Ionflow: Ionomics data network and enrichment analysis

Wanchang Lin

01-12-2020

Contents

This vignette explains how to perform ionomics data analysis including gene network and enrichment analysis by using a modification of the R package, ionflow. The modification(`ionflow_funcs`) was made by Wanchang Lin (w.lin@imperial.ac.uk) and Jacopo Iacovacci(j.iacovacci@imperial.ac.uk).

Data preparation

To explore the process, we'll use the ionomics data set:

```
ion_data <- read.table("../test-data/iondata.tsv", header = T, sep = "\t")
dim(ion_data)
#> [1] 9999   16
```

Ten random data records are shown as:

```
sample_n(ion_data, 10)
```

Table 1: Samples of raw data

Knockout	Batch_ID	Ca	Cd	Co	Cu	Fe	K	Mg	Mn	Mo	Na	Ni	P	S	Zn
YOR132W	25	46.24	1.70	0.15	1.61	6.74	2766.94	464.37	1.75	0.98	195.52	0.79	3889.54	499.70	15.98
YEL013W	79	56.96	0.86	0.11	1.34	3.86	1243.08	508.83	0.72	1.36	302.75	0.84	3895.54	345.75	10.06
YGL234W	81	50.38	1.06	0.22	2.46	14.83	2240.52	451.96	0.94	2.45	6.71	0.54	3975.00	645.71	28.13
YDR128W	10	12.61	0.69	0.19	1.68	6.38	2938.43	369.27	0.92	0.55	81.09	1.96	2784.41	283.18	14.17
YER079W	14	44.86	1.03	0.16	2.04	8.83	3546.19	724.70	1.61	1.68	217.21	1.52	4762.49	491.50	18.97
YLR203C	80	19.90	1.16	0.17	1.21	6.66	1142.74	831.45	0.70	0.58	212.47	1.19	5313.99	601.53	14.72
YBR182C	4	28.46	0.82	0.12	1.15	3.20	1545.79	157.46	0.78	0.54	42.91	0.77	1194.95	149.42	11.54
YBR259W	7	39.06	0.91	0.17	1.85	6.09	2647.08	482.72	1.10	0.48	171.09	0.90	3117.78	473.90	18.50
YAL034C	15	39.31	0.99	0.16	1.72	3.71	3407.73	715.12	1.40	1.21	230.25	1.43	4578.58	505.24	18.16
YDR350C	11	40.32	0.67	0.18	1.46	9.09	911.09	478.56	0.75	0.32	144.15	1.40	2785.76	459.29	12.35

The first few columns are meta information such as gene ORF and batch id. The rest is the ionomics data.

Data pre-processing

The raw data set should be pre-processed. The pre-processing function `PreProcessing` has functions:

- log transformation
- batch correction
- outlier detection
- standardisation

The raw data are at first log transformed and then followed by the batch correction. User can chose not to perform batch correction, otherwise default will be either *median* or *median* plus *std* method. If there is quality control for the batch correction, the user can use it and indicates in the argument of `control_lines`. Also one argument gives the user the option on how to use these control lines (`control_use`): If `control_use` is `control`, these control lines (data rows) are used for the batch correction factor; if `control.out`, others lines are used.

This data set has a control line: **YDL227C** mutant. The code segment below is to identify it:

```
max(with(ion_data, table(Knockout)))
#> [1] 1617
which.max(with(ion_data, table(Knockout)))
#> YDL227C
#>     209
```

The next stage is outlier detection. Here only univariate methods are implemented, including *mad*, *IQR*, and *log.FC.dist*. And like batch correction, the user can skip this procedure by setting `method_outliers = none` in the function argument. There is a threshold to control the number of outliers. The larger the threshold (`thres_outl`) the more outlier removal.

