BioNet Tutorial

Daniela Beisser and Marcus Dittrich

May 3, 2016

Abstract

The first part of this tutorial exemplifies how an integrated network analysis can be conducted using the BioNet package. Here we will integrate gene expression data from different lymphoma subtypes and clinical survival data with a comprehensive protein-protein interaction (PPI) network based on HPRD. This is shown first in a quick start and later in a more detailed analysis. The second part will focus on the integration of gene expression data from Affymetrix single-channel microarrays with the human PPI network.

1 Quick Start

The quick start section gives a short overview of the essential BioNet methods and their application. A detailed analysis of the same data set of diffuse large B-cell lymphomas is presented in section **??** .
The major aim of the presented integrated network analysis is to identify modules, which are differentially expressed between two different lymphoma subtypes (ABC and GCB) and simultaneously are risk associated (measured by the survival analysis).
First of all, we load the BioNet package and the required data sets, containing a human protein-protein interaction network and p-values derived from differential expression and survival analysis.

```
> library(BioNet)
> library(DLBCL)
> data(dataLym)
> data(interactome)
```

Then we need to aggregate these two p-values into one p-value.

```
> pvals <- cbind(t = dataLym$t.pval, s = dataLym$s.pval)
> rownames(pvals) <- dataLym$label
> pval <- aggrPvals(pvals, order = 2, plot = FALSE)
```

Next a subnetwork of the complete network is derived, containing all the proteins which are represented by probesets on the microarray. And self-loops are removed.

```
> subnet <- subNetwork(dataLym$label, interactome)
> subnet <- rmSelfLoops(subnet)
> subnet
```

```
A graphNEL graph with undirected edges
Number of Nodes = 2559
Number of Edges = 7788
```

To score each node of the network we fit a Beta-uniform mixture model (BUM) [**?**] to the p-value distribution and subsequently use the parameters of the model for the scoring function [**?**]. A false-discovery rate (FDR) of 0.001 is chosen.

```
> fb <- fitBumModel(pval, plot = FALSE)
> scores <- scoreNodes(subnet, fb, fdr = 0.001)
```

Here we use a fast heuristic approach to calculate an approximation to the optimal scoring subnetwork. An optimal solution can be calculated using the `heinz` algorithm [**?**] requiring a commercial CPLEX license, see section **??** and **??** for installation.

```
> module <- runFastHeinz(subnet, scores)
> logFC <- dataLym$diff
> names(logFC) <- dataLym$label
```

Both 2D and 3D module visualization procedures are available in BioNet. For a 3D visualization, see section **??**. Alternatively, the network could be easily exported in Cytoscape format, see section **??**.

```
> plotModule(module, scores = scores, diff.expr = logFC)
```

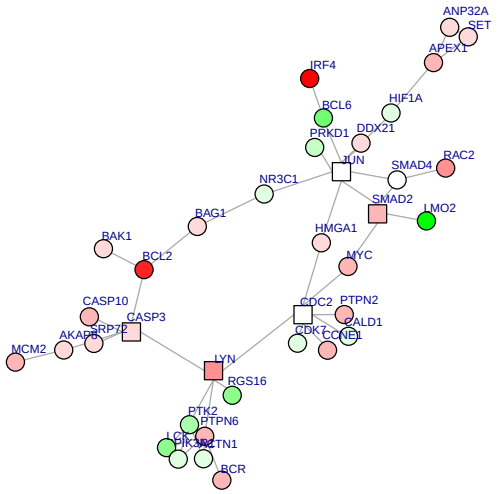

Figure 1: Resultant functional module. Differential expression between ABC and GCB B-cell lymphoma is coloured in red and green, where green shows an upregulation in ACB and red an upregulation in GBC. The shape of the nodes depicts the score: rectangles indicate a negative score, circles a positive score.

2 Heuristics to Calculate High-scoring Subnetworks

To calculate high-scoring subnetworks without an available CPLEX license a heuristics is included in the BioNet package. The following depicts a short outline of the algorithm:

1. In the first step all positive connected nodes are aggregated into meta-nodes.
2. By defining an edge score based on the node's scores that are on the endpoints of an edge, the node scores are transfered to the edges.
3. On these edge scores a minimum spanning tree (MST) is calculated.
4. All paths between positive meta-nodes are calculated based on the MST to obtain the negative nodes between the positives.
5. Upon these negative nodes again a MST is calculated from which the path with the highest score, regarding node scores of negative nodes and the positive meta-nodes they connect, gives the resulting approximated module.

To validate the performance of the heuristic we simulate artificial signal modules. For this we use the induced subnetwork of the HPRD-network comprising the genes present on the hgu133a Affymetrix chip. Within this network we set artificial signal modules of biological relevant sizes of 30 and 150 nodes, respectively; the remaining genes are considered as background noise. For all considered genes we simulate microarray data and analyze subsequently the simulated gene expression data analogously to the real expression analysis. We scan a large range of FDRs between 0 and 0.8 and evaluate the obtained solutions in terms of recall (true positive rate) and precision (ratio of true positives to all positively classified), for the optimal solution, our heuristic and a heuristic implemented in the Cytoscape plugin *jActiveModules* **?**]. The results of the heuristic implemented in the BioNet package are clearly closer to the optimal solutions, than the results of the other heuristical approach. Especially for a strong signal with 150 genes, our heuristic yields a good approximation of the maximum-scoring subnetwork.

