consensus-based dynamic analysis provides the user with one final dynamic network obtained from the data changes over time based on dynamic bayesian network

inference. The consensus-based dynamic analysis is only conductable with time-series data sets measured for both phosphoproteome and genome/transcriptome data in parallel.

6.1 Intersection analysis

The intersection analysis of *pwOmics* is a simple comparative analysis of the results of upstream and downstream analysis. Thus, it enables a comparison of single time point data and time-series data, the latter also for non-corresponding time points in the different data sets. The comparison is possible on the three different functional levels considered in this package: On the pathway level, the transcription factor level and gene/transcript level. The parameters 'updown' and 'phospho' enable to define, whether the direction of regulation should be considered in the comparison and whether the phosphosite information should be considered. The consensus molecule sets (C) for each of the molecular levels (protein level, transcription factor level and gene/transcript level) are defined as:

1. if phosphoprotein downstream activity is induced:
   $C = (X' \uparrow \cap X \uparrow) \cup (X' \downarrow \cap X \downarrow)$

2. if phosphoprotein downstream activity is inhibited:
   $C = (X' \uparrow \cap X \downarrow) \cup (X' \downarrow \cap X \uparrow)$

3. if no information is available for a specific phosphoprotein:
   $C = (X' \uparrow \cap X \uparrow) \cup (X' \downarrow \cap X \downarrow) \cup (X' \uparrow \cap X \downarrow) \cup (X' \downarrow \cap X \uparrow)$

with $\uparrow$ = upregulated and $\downarrow$ = downregulated and X = molecules in downstream analysis, X' = molecules in upstream analysis.

```
getProteinIntersection(data_omics, tp_prot = 4, tp_genes = 18,
                       updown = TRUE, phospho = TRUE)
getTFIntersection(data_omics, tp_prot = 13, tp_genes = 13,
                       updown = TRUE, phospho = TRUE)
getGenesIntersection(data_omics, tp_prot = 24, tp_genes = 24,
                        updown = TRUE, phospho = TRUE)
```

These functions not only enable a comparison of the same timepoints on the distinct levels, but for time-series data sets also for non-matching time points: With the time resolution of measuring omics data in most cases being predefined by expected signaling changes and financial limitations the potential in the interpretation of the results is strongly confined to the experimental design decisions. Thus, measured signaling changes, which naturally consist of a superposition of diverse time-scales of transcriptional and translational processes and comprehend diverse frequency patterns (Yosef, N and Regev A(2011)), are dependent on the sampling. This means for some of the signaling axes it might be the case, that

- changes are not detected at all as their rate is too high,
- some are represented in the data and

- some might be so slow that their change is not considered significant and thus are excluded from analysis.

As the corresponding signaling changes are not expected to be seen at the same time point in phosphoproteome data and gene/transcript data it is necessary to enable also the comparison of non-corresponding time points.
The option to compare non-corresponding time points cannot account for the changes not captured during measurement, however, it gives the possibility to consider also the time needed for regulatory control mechanisms in the interpretation of the measurement results.
In case the user wants to compare the corresponding time points on the three levels simultaneously he can do so by using the following function:

```
gettpIntersection(data_omics, updown = TRUE, phospho = TRUE)
```

The parameters 'updown' and 'phospho' allow to select whether different regulation directions in the data sets (phosphoproteome vs. transcriptome) should be filtered out in the comparison, and whether the downstream regulation influence of phosphoproteins read in with the 'readPhosphodata' function should be considered in the comparison.

6.2 Static consensus analysis

The static consensus analysis goes one step ahead and integrates the results gained from the comparative analysis of the corresponding time points to a consensus net for each time point. The change of this consensus net over time gives a first insight into the changes seen statically at the different time points. However, the static consensus nets do not yet include information gathered over time - as it is the case for the consensus-based dynamic analysis (see Section 6.3). This is why the static consensus analysis is also applicable for single time point measurements.