Standardisation provides three methods: *std*, *mad* or *custom*. If the method is *custom*, the user uses a specific *std* file like:

```
std <- read.table("../test-data/user_std.tsv", header = T, sep = "\t")
std
#>    Ion     sd
#> 1   Ca 0.1508
#> 2   Cd 0.0573
#> 3   Co 0.0580
#> 4   Cu 0.0735
#> 5   Fe 0.1639
#> 6    K 0.0940
#> 7   Mg 0.0597
#> 8   Mn 0.0771
#> 9   Mo 0.1142
#> 10  Na 0.1075
#> 11  Ni 0.0784
#> 12   P 0.0597
#> 13   S 0.0801
#> 14  Zn 0.0671
```

The pre-processing procedure returns not only processed ionomics data but also a symbolic data set. This data set is based on the ionomics data and is determined by a threshold(`thres_symb`):

- `0` if ionomics value is located in `[-thres_symb, thres_symb]`
- `1` if ionomics value is larger than `thres_symb`
- `-1` if ionomics value is smaller than `-thres_symb`

Note that the symbolic data set is important since the key part of the network and enrichment analysis is based on the hierarchical clustering of symbolic data.

Let's run the pre-process procedure:

```
pre <- PreProcessing(data = ion_data,
                     var_id = 1, batch_id = 2, data_id = 3,
                     method_norm = "median",
                     control_lines = "YDL227C",
                     control_use = "control",
                     method_outliers = "IQR",
                     thres_outl = 3,
                     stand_method = "std",
                     stdev = NULL,
                     thres_symb = 3)

names(pre)
#> [1] "stats.raw_data"    "stats.outliers"    "stats.batch_data"
#> [4] "data.long"         "data.gene.logFC"   "data.gene.zscores"
#> [7] "data.gene.symb"    "plot.dot"          "plot.hist"
```

The results include summaries of raw data and processed data. The latter is:

```
pre$stats.batch_data %>%
  kable(caption = 'Processed data summary', digits = 2, booktabs = T) %>%
  kable_styling(full_width = F, font_size = 10)
```

Table 2: Processed data summary

Ion	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	Variance
Ca	-4.45	-0.28	-0.13	-0.12	0.02	2.35	0.11
Cd	-1.70	0.03	0.10	0.11	0.17	0.93	0.03
Co	-2.80	0.02	0.09	0.06	0.15	1.60	0.05
Cu	-0.66	-0.10	-0.03	-0.01	0.04	5.28	0.04
Fe	-7.48	-0.17	-0.06	-0.02	0.07	6.88	0.14
K	-2.21	-0.17	-0.01	-0.08	0.09	1.83	0.08
Mg	-1.84	-0.06	0.01	-0.01	0.07	1.69	0.03
Mn	-4.11	-0.24	-0.08	-0.13	0.01	1.78	0.06
Mo	-2.03	-0.26	-0.08	-0.08	0.09	4.44	0.13
Na	-7.41	-0.53	-0.22	-0.33	-0.04	1.25	0.24
Ni	-2.40	-0.01	0.09	0.12	0.21	7.90	0.12
P	-1.18	-0.06	0.00	-0.01	0.06	1.45	0.02
S	-2.38	-0.03	0.05	0.06	0.16	2.38	0.04
Zn	-0.46	-0.08	-0.03	-0.01	0.03	4.60	0.02

The pre-processed data and symbolic data are like this:

```
pre$data.gene.zscores %>% head() %>%
  kable(caption = 'Processed data', digits = 2, booktabs = T) %>%
  kable_styling(full_width = F, font_size = 10,
                latex_options = c("striped", "scale_down"))
```

Table 3: Processed data

Line	Ca	Cd	Co	Cu	Fe	K	Mg	Mn	Mo	Na	Ni	P	S	Zn
YAL004W	-1.16	0.75	1.19	-0.47	0.04	0.61	0.51	-0.84	-0.08	-1.84	1.71	0.52	0.33	-0.09
YAL005C	-1.67	0.84	0.55	0.58	-2.79	0.59	0.31	-1.16	-1.42	-0.12	1.48	0.73	0.13	-0.13
YAL007C	-2.12	0.64	0.23	-0.53	-0.24	0.79	-0.09	-0.14	1.22	-0.92	0.00	0.09	-0.29	-0.65
YAL008W	-2.34	1.13	0.21	-0.73	-2.16	0.52	-0.02	-0.87	0.93	-0.58	0.02	-0.09	-0.73	-0.47
YAL009W	-1.18	0.66	0.55	-1.11	-3.91	0.22	0.09	-0.18	1.50	-0.84	-0.09	0.14	0.01	-0.36
YAL010C	-1.28	1.43	2.27	0.46	1.53	-2.75	0.04	-0.74	-9.71	-4.30	2.42	-0.98	-0.05	-0.01

```
pre$data.gene.symb %>% head() %>%
  kable(caption = 'Symbolic data', booktabs = T) %>%
  kable_styling(full_width = F, font_size = 10)
```