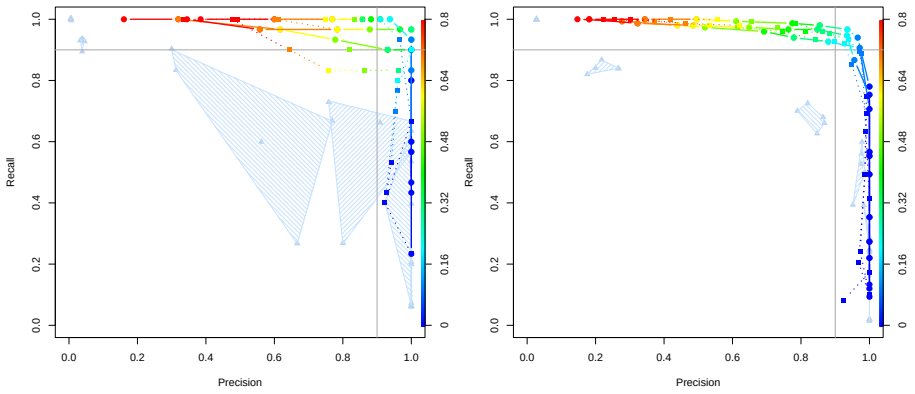

Figure 2: Performance validation. Plot of the recall vs. precision of a batch of solutions calculated for a wide range of FDRs (colouring scheme) with three replications each, for the exact solution and two heuristics. For the algorithm by **?**] we display the 6 convex hulls (triangles) of solutions (solutions 5 and 6 partially overlap) obtained by applying it recursively to five independent simulations. We evaluated two different signal component sizes (30, left plot and 150, right plot) with the same procedure. Clearly, the presented exact approach (solid line) captures the signal with high precision and recall over a relatively large range of FDRs. The results of the BioNet heuristic algorithm (dotted line) are much closer to the optimal solution over the entire range of FDRs compared to the *jActiveModules* heuristic; in particular, in the important region of high recall and precision.

3 Diffuse Large B-cell Lymphoma Study

Integrated network analysis not only focuses on the structure (topology) of the underlying graph but integrates external information in terms of node and edge attributes. Here we exemplify how an integrative network analysis can be performed using a protein-protein interaction network, microarray and clinical (survival) data, for details see **?**].

3.1 The data

First, we load the microarray data and interactome data which is available as expression set and a graph object from the BioNet package. The graph objects can be either in the *graphNEL* format, which is used in the package `graph` and `RBGL` [**?** **?** **?**] or in the *igraph* format from the package `igraph` [**?**].

```
> library(BioNet)
> library(DLBCL)
> data(exprLym)
> data(interactome)
```

Here we use the published gene expression data set from diffuse large B-cell lymphomas (DLBCL) [**?**]. In particular, gene expression data from 112 tumors

with the germinal center B-like phenotype (GCB DLBCL) and from 82 tumors with the activated B-like phenotype (ABC DLBCL) are included in this study. The expression data has been precompiled in an `ExpressionSet` structure.

```
> exprLym
```

```
ExpressionSet (storageMode: lockedEnvironment)
assayData: 3583 features, 194 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: Lym432 Lym431 ... Lym274 (194 total)
  varLabels: Subgroup IPI ... Status (5 total)
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
Annotation:
```

For the network data we use a data set of literature-curated human protein-protein interactions obtained from HPRD [**?**]. Altogether the entire network used here comprises 9386 nodes and 36504 edges.

```
> interactome
```

```
A graphNEL graph with undirected edges
Number of Nodes = 9386
Number of Edges = 36504
```

From this we derive a *Lymphochip*-specific interactome network as the vertex-induced subgraph extracted by the subset of genes for which we have expression data on the *Lymphochip*. This can easily be done, using the `subNetwork` command.

```
> network <- subNetwork(featureNames(exprLym), interactome)
> network
```

```
A graphNEL graph with undirected edges
Number of Nodes = 2559
Number of Edges = 8538
```

Since we want to identify modules as connected subgraphs we focus on the largest connected component.

```
> network <- largestComp(network)
> network
```

```
A graphNEL graph with undirected edges
Number of Nodes = 2034
Number of Edges = 8399
```

So finally we derive a *Lymphochip* network which comprises 2034 nodes and 8399 edges.

3.2 Calculating the p-values

Differential expression In the next step we use `rowttest` from the package `genefilter` to analyse differential expression between the ABC and GCB subtype:

```
> library(genefilter)
> library(impute)
> expressions <- impute.knn(exprs(exprLym))$data

Cluster size 3583 broken into 2332 1251
Cluster size 2332 broken into 1628 704
Cluster size 1628 broken into 1 1627
Done cluster 1
Cluster size 1627 broken into 727 900
Done cluster 727
Done cluster 900
Done cluster 1627
Done cluster 1628
Done cluster 704
Done cluster 2332
Done cluster 1251

> t.test <- rowttests(expressions, fac = exprLym$Subgroup)

> t.test[1:10, ]
```