The static consensus analysis is conducted by generation of a Steiner tree (Kleinberg, J and Tardos E, 2006) on basis of consensus proteins and TFs identified in downstream and upstream analysis for each corresponding time point. The underlying graph used is the protein-protein-interaction (PPI) graph from the STRING database reduced to the connected nodes. The consensus proteins and TFs are considered as terminal nodes and are connected via a shortest path-based approximation of the Steiner tree algorithm (Takahashi, H and A Matsuyama, 1980; Sadeghi, A and Fröhlich, H, 2013) across the reduced PPI-STRING-graph. Subsequently knowledge of TF-target gene pairs from the chosen database is used to expand the graph with matching genes/transcripts from both upstream and downstream analysis. In case the consensus graph contains corresponding proteins and genes/transcripts, feedback loops are added automatically.

```
consensusGraphs = staticConsensusNet(data_omics, updown = TRUE,
                                     phospho = TRUE)
```

The parameters 'updown' and 'phospho' give the same filtering options as in the intersection analysis.

6.3 Consensus-based dynamic analysis

Unlike the static consensus analysis, the consensus-based dynamic analysis takes into consideration also the signaling changes over time by applying dynamic bayesian network inference. The packages used for the consensus-based dynamic analysis are *longitudinal* (Opgen-Rhein, R and Strimmer, K, 2006; Opgen-Rhein R and Strimmer K, 2006) to adjust the format of the data and the actual network inference part is done via the *ebdbNet* (Rau, A et al., 2010) package. This package includes an iterative empirical Bayesian procedure with a Kalman filter estimating the posterior distributions of the network parameters. The defined prior structure of the network is used for a straightforward estimation of hyperparameters via an expectation maximization (EM)-like algorithm and the dimension of the hidden states are determined via the singular value decomposition (SVD) of a block-Hankel matrix.

The nodes included into the network inference step are nodes which are part of the static consensus graphs from corresponding time points of the two different measurement types, i.e.

1. proteins identified in upstream and downstream analysis at the same time points,

2. Steiner nodes identified via static consensus analysis,

3. TFs identified in upstream and downstream analysis at the same time points and

4. genes/transcripts identified in upstream and downstream analysis at the same time points.

However, all measured time points of these nodes are taken into consideration. To apply dynamic network inference a reasonable number of measurements needs to be available. As in most cases of parallel phosphoprotein and gene/transcript measurements only very few corresponding time steps are available it is necessary to artificially introduce additional time steps. This is done by generating smoothing splines applied on the log fold changes provided by the user under the simplifying assumption of a gradual change of signaling between the different time points.

This assumption, however, has to be applied consciously and carefully, as there might be higher frequency signaling components superimposed (see for a comprehensive analysis of temporal dynamics of gene expression (Yosef N and Regev A, 2011). In theory a signal has to be sampled two times its maximal frequency in order to be able to reconstruct it exactly from time discrete measurements (Nyquist-Shannon sampling theorem (Shannon, CA and Weaver W, 1949; Nyquist, H, 1928). This means only exact interpretation of those signaling axes are possible that have a frequency which is smaller than half of the sampling frequency. However, under certain preconditions on signal structure and the sampling operator reconstruction of the original signal can be done with a lower sampling rate (Blumensath, T and Michael ED, 2009). This is an interesting

starting point for a more comprehensive dynamic analysis of the expected signals and the sampling needed for an extensive data mining of omics data sets measured in parallel, but exceeds the scope of this package.

The number of time points generated additionally via smoothing splines is based on simulation results of *ebdbNet* analysis for median area under the curve (AUC) values of receiver operating characteristic (ROC) curves: In their results it was shown that a plateau at around 50 to 75 time points was reached. Thus in *pwOmics* 50 time points are predicted with smoothing splines in order to apply dynamic bayesian network inference on omics data sets measured in parallel.