Table 4: Symbolic data

Line	Ca	Cd	Co	Cu	Fe	K	Mg	Mn	Mo	Na	Ni	P	S	Zn
YAL004W	0	0	0	0	0	0	0	0	0	0	0	0	0	0
YAL005C	0	0	0	0	0	0	0	0	0	0	0	0	0	0
YAL007C	0	0	0	0	0	0	0	0	0	0	0	0	0	0
YAL008W	0	0	0	0	0	0	0	0	0	0	0	0	0	0
YAL009W	0	0	0	0	-1	0	0	0	0	0	0	0	0	0
YAL010C	0	0	0	0	0	0	0	0	-1	-1	0	0	0	0

The symbolic data are calculated from the processed data with control of `thres_symb` (here it is 3). You can obtain a new symbol data set by re-assigning a new threshold to the function `symbol_data`:

```
data_symb <- symbol_data(pre$data.gene.zscores, thres_symb = 2)
data_symb %>% head() %>%
  kable(caption = 'Symbolic data with threshold of 2', booktabs = T) %>%
  kable_styling(full_width = F, font_size = 10)
```

The `thres_symb` is a crucial value to get the symbolic data. Before re-setting this threshold, the user should check the summary of processed data and pay attention to the maximum values. For example, some ions (for example, *Cd* and *Mn*) are all zero even with 2 of `thres_symb`.

The pre-processed data distribution is:

Table 5: Symbolic data with threshold of 2

Line	Ca	Cd	Co	Cu	Fe	K	Mg	Mn	Mo	Na	Ni	P	S	Zn
YAL004W	0	0	0	0	0	0	0	0	0	0	0	0	0	0
YAL005C	0	0	0	0	-1	0	0	0	0	0	0	0	0	0
YAL007C	-1	0	0	0	0	0	0	0	0	0	0	0	0	0
YAL008W	-1	0	0	0	-1	0	0	0	0	0	0	0	0	0
YAL009W	0	0	0	0	-1	0	0	0	0	0	0	0	0	0
YAL010C	0	0	1	0	0	-1	0	0	-1	-1	1	0	0	0

```
pre$plot.hist
```