The result looks as follows:

	statistic	dm	p.value
MYC(4609)	-3.38	-0.41	0.00
KIT(3815)	1.37	0.12	0.17
ETS2(2114)	0.51	0.04	0.61
TGFBR3(7049)	-2.89	-0.43	0.00
CSK(1445)	-3.61	-0.26	0.00
ISGF3G(10379)	-2.94	-0.25	0.00
RELA(5970)	-0.95	-0.07	0.34
TIAL1(7073)	-2.03	-0.09	0.04
CCL2(6347)	-0.94	-0.16	0.35
SELL(6402)	-2.18	-0.26	0.03

Survival analysis The survival analysis implemented in the package *survival* can be used to assess the risk association of each gene and calculate the associated p-values. As this will take some time, we here use the precalculated p-values from the BioNet package.

```
> data(dataLym)
> ttest.pval <- t.test[, "p.value"]
> surv.pval <- dataLym$s.pval
> names(surv.pval) <- dataLym$label
> pvals <- cbind(ttest.pval, surv.pval)
```

3.3 Calculation of the score

We have obtained two p-values for each gene, for differential expression and survival relevance. Next we aggregate these two p-values for each gene into one p-value of p-values using order statistics. The function `aggrPvals` calculates the second order statistic of the p-values for each gene.

```
> pval <- aggrPvals(pvals, order = 2, plot = FALSE)
```

Now we can use the aggregated p-values to fit the Beta-uniform mixture model to the distribution. The following plot shows the fitted Beta-uniform mixture model in a histogram of the p-values.

```
> fb <- fitBumModel(pval, plot = FALSE)
> fb

Beta-Uniform-Mixture (BUM) model

3583 pvalues fitted

Mixture parameter (lambda):        0.537
shape parameter (a):                 0.276
log-likelihood:                        1651.0
```

```
> dev.new(width = 13, height = 7)
> par(mfrow = c(1, 2))
> hist(fb)
> plot(fb)
> dev.off()

null device
          1
```

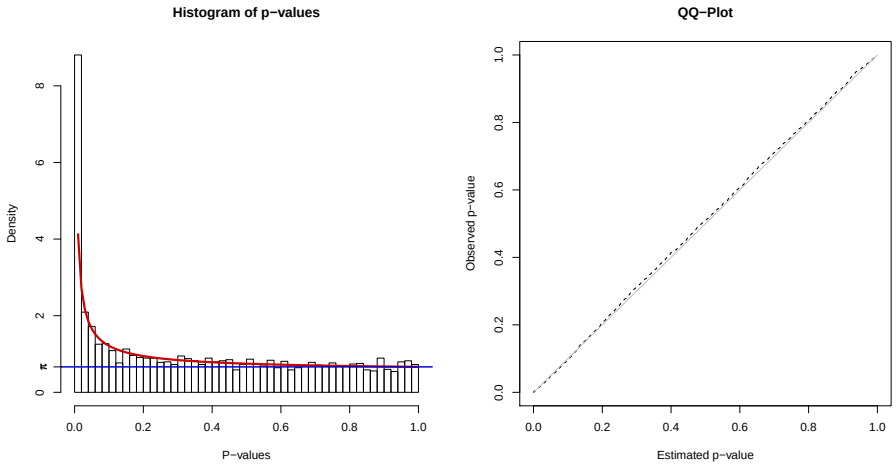

Figure 3: Histogram of p-values, overlayed by the fitted BUM model coloured in red and the π-upper bound displayed as a blue line. The right plot shows a quantile-quantile plot, indicating a nice fit of the BUM model to the p-value distribution.

The quantile-quantile plot indicates that the BUM model fits nicely to the p-value distribution. A plot of the log-likelihood surface can be obtained with `plotLLSurface`. It shows the mixture parameter λ (x-axis) and the shape parameter a (y-axis) of the Beta-uniform mixture model. The circle in the plot depicts the maximum-likelihood estimates for λ and a.

```
> plotLLSurface(pval, fb)
```

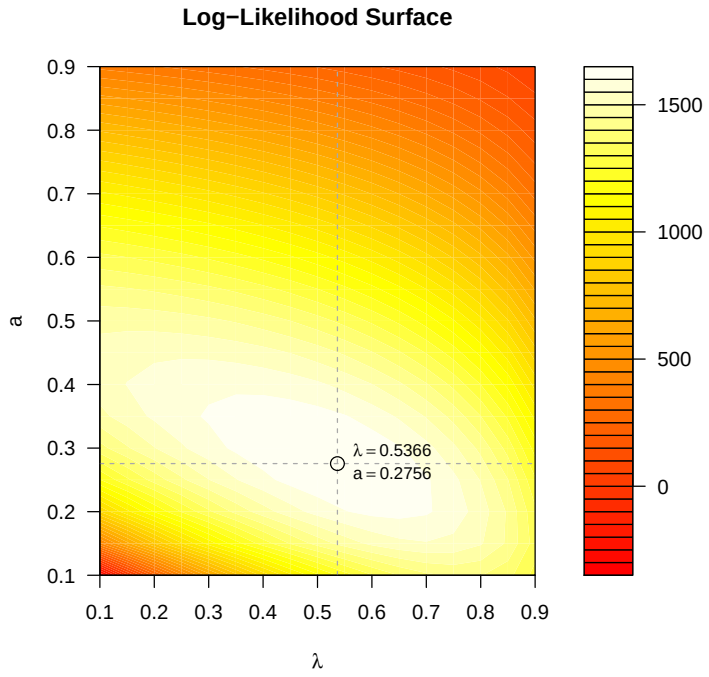

Figure 4: Log-likelihood surface plot. The range of the colours shows an increased log-likelihood from red to white. Additionaly, the optimal parameters λ and a for the BUM model are highlighted.