After generation of these time points a block-Hankel matrix of autocovariances is constructed based on the time series abundance/expression data. For this the user needs to provide the *laghankel* parameter, specifying the maximum relevant time lag to be used in constructing the block-Hankel matrix. With a singular value decomposition (see function 'hankel' of *ebdbNet* package) the number of hidden states can be determined. Here, the user can specify the *cutoffhankel* parameter to choose the cutoff to determine the desired percent of total variance explained by the singular values. Additional parameters on convergence criteria and iterations performed can be specified. For further details the user is referred to (Rau, A et al., 2010).

```
library(ebdbNet)
library(longitudinal)
dynInferredNet = consDynamicNet(data_omics, consensusGraphs,
                 laghankel = 3, cutoffhankel = 0.9)
```

7 Time profile clustering

An additional analysis option is clustering of co-regulation patterns over time. It provides information about the signaling molecules with common dynamic behaviour and thus allows to draw conclusions in terms of signaling chronology. Time profile clustering is performed as soft clustering based on the *Mfuzz* package (Futschik, M, 2012). The advantage of this clustering method is that a protein, TF or gene/transcript can be assigned to several clusters, thus reducing the sensitivity to noise and the information loss hard clustering exhibits. It is implemented as fuzzy c-means algorithm (Hathaway, R and Bezdek, J, 1986) and iteratively optimizes the objective function to minimize the variation of objects within the clusters. The user needs to provide a 'min.std' threshold parameter if proteins or genes/transcripts with a low standard deviation should be excluded. In addition the maximum number of cluster centers which should be tested in the 'minimum distance between cluster centroid test' has to by given. This number is used as initial number to determine the data-specific maximal cluster number based on the number of distinct data points. For more details see (Futschik, M, 2012) and (Schwämmle, V and Jensen ON, 2010).

```
library(Mfuzz)
fuzzyClusters = clusterTimeProfiles(dynInferredNet,
                                    min.std = 0,
                                    ncenters = 12)
```

```
fuzzyClusters$size
fuzzyClusters$cluster
```

8 Identification of signaling axes

Based on the provided data it is also possible to follow the signal through the different cellular levels individually, such that individual signaling axes can be identified.
The user can seach for individual axes downstream of a certain phosphoprotein by using the 'findSignalingAxes' function and indicating the time point of interest. The time point needs to comply with the ones read in in the 'readOmics' function.

```
MAPK1_axis1 = findSignalingAxes(data_omics,
                                phosphoprot = "MAPK1", tpDS = 1)
MAPK1_axis18 = findSignalingAxes(data_omics,
                                phosphoprot = "MAPK1", tpDS = 18)
BLNK_axis13 = findSignalingAxes(data_omics,
                                phosphoprot = "BLNK", tpDS = 13)
```

The output is a list of pathways that are identified for this time point downstream of the phosphoprotein of interest. Sublists give information on the transcription factors in these pathways, the direction of regulation (up/down), the target genes and their matching transcripts from the transcriptome data set.
Based on these results the user can produce a summarized results table containing those target genes of the axis, which are matching the transcriptome data, the direction of their regulation coming from the phosphoproteome data set (up/down), the direction of regulation coming from the transcriptome data set (up/down) and the summarized complying information:

```
MAPK1_transcripts = get_matching_transcripts(data_omics,
                                             MAPK1_axis1)
```

If the user is interested in an overall analysis of all signaling axes in the data sets, he can apply:

```
generate_DSSignalingBase(data_omics,
                         timepoints = c(0.25, 1, 4, 8, 13, 18, 24))
```

with chosen time points. This function creates a folder structure for each consensus protein in the working directory, which is filled with .RData objects and .csv tables that carry the information for the individual time points of interest.

9 Visualization

To complement the results from the different comparisons and analyses (accessible via the 'get...' functions) the *pwOmics* package provides visualization functions for the different analyses. The consensus graphs of the static analysis for one or more corresponding time points can be plotted with the following function (see Figure 2 and Figure 3):