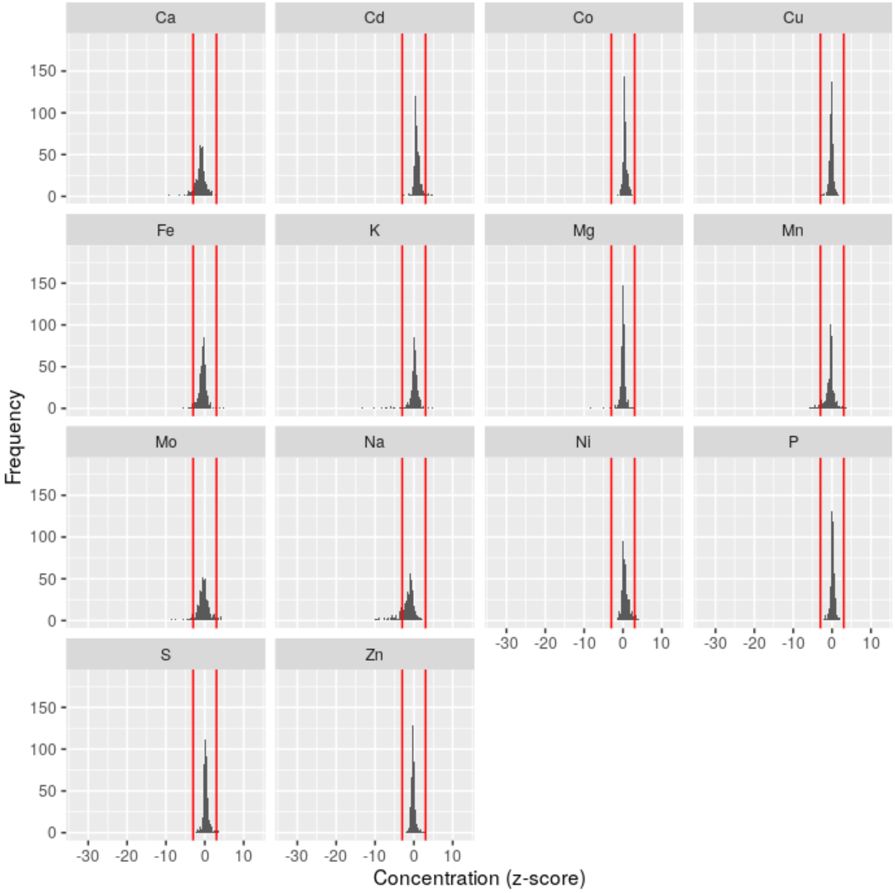

Figure 1: Ionomcs data distribution plot

Data filtering

There are a lot of ways to filter genes. Here genes are filtered according to symbolic data: remove genes with all values which are zero.

```
data <- pre$data.gene.zscores
data_symb <- pre$data.gene.symb
idx <- rowSums(abs(data_symb[, -1])) > 0
dat <- data[idx, ]
dat_symb <- data_symb[idx, ]
dim(dat)
#> [1] 549  15
```

Data clustering

The hierarchical cluster analysis is the key part of gene network and gene enrichment analysis. The methodology is as follow:

- Compute the distance of symbolic data
- Hierarchical cluster analysis on the distance
- Identify clusters/groups with a threshold of minimal number of cluster size

One example is:

```
clust <- gene_clus(dat_symb[, -1], min_clust_size = 10)
names(clust)
#> [1] "clus"    "idx"     "tab"     "tab_sub"
```

The cluster centres are:

```
clust$tab_sub
#>   cluster nGenes
#> 1      11     72
#> 2       7     36
#> 3       1     27
#> 4      18     15
#> 5       5     12
#> 6       3     11
#> 7       8     11
```

This shows clusters and the number of genes (larger than `min_cluster_size`).