The nodes of the network are now scored using the fitted BUM model and a *FDR* of 0.001.

```
> scores <- scoreNodes(network = network, fb = fb,
+     fdr = 0.001)
```

In the next step the network with the scores and edges is written to a file and the `heinz` algorithm is used to calculate the maximum-scoring subnetwork. In order to run `heinz` self-loops have to be removed from the network.

```
> network <- rmSelfLoops(network)
> writeHeinzEdges(network = network, file = "lymphoma_edges_001",
+     use.score = FALSE)
```

```
[1] TRUE
```

```
> writeHeinzNodes(network = network, file = "lymphoma_nodes_001",
+     node.scores = scores)
```

```
[1] TRUE
```

3.4 Calculation of the maximum-scoring subnetwork

In the following the `heinz` algorithm is started using the `heinz.py` python script. This starts the integer linear programming optimization and calculates the maximum-scoring subnetwork using `CPLEX`.

The command is: "heinz.py -e lymphoma_edges_001.txt -n lymphoma_nodes_001.txt -N True -E False" or `runHeinz` on a linux machine with CPLEX installed.

The output is precalculated in `lymphoma_nodes_001.txt.0.hnz` and `lymphoma_edges_001.txt.0.hnz` in the subdirectory "extdata" of the R BioNet library directory.

```
> datadir <- file.path(path.package("BioNet"), "extdata")
> dir(datadir)
```

```
[1] "ALL_cons_n.txt.0.hnz"
[2] "ALL_edges_001.txt.0.hnz"
[3] "ALL_n_resample.txt.0.hnz"
[4] "ALL_nodes_001.txt.0.hnz"
[5] "cytoscape.sif"
[6] "lymphoma_edges_001.txt.0.hnz"
[7] "lymphoma_nodes_001.txt.0.hnz"
[8] "n.weight.NA"
[9] "weight.EA"
```

The output is loaded as a graph and plotted with the following commands:

```
> module <- readHeinzGraph(node.file = file.path(datadir,
+     "lymphoma_nodes_001.txt.0.hnz"), network = network)
> diff <- t.test[, "dm"]
> names(diff) <- rownames(t.test)
```

```
> plotModule(module, diff.expr = diff, scores = scores)
```

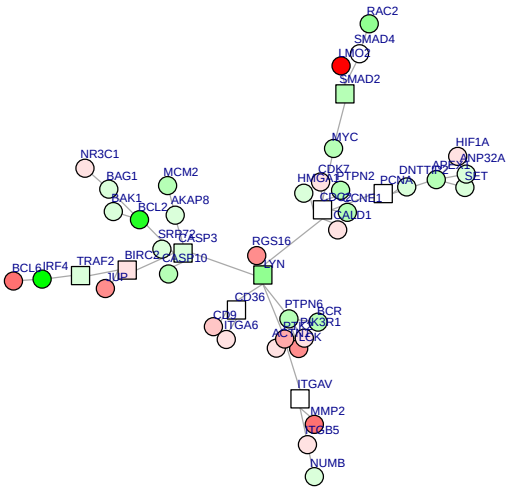

Figure 5: Resultant functional module for the lymphoma data set. Differential expression between ABC and GCB B-cell lymphoma is coloured in red and green, where green shows an upregulation in ACB and red an upregulation in GBC. The shape of the nodes depicts the score: rectangles indicate a negative score, circles a positive score.

The log fold-changes are visualized by the colouring of the nodes, the shape of the nodes depicts the score (positive=circle, negative=square). It is also possible to visualize the module in 3D with the function `plot3dModule`, but for this the rgl package, a 3D real-time rendering system, has to be installed. The plot can be saved to pdf-file with the function `save3dModule`. And the resulting module would look as following.

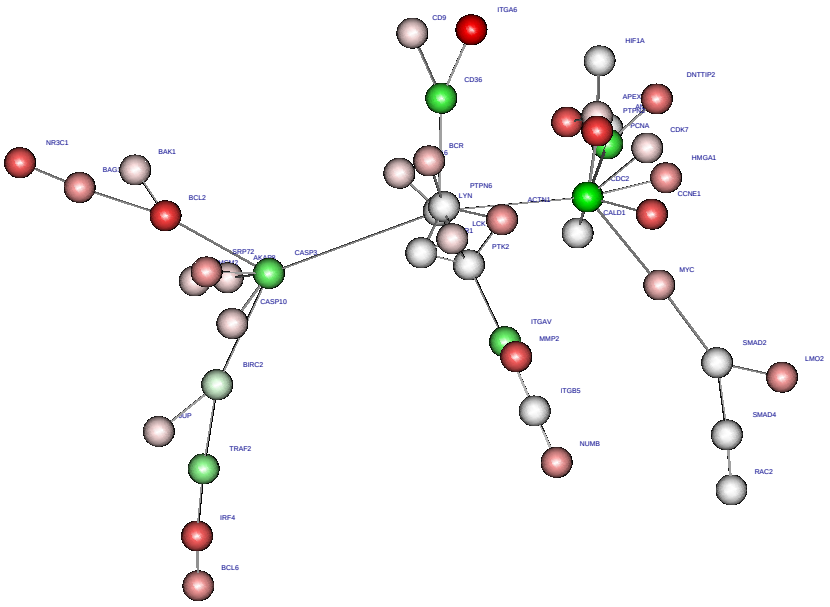

Figure 6: 3D visualization of the same functional module shown in **??**. Here, only the scores are depicted by the colouring of the nodes, positives in red and negatives in green.