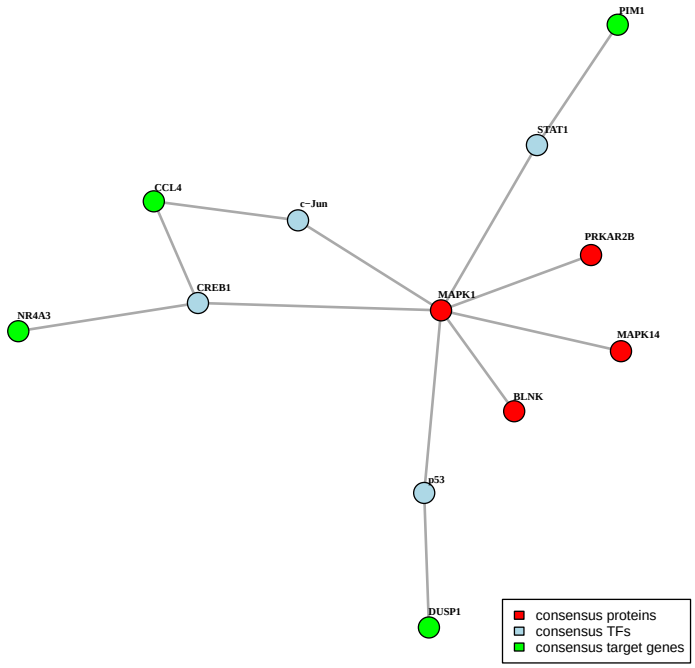

Figure 2: *pwOmics* static consensus graph: Time point 1 hr.

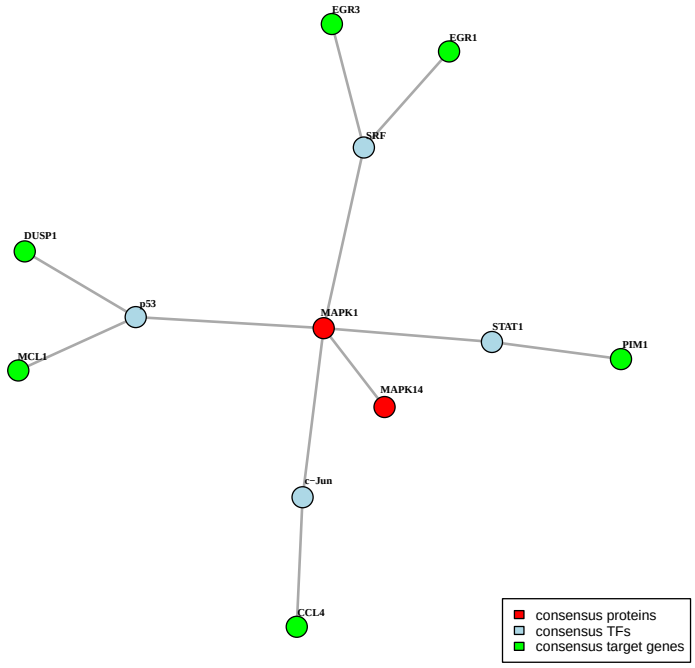

Figure 3: *pwOmics* static consensus graph: Time point 24 hrs.

```
plotConsensusGraph(consensusGraphs, data_omics)
```

The consensus-based dynamic analysis result can be visualized as follows (see Figure 4):

```
plotConsDynNet(dynInferredNet, sig.level = 0.7)
```

Here, the parameter 'sig.level' is the significance level used as cutoff for plotting edges in the network and has to be specified in the range between 0 and 1. Furthermore the user can indicate if unconnected nodes should be removed and provide additional *igraph* (Csardi, G and Nepusz T, 2006) layout parameters.

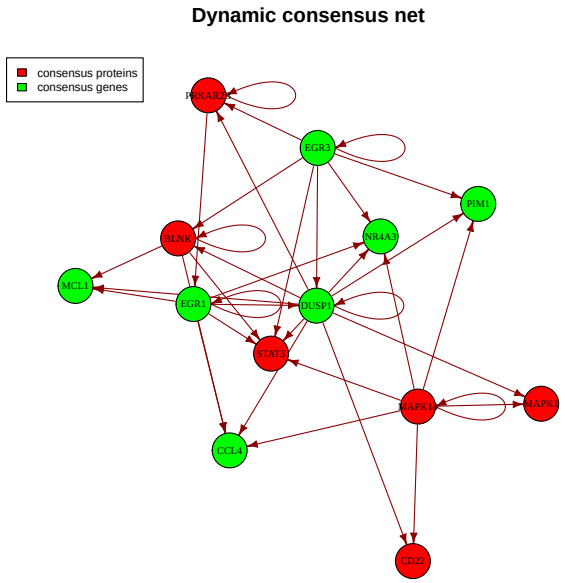

Figure 4: *pwOmics* dynamic network graph.