The identified gene located in those clusters are:

```
sum(clust$idx)                     #' numbers of all genes
#> [1] 184
as.character(dat[,1][clust$idx])  #' and they are
```

```
#>   [1] "YAL009W"   "YAL013W"   "YAL014C"   "YAL022C"   "YAL027W"   "YAL042W"  
#>   [7] "YAL046C"   "YAL051W"   "YAL066W"   "YAR003W"   "YAR031W"   "YAR035W"  
#>  [13] "YBR185C"   "YBR194W"   "YBR195C"   "YBR199W"   "YBR204C"   "YBR216C"  
#>  [19] "YBR217W"   "YBR218C"   "YBR219C"   "YBR221C"   "YBR233W"   "YBR240C"  
#>  [25] "YBR245C"   "YBR249C"   "YBR260C"   "YCL032W"   "YCL033C"   "YCL050C"  
#>  [31] "YCL060C"   "YCR005C"   "YDR063W"   "YDR095C"   "YDR096W"   "YDR109C"  
#>  [37] "YDR127W"   "YDR133C"   "YDR135C"   "YDR137W"   "YDR143C"   "YDR146C"  
#>  [43] "YDR147W"   "YDR158W"   "YDR181C"   "YDR183W"   "YDR186C"   "YDR198C"  
#>  [49] "YDR209C"   "YDR220C"   "YDR221W"   "YDR222W"   "YDR223W"   "YDR227W"  
#>  [55] "YDR338C"   "YDR344C"   "YDR370C"   "YDR374C"   "YDR379W"   "YDR415C"  
#>  [61] "YDR430C"   "YEL003W"   "YEL005C"   "YEL008W"   "YEL046C"   "YER004W"  
#>  [67] "YER007C-A" "YER007W"   "YER032W"   "YER034W"   "YER053C"   "YER054C"  
#>  [73] "YER056C-A" "YER067W"   "YER068C-A" "YER074W"   "YER086W"   "YGL205W"  
#>  [79] "YGL213C"   "YGL217C"   "YGL241W"   "YGL257C"   "YGR007W"   "YGR008C"  
#>  [85] "YGR034W"   "YGR043C"   "YGR045C"   "YGR070W"   "YGR071C"   "YGR077C"  
#>  [91] "YGR079W"   "YGR087C"   "YGR122W"   "YGR144W"   "YGR169C"   "YGR182C"  
#>  [97] "YGR187C"   "YGR193C"   "YGR194C"   "YGR203W"   "YGR205W"   "YGR207C"  
#> [103] "YHL006C"   "YHL019C"   "YHL021C"   "YHL028W"   "YHL032C"   "YHL037C"  
#> [109] "YHL041W"   "YHL046C"   "YHR012W"   "YHR046C"   "YHR057C"   "YHR073W"  
#> [115] "YHR075C"   "YHR123W"   "YHR143W"   "YHR150W"   "YHR152W"   "YHR155W"  
#> [121] "YJL148W"   "YJL162C"   "YJL198W"   "YJL199C"   "YJL206C-A" "YKL023W"  
#> [127] "YKL027W"   "YKL114C"   "YKL127W"   "YKL129C"   "YKL132C"   "YKL136W"  
#> [133] "YKL147C"   "YKL148C"   "YKL150W"   "YKL151C"   "YKL158W"   "YKL161C"  
#> [139] "YKL163W"   "YKL164C"   "YKL166C"   "YKL171W"   "YKL174C"   "YLL019C"  
#> [145] "YLL020C"   "YLL021W"   "YLL029W"   "YLL038C"   "YLR130C"   "YLR171W"  
#> [151] "YLR172C"   "YLR177W"   "YLR182W"   "YLR213C"   "YLR214W"   "YLR220W"  
#> [157] "YLR266C"   "YLR278C"   "YLR313C"   "YLR348C"   "YLR351C"   "YLR354C"  
#> [163] "YLR363C"   "YLR376C"   "YLR380W"   "YLR381W"   "YLR398C"   "YLR451W"  
#> [169] "YOL153C"   "YOR097C"   "YOR126C"   "YOR131C"   "YOR142W"   "YOR163W"  
#> [175] "YOR188W"   "YOR208W"   "YOR213C"   "YOR216C"   "YOR223W"   "YOR228C"  
#> [181] "YOR229W"   "YOR245C"   "YOR247W"   "YOR255W"
```

Gene network

The gene network uses both the ionomics and symbolic data. The similarity measures on ionomics data are used to construct the network. Before creating a network, these analyses are further filtered by:

- clustering of symbolic data;
- and the similarity threshold located between 0 and 1;

The methods implemented are: *pearson*, *spearman*, *kendall*, *cosine*, *mahal_cosine* or *hybrid_mahal_cosine*. The first three methods are correlation methods and *cosine* is similar to the Pearson correlation which is the cosine similarity between two centred vectors. For the last two methods, see publication: Extraction and Integration of Genetic Networks from Short-Profile Omic Data Sets for details.

For example, we use the Pearson correlation as similarity measure for network analysis:

```r
net <- GeneNetwork(data = dat,
                   data_symb = dat_symb,
                   min_clust_size = 10,
                   thres_corr = 0.75,
                   method_corr = "pearson")
```

The network with nodes coloured by the symbolic data clustering is:

```r
net$plot.pnet1
```