The resulting subnetwork consists of 46 nodes and 50 edges. It has a cumulative sum of the scores of 71.07 from 37 positive (coloured in red) and 9 negative nodes (coloured in green).

```
> sum(scores[nodes(module)])
```

```
[1] 71.07341
```

```
> sum(scores[nodes(module)] > 0)
```

```
[1] 37
```

```
> sum(scores[nodes(module)] < 0)
```

```
[1] 9
```

We capture an interactome module that has been described to play a major biological role in the GCB and ABC DLBCL subtypes. It includes for example, the proliferation module which is more highly expressed in the ABC DLBCL subtype [**?**] comprising the genes: MYC, CCNE1, CDC2, APEX1, DNTTIP2, and PCNA. Likewise, genes IRF4, TRAF2, and BCL2, which are associated with the potent and oncogenic NFκB pathway.

4 ALL Study

This section describes the integrated network approach applied to the analysis of Affymetrix microrray data. In addition to the previous section, the data is

analysed for differential expression using the package `limma` [**?**]. The resulting module can be exported in various formats and for example displayed with Cytoscape [**?**].

4.1 The data

First, we load the microarray data and the human interactome data, which is available as a graph object from the BioNet package. The popular Acute Lymphoblastic Leukemia (ALL) data set [**?**] with 128 arrays is used as an example for an Affymetrix single-channel microarray. This data is available in the package `ALL` [**?**] as a normalised `ExpressionSet`.

```
> library(BioNet)
> library(DLBCL)
> library(ALL)
> data(ALL)
> data(interactome)
```

The mircroarray data gives results from 128 samples of patients with T-cell ALL or B-cell ALL using Affymetrix hgu95av2 arrays. Aim of the integrated analysis is to capture significant genes in a functional module, all of which are potentially involved in acute lymphoblastic leukemia and show a significant difference in expression between the B- and T-cell samples.

```
> ALL
```

```
ExpressionSet (storageMode: lockedEnvironment)
assayData: 12625 features, 128 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: 01005 01010 ... LAL4 (128 total)
  varLabels: cod diagnosis ... date last seen (21
    total)
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
  pubMedIds: 14684422 16243790
Annotation: hgu95av2
```

For the network data we use a data set of literature-curated human protein-protein interactions that have been obtained from HPRD [**?**]. Altogether the entire network used here comprises 9386 nodes and 36504 edges.

```
> interactome
```

```
A graphNEL graph with undirected edges
Number of Nodes = 9386
Number of Edges = 36504
```

In the next step we have to map the Affymetrix identifiers to the protein identifiers of the PPI network. Since several probesets represent one gene, we have

to select one or concatenate them into one gene. One possibility is to use the probeset with the highest variance for each gene. This is accomplished for the `ExpressionSet` with the `mapByVar`. It also maps the Affymetrix IDs to the identifiers of the network using the chip annotations and network geneIDs, which are unique, and returns the network names in the expression matrix. This reduces the expression matrix to the genes which are present in the network. Please note that the number of nodes and edges in the network and resulting module can slightly vary depending on the version of chip annotation used.

```
> mapped.eset <- mapByVar(ALL, network = interactome,
+     attr = "geneID")
> mapped.eset[1:5, 1:5]

                 01005     01010    03002    04006    04007
MAPK3(5595)   7.597323  7.479445 7.567593 7.384684 7.905312
TIE1(7075)    5.046194  4.932537 4.799294 4.922627 4.844565
CYP2C19(1557) 3.900466  4.208155 3.886169 4.206798 3.416923
BLR1(643)     5.903856  6.169024 5.860459 6.116890 5.687997
DUSP1(1843)   8.570990 10.428299 9.616713 9.937155 9.983809
```

The data set is reduced to 6134 genes. To find out how many genes are contained in the human interactome we calculate the intersect.

```
> length(intersect(rownames(mapped.eset), nodes(interactome)))

[1] 6134
```

Since the human interactome contains 6134 genes from the chip we can either extract a subnetwork with the method `subNetwork` or preceed with the whole network. Automatically the negative expectation value is used later when deriving the scores for the nodes without intensity values. We continue by extracting the subnetwork. Furthermore, we want to identify modules as connected subgraphs, therefore we use the largest connected component of the network and remove existing self-loops.

```
> network <- subNetwork(rownames(mapped.eset), interactome)
> network

A graphNEL graph with undirected edges
Number of Nodes = 6134
Number of Edges = 24400

> network <- largestComp(network)
> network <- rmSelfLoops(network)
> network

A graphNEL graph with undirected edges
Number of Nodes = 5609
Number of Edges = 22772
```

So finally we derive a *chip-specific* network which comprises 5609 nodes and 22772 edges.