However, as the user can access the networks easily the *tkplot* function from the *igraph* R package is a nice interactive graph drawing alternative. In addition plot parameters can be easily changed as the result networks are of class 'igraph'. In order to plot the results from time profile clustering (see Figure 5) the following function can be used:

```
plotTimeProfileClusters(fuzzyClusters)
```

The different colours represent the different clusters. The legend is only shown if the number of genes and proteins is not too large. Otherwise the user can easily access this information by having a look to the output of the 'clusterTimeProfiles' function which provides information about cluster centers, the number of data points in each cluster of the closest hard clustering, cluster indices, and

additional parameters explained in detail in the 'mfuzz' documentation. In the legend the attachments '_g' and '_p', respectively, indicate, if the node originally derives from phosphoprotein or gene/transcript measurements.

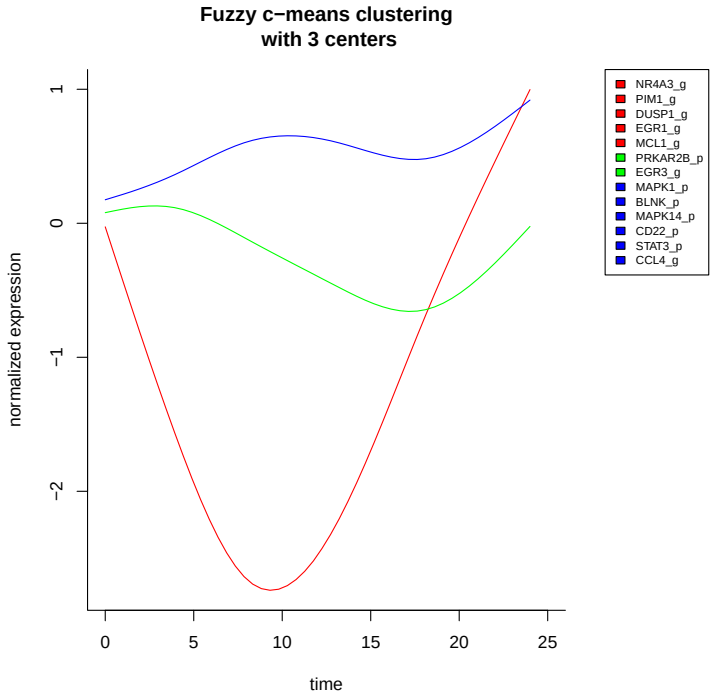

Figure 5: *pwOmics* time profile clusters.

It is furthermore possible to investigate temporal correlations of phosphoprotein and transcriptome measurements.

```
temp_correlations(consensusGraphs[[1]],
    timepointsprot = c(1,4,8,13,18,24),
    timepointstrans = c(1,4,8,13,18,24),
    foldername = "~/TempCorr_",
    trans_sign = "~/transcriptome_signif.txt",
    trans_sign_names = c("FC_1", "FC_4", "FC_8", "FC_13",
                         "FC_18", "FC_24"),
    phospho_sign = "~/phospho_anno.txt",
    phospho_sign_names = c("rat_1", "rat_4","rat_8","rat_13",
                           "rat_18", "rat_24"))
```

This function needs as input the static consensus graph with the consensus proteins of interest, the protein and transcriptome measurement time points for which correlation should be investigated, the name of the folder that should be generated as results folder, the phosphoproteome and transcriptome measurement files and the corresponding column names for the selected time points. It is necessary that the gene symbols read in in the 'readOmics' function are reused here. In the tab-delimited phosphoproteome measurement file (.txt) additional

phosphoprotein information can be stored in columns ‘Amino.acid’, ‘Position’ and ‘Multiplicity’. This allows to compare correlations for individual phosphorylations.