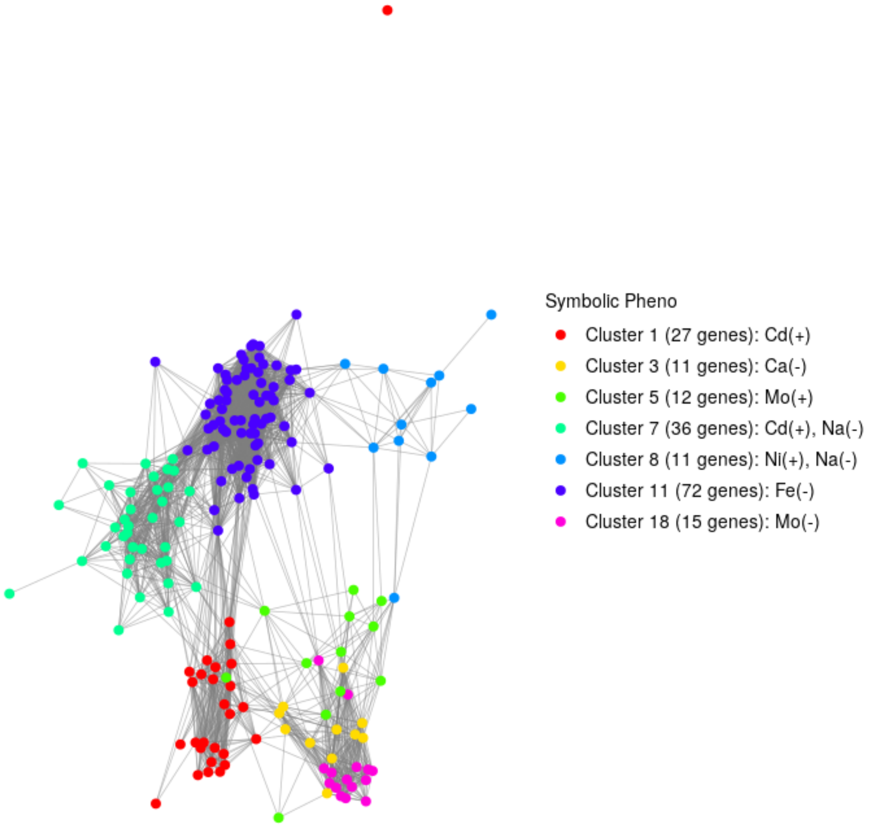

Figure 2: **Network with Pearson correlation: symbolic clustering**

The same network, but nodes are coloured by the network community detection:

```
net$plot.pnet2
```

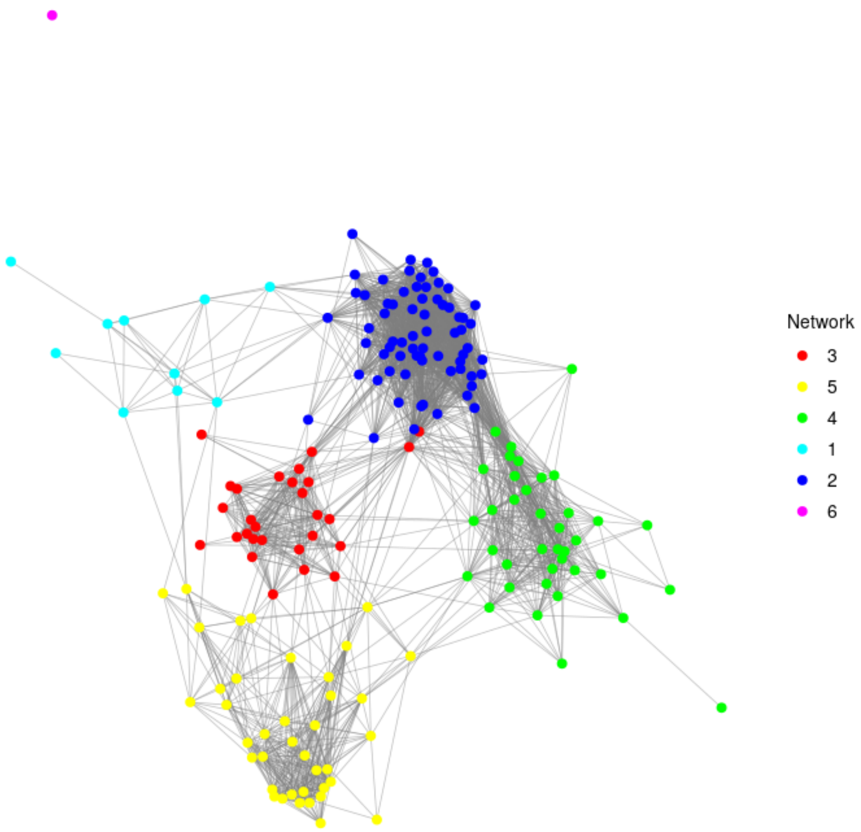

Figure 3: Network with Pearson correlation: community detction

The network analysis also returns a network impact and betweenness plot:

```
net$plot.impact_betweenness
```