4.2 Calculating the p-values

Differential expression In the next step we use `limma` [**?**] to analyse differential expression between the B-cell and T-cell groups.

```
> library(limma)
> design <- model.matrix(~-1 + factor(c(substr(unlist(ALL$BT),
+     0, 1))))
> colnames(design) <- c("B", "T")
> contrast.matrix <- makeContrasts(B - T, levels = design)
> contrast.matrix

      Contrasts
Levels B - T
     B     1
     T    -1

> fit <- lmFit(mapped.eset, design)
> fit2 <- contrasts.fit(fit, contrast.matrix)
> fit2 <- eBayes(fit2)
```

We get the corresponding p-values and and calculate the scores thereupon.

```
> pval <- fit2$p.value[, 1]
```

4.3 Calculation of the score

We have obtained the p-values for each gene for differential expression. Next, the p-values are used to fit the Beta-uniform mixture model to their distribution [**? ?**]. The following plot shows the fitted Beta-uniform mixture model in a histogram of the p-values. The quantile-quantile plot indicates that the BUM model fits to the p-value distribution. Although the data shows a slight deviation from the expected values, we continue with the fitted parameters.

```
> fb <- fitBumModel(pval, plot = FALSE)
> fb

Beta-Uniform-Mixture (BUM) model

6134 pvalues fitted

Mixture parameter (lambda):        0.456
shape parameter (a):                 0.143
log-likelihood:                        11978.8

> dev.new(width = 13, height = 7)
> par(mfrow = c(1, 2))
> hist(fb)
> plot(fb)
```

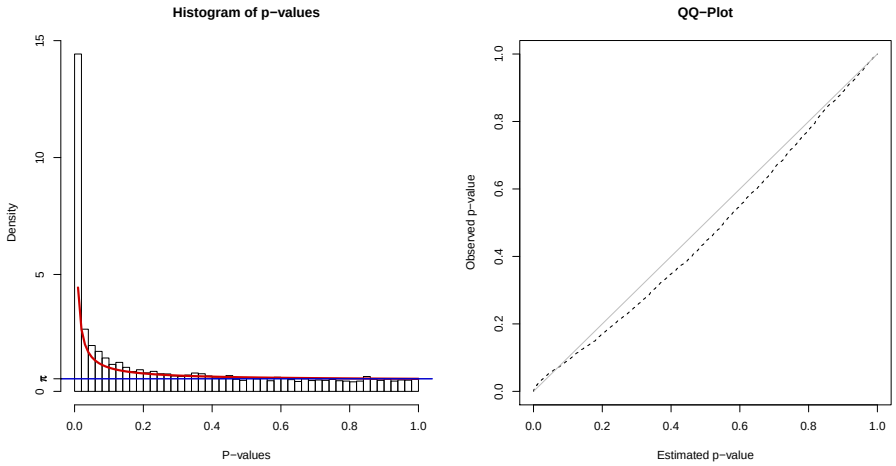

Figure 7: Histogram of p-values, overlayed by the fitted BUM model in red and the π-upper bound displayed as a blue line. The right plot shows a quantile-quantile plot, in which the estimated p-values from the model fit deviate slightly from the observed p-values.

The nodes of the network are now scored using the fitted BUM model and a *FDR* of 1e-14. Such a low *FDR* was chosen to obtain a small module, which can be visualized.

```
> scores <- scoreNodes(network = network, fb = fb,
+     fdr = 1e-14)
```

In the next step the network with the scores and edges is written to file and the `heinz` algorithm is used to calculate the maximum-scoring subnetwork.

```
> writeHeinzEdges(network = network, file = "ALL_edges_001",
+     use.score = FALSE)
```

```
[1] TRUE
```

```
> writeHeinzNodes(network = network, file = "ALL_nodes_001",
+     node.scores = scores)
```

```
[1] TRUE
```

4.4 Calculation of the maximum-scoring subnetwork

In the following the `heinz` algorithm is started using the `heinz.py` python script. A new implementation Heinz v2.0 is also available at `https://software.cwi.nl/software/heinz` , with slightly different and additional options. This starts the integer linear programming optimization and calculates the maximum-scoring subnetwork using `CPLEX`.

The command is: "heinz.py -e ALL_edges_001.txt -n ALL_nodes_001.txt -N True -E False" or `runHeinz` on a linux machine with CPLEX installed.

The output is precalculated in `ALL_nodes_001.txt.0.hnz` and `ALL_edges_001.txt.0.hnz` in the R BioNet directory, subdirectory extdata. The output is loaded as a graph with the following commands:

```
> datadir <- file.path(path.package("BioNet"), "extdata")
> module <- readHeinzGraph(node.file = file.path(datadir,
+     "ALL_nodes_001.txt.0.hnz"), network = network)
```

Attributes are added to the module, to depict the difference in expression and the score later.

```
> nodeDataDefaults(module, attr = "diff") <- ""
> nodeData(module, n = nodes(module), attr = "diff") <- fit2$coefficients[nodes(module),
+     1]
> nodeDataDefaults(module, attr = "score") <- ""
> nodeData(module, n = nodes(module), attr = "score") <- scores[nodes(module)]
> nodeData(module)[1]

$`BTK(695)`
$`BTK(695)`$geneID
[1] "695"

$`BTK(695)`$geneSymbol
[1] "BTK"

$`BTK(695)`$diff
[1] 0.919985

$`BTK(695)`$score
[1] -8.866629
```