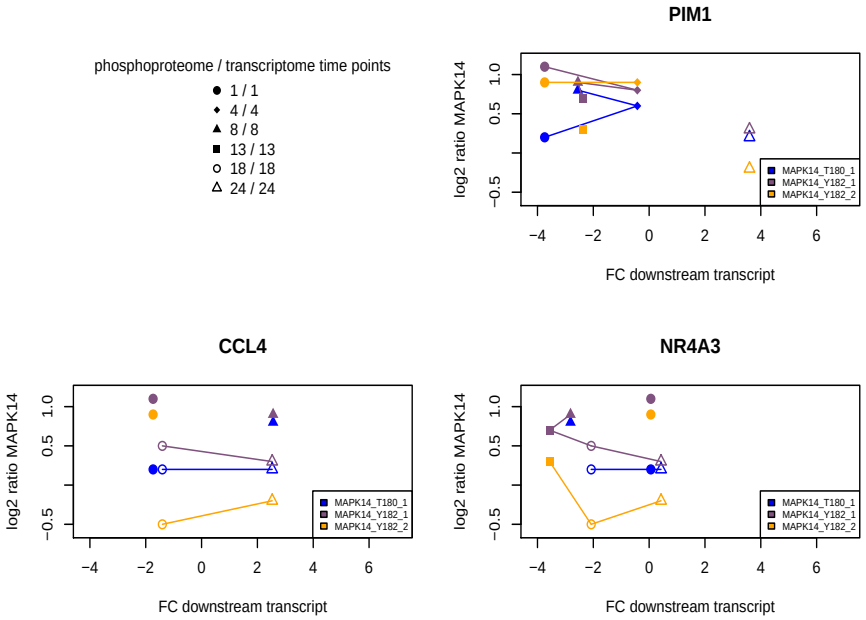

Figure 6: Temporal correlation of MAPK14 with transcripts identified to be resulted downstream.

10 References

- Yeger-Lotem, Esti et al. (2009). “Bridging high-throughput genetic and transcriptional data reveals cellular responses to alpha-synuclein toxicity”. In: Nature Genetics 41.3, pp. 316–323. issn: 1546-1718. doi: 10.1038/ng.337.
- Boris Kholodenko et al. (2012). “Computational Approaches for Analyzing Information Flow in Biological Networks”. In: Science Signaling 5, re1.
- Nishimura, Darryl (2001). “BioCarta”. In: Biotech Software & Internet Report 2.3, pp. 117–120. issn: 1527-9162. doi: 10.1089/152791601750294344. url: http://online.liebertpub.com/doi/abs/10.1089/152791601750294344 (visited on 01/02/2015).
- Milacic, Marija et al. (2012). “Annotating cancer variants and anti-cancer therapeutics in reactome”. In: Cancers 4.4, pp. 1180–1211. issn: 2072-6694. doi: 10.3390/cancers4041180.