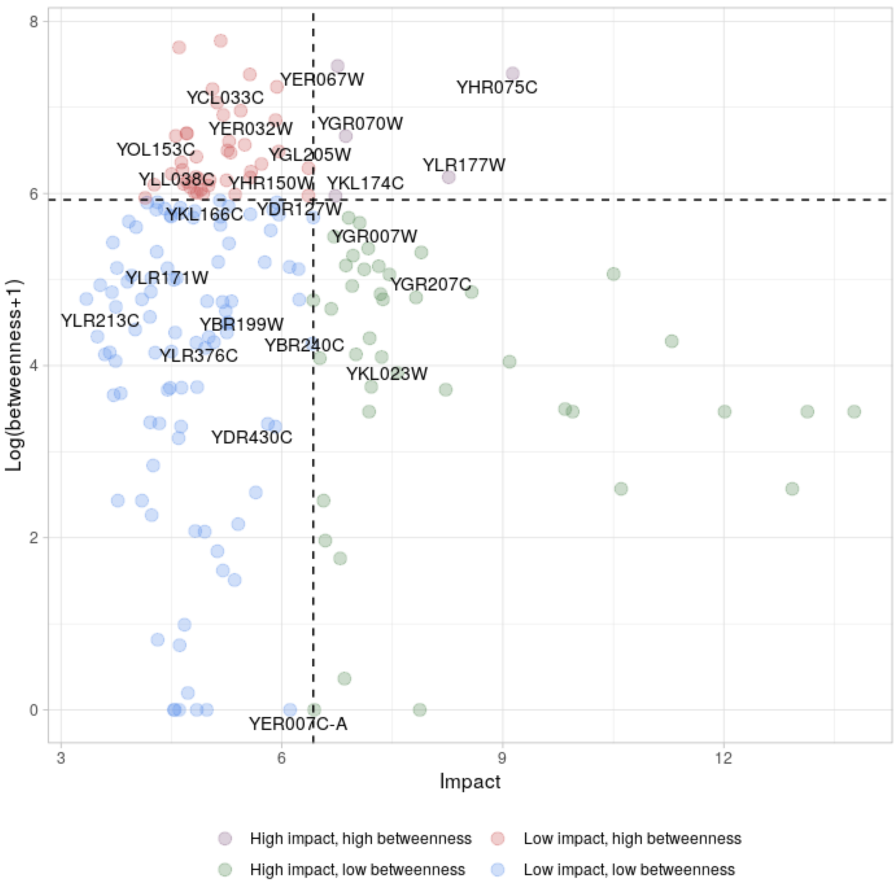

Figure 4: Network with Pearson correlation: impact and betweeness

For comparison purposes, we use different similarity methods. Here we choose *Cosine*:

```
net_1 <- GeneNetwork(data = dat,
                     data_symb = dat_symb,
                     min_clust_size = 10,
                     thres_corr = 0.75,
                     method_corr = "cosine")
net_1$plot.pnet1
```