We save the module as XGMML file and look at it with the software Cytoscape [**?**], colouring the node by their "diff" attribute and changing the node shape according to the "score".

```
> saveNetwork(module, file = "ALL_module", type = "XGMML")

[1] "...adding nodes"
[1] "...adding edges"
[1] "...writing to file"
```

The resulting network with 31 nodes and 32 edges and coloured nodes, looks like this:

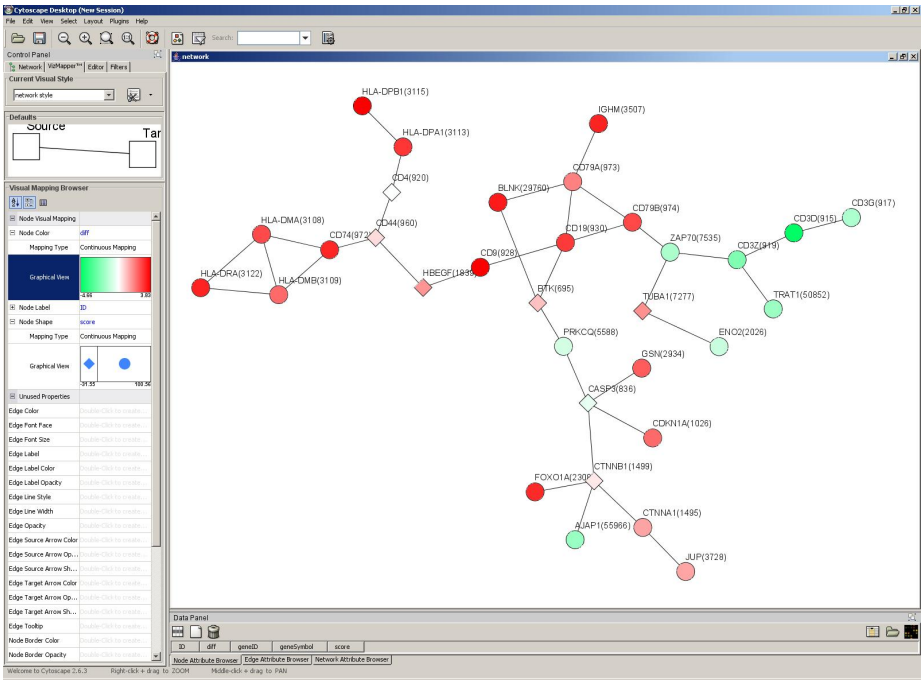

Figure 8: Resultant module visualized in Cytoscape. Significantly upregulated genes are coloured in red, genes that show significant downregulation are coloured in green, for the contrast B vs. T cells. The score of the nodes is shown by the shape of the nodes, circles indicate a positive score, diamonds a negative score.

The module comprises several parts, one part showing a high upregulation in the B-T contrast (CD79A, BLNK, CD19, CD9, CD79B) participates in B cell activation and differentiation and response to stimuli according to their GO annotation. While the other large upregulated part is involved in antigen processing and presentation and immune response (HLA-DMA, HLA-DPA1, CD4, HLA-DMB, HLA-DRB5, HLA-DPB1). T cell/leukocyte activating genes (CD3D, CD3G, CD3Z, ENO2, TRAT1, ZAP70) are coloured in green. The lower middle part is involved in negative regulation of apoptosis, developmental processes and programmed cell death. Most of them are involved in overall immune system processes and as expected, mostly B and T cell specific genes comprise the resulting module, as this contrast was used for the test of differential expression.

5 Consensus modules

To assess the variation inherent in the integrated data, we use a jackknife resampling procedure resulting in an ensemble of optimal modules. A consensus approach summarizes the ensemble into one final module containing maximally robust nodes and edges. The resulting consensus module visualizes variable regions by assigning support values to nodes and edges. The consensus module is calculated using the acute lymphoblastic leukemia data from the previous section.

First, we perform the same steps as explained in **??** and start with the data obtained up to subsection **??**. We use the *chip-specific* network which comprises 5609 nodes and 22772 edges and the ALL microarray dataset.

5.1 Calculating the p-values

Differential expression We resample the microarrays and calculate p-values using a standard two-sided t-test for the differential expression between the B-cell and T-cell groups. 100 jackknife replicates are created and used to test for differential expression.

Depending on the number of resamples the next steps can take a while.

```
> j.repl <- 100
> resampling.pvals <- list()
> for (i in 1:j.repl) {
+     resampling.result <- resamplingPvalues(exprMat = mapped.eset,
+         groups = factor(c(substr(unlist(ALL$BT),
+             0, 1))), resampleMat = FALSE, alternative = "two.sided")
+     resampling.pvals[[i]] <- resampling.result$p.values
+     print(i)
+ }
```

We use the obtained p-values to calculate scores thereupon.