- Croft, David et al. (2014). "The Reactome pathway knowledgebase". In: Nucleic Acids Research 42.Database issue, pp. D472–477. issn: 1362-4962. doi: 10.1093/nar/gkt1102.
- Schaefer, Carl F. et al. (2009). "PID: the Pathway Interaction Database". In: Nucleic Acids Research 37.Database issue, pp. D674–679. issn: 1362-4962. doi: 10.1093/nar/gkn653.
- Kanehisa, Minoru et al. (2014). "Data, information, knowledge and principle: back to metabolism in KEGG". In: Nucleic Acids Research 42.Database issue, pp. D199–205. issn: 1362-4962. doi: 10.1093/nar/gkt1076.
- Kanehisa, M. and S. Goto (2000). "KEGG: kyoto encyclopedia of genes and genomes". In: Nucleic Acids Research 28.1, pp. 27–30. issn: 0305-1048. 19
- Franceschini, Andrea et al. (2013). "STRING v9.1: protein-protein interaction networks, with increased coverage and integration". In: Nucleic Acids Research 41. Database issue, pp. D808–D815. issn: 0305-1048. doi: 10.1093/ nar / gks1094. url: http : / / www . ncbi . nlm . nih . gov / pmc / articles / PMC3531103/ (visited on 01/02/2015).
- Lachmann, Alexander et al. (2010). "ChEA: transcription factor regulation inferred from integrating genome-wide ChIP-X experiments". In: Bioinformatics (Oxford, England) 26.19, pp. 2438–2444. issn: 1367-4811. doi: 10.1093/bioinformatics/btq466.
- Portales-Casamar, Elodie et al. (2009). "The PAZAR database of gene regu- latory information coupled to the ORCA toolkit for the study of regulatory sequences". In: Nucleic Acids Research 37.suppl 1, pp. D54–D60. issn: 0305-1048, 1362-4962. doi: 10.1093/nar/gkn783.
- Portales-Casamar, Elodie et al. (2007). "PAZAR: a framework for collection and dissemination of cis-regulatory sequence annotation". In: Genome Biology 8.10, R207. issn: 1465-6906. doi: 10.1186/gb- 2007-8- 10- r207.
- Morgan M Carlson M, Tenenbaum D and Arora S. AnnotationHub: Client to access AnnotationHub resources. R package version 2.22.0.
- Yosef, Nir and Aviv Regev (2011). "Impulse control: Temporal dynamics in gene transcription". In: Cell 144.6, pp. 886–896. issn: 0092-8674. doi: 10. 1016/j.cell.2011.02.015. url: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3148525/ (visited on 01/03/2015).
- Kleinberg, J and E Tardos (2006). Algorithm Design. Pearson, Boston, MA.
- Takahashi, H and A Matsuyama (1980). "An approximate solution for the Steiner problem in graphs". In: Math. Jap. 24, pp. 573–577.
- Sadeghi, Afshin and Holger Fröhlich (2013). "Steiner tree methods for optimal sub-network identification: an empirical study". In: BMC Bioinformatics 14, p. 144.

- Opgen-Rhein, R and K Strimmer (2006). "Using regularized dynamic correlation to infer gene dependency networks from time-series microarray data". In: Proceedings of the 4th International Workshop on Computational Systems Biology, WCSB 2006, pp. 73–76.
- Rainer Opgen-Rhein, Korbinian Strimmer (2006). "Inferring gene dependency networks from genomic longitudinal data: a functional data approach". In: REVSTAT – Statistical Journal Volume 4.1, pp. 53–65. issn: 1645-6726.
- Rau, Andrea et al. (2010). "An empirical Bayesian method for estimating biological networks from temporal microarray data". In: Statistical Applications in Genetics and Molecular Biology 9.1.
- Shannon, CA and W Weaver (1949). The mathematical theory of communication. University of Illinois Press.
- Nyquist, H (1928). "Certain topics in telegraph transmission theory". In: Transactions of the A. I. E. E. pp. 617–644.
- Blumensath, Thomas and Michael E. Davies (2009). "Sampling Theorems for Signals From the Union of Finite-Dimensional Linear Subspaces". In: IEEE Transactions on Information Theory 55.4, pp. 1872–1882.
- Futschik, Matthias (2012). Mfuzz: Soft clustering of time series gene expression data. R package version 2.22.0. url: http://itb.biologie.huberlin. de/ futschik/software/R/Mfuzz/.
- Hathaway, R and Bezdek, J (1986). "Local convergence of the fuzzy c-means algorithm". In: Pattern Recognition 19, pp. 477-480.
- Schwämmle, Veit and Ole Nørregaard Jensen (2010). "A simple and fast method to determine the parameters for fuzzy c–means cluster analysis". In: Bioinformatics 26.22, pp. 2841–2848. issn: 1367-4803, 1460-2059. doi: 10. 1093/bioinformatics/btq534. url: http://bioinformatics.oxfordjournals.org/content/26/22/2841 (visited on 01/03/2015).
- Csardi, Gabor and Tamas Nepusz (2006). "The igraph software package for complex network research". In: InterJournal Complex Systems, p. 1695. url: http://igraph.org.

11 Session Information

```
> sessionInfo()

R version 3.4.0 (2017-04-21)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 16.04.2 LTS

Matrix products: default
BLAS: /home/biocbuild/bbs-3.5-bioc/R/lib/libRblas.so
LAPACK: /home/biocbuild/bbs-3.5-bioc/R/lib/libRlapack.so

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

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

loaded via a namespace (and not attached):
[1] compiler_3.4.0 tools_3.4.0
```