5.2 Calculation of the score

For each jackknife replicate a BUM model is fitted to the p-value distribution, which is used to calculate node scores. The same *FDR* as before is used.

```
> fb <- lapply(resampling.pvals, fitBumModel, plot = FALSE,
+     starts = 1)
> resampling.scores <- c()
> for (i in 1:j.repl) {
+     resampling.scores[[i]] <- scoreNodes(network = network,
+         fb = fb[[i]], fdr = 1e-14)
+ }
```

We create a matrix of scores to calculate the modules. This creates one node file as input for the ILP calculation. Alternatively one node file can be created for each jackknife resample, which then can be run in parallel on a cluster or multicore machine. This approach is preferable, due to possibly many jackknife resamples. For simplification we use the matrix variant here.

```
> score.mat <- as.data.frame(resampling.scores)
> colnames(score.mat) <- paste("resample", (1:j.repl),
+     sep = "")
```

The node scores are written to file in the next step. For the edge scores the binary interactions are written to file with a score of 0.

```
> writeHeinzEdges(network = network, file = "ALL_e_resample",
+     use.score = FALSE)
> writeHeinzNodes(network = network, file = "ALL_n_resample",
+     node.scores = score.mat)
```

5.3 Calculation of the optimal subnetworks

In the following the `heinz` algorithm is started using the `heinz.py` python script. This starts the integer linear programming optimization and calculates the maximum-scoring subnetwork using `CPLEX`. In contrast to previous calculations we define a size the resulting modules should have. We set it with the size parameter s to the size of the original module, which was 31. This fixes the size of the output modules for a later consensus module calculation.

The command is: "heinz.py -e ALL_e_resample.txt -n ALL_n_resample.txt -N True -E False -S 31"

The output is precalculated in `ALL_n_resample.txt.0.hnz` and `ALL_e_resample.txt.0.hnz` in the R BioNet directory, subdirectory extdata. The output is loaded as a list of graphs with the following commands:

```
> datadir <- file.path(path.package("BioNet"), "extdata")
> modules <- readHeinzGraph(node.file = file.path(datadir,
+     "ALL_n_resample.txt.0.hnz"), network = network)
```

5.4 Calculation of the consensus module

We have obtained now 100 modules from the resampled data. These are used to calculate consensus scores for the nodes and edges of the network and recalculate an optimal module. This module, termed consensus module, captures the variance in the microarray data and depicts the robust solution. Confidence values for the nodes and edges can be visualized by the node size and edge width, allowing to identify stable parts of the module.

We therefore use the modules to calculate consensus scores in the following and rescore the network:

```
> cons.scores <- consensusScores(modules, network)
> writeHeinz(network = network, file = "ALL_cons",
+     node.scores = cons.scores$N.scores, edge.scores = cons.scores$E.scores)
```

```
[1] TRUE
```

They are run using CPLEX: "heinz.py -e ALL_cons_e.txt -n ALL_cons_n.txt -N True -E True -S 31". Mind to also use the edge scores with -E True.

The results are loaded in R and visualized.

```
> datadir <- file.path(path.package("BioNet"), "extdata")
> cons.module <- readHeinzGraph(node.file = file.path(datadir,
+     "ALL_cons_n.txt.0.hnz"), network = network)
```

```
> cons.edges <- sortedEdgeList(cons.module)
> E.width <- 1 + cons.scores$E.freq[cons.edges] *
+     10
> N.size <- 1 + cons.scores$N.freq[nodes(cons.module)] *
+     10

> plotModule(cons.module, edge.width = E.width,
+     vertex.size = N.size, edge.label = cons.scores$E.freq[cons.edges] *
+         100, edge.label.cex = 0.6)
```

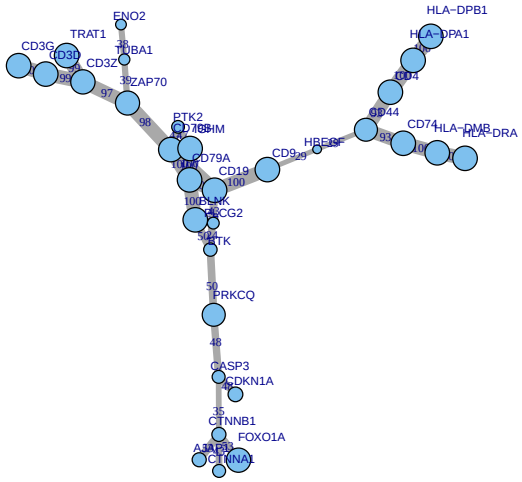

Figure 9: Resultant consensus module. The size of the nodes and the width of the edges depict the robustness of this node or edge as calculated from the jackknife replicates.

6 Installation

6.1 The BioNet package

The BioNet package is freely available from Bioconductor at http://www.bioconductor.org.

6.2 External code to call CPLEX

The algorithm to identify the optimal scoring subnetwork is based on the software dhea (district heating) from **?**]. The C++ code was extended in order to generate suboptimal solutions and is controlled over a Python script. The

dhea code uses the commercial CPLEX callable library version 9.030 by ILOG, Inc. (Sunnyvale,CA). In order to calculate the optimal solution a CPLEX library is needed. The other routines, the dhea code and heinz.py Python script (current version 1.63) are publicly available for academic and research purposes within the heinz (heaviest induced subgraph) package of the open source library LiSA (`http://www.planet-lisa.net`). The dhea code has to be included in the same folder as heinz.py, in order to call the routine by the Python code. To calculate the maximum-scoring subnetwork without an available CPLEX license a heuristic is included in the BioNet package, see `runFastHeinz`.
The new version of Heinz v2.0 is available at `https://software.cwi.nl/software/heinz`**.**