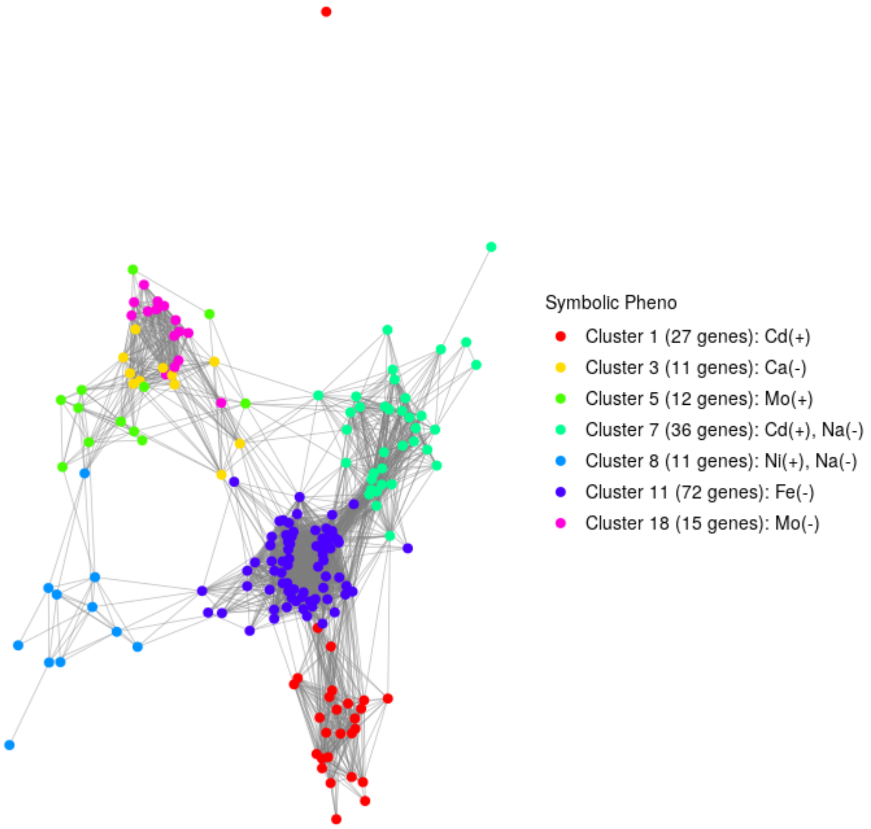
Symbolic Pheno
Cluster 1 (27 genes): Cd(+)
Cluster 3 (11 genes): Ca(-)
Cluster 5 (12 genes): Mo(+)
Cluster 7 (36 genes): Cd(+), Na(-)
Cluster 8 (11 genes): Ni(+), Na(-)
Cluster 11 (72 genes): Fe(-)
Cluster 18 (15 genes): Mo(-)

Figure 5: Network analysis based on Cosine

```
net_1$plot.pnet2
```

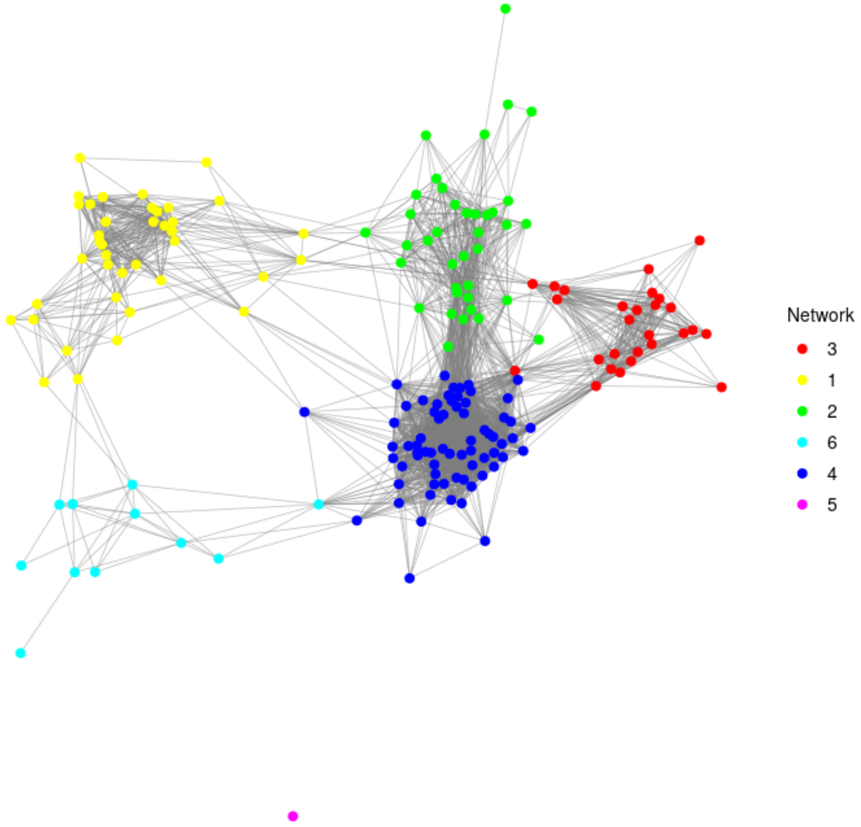

Figure 6: Network analysis based on Cosine

Use *Hybrid Mahalanobis Cosine*:

```
net_2 <- GeneNetwork(data = dat,
                     data_symb = dat_symb,
                     min_clust_size = 10,
                     thres_corr = 0.75,
                     method_corr = "mahal_cosine")
net_2$plot.pnet1
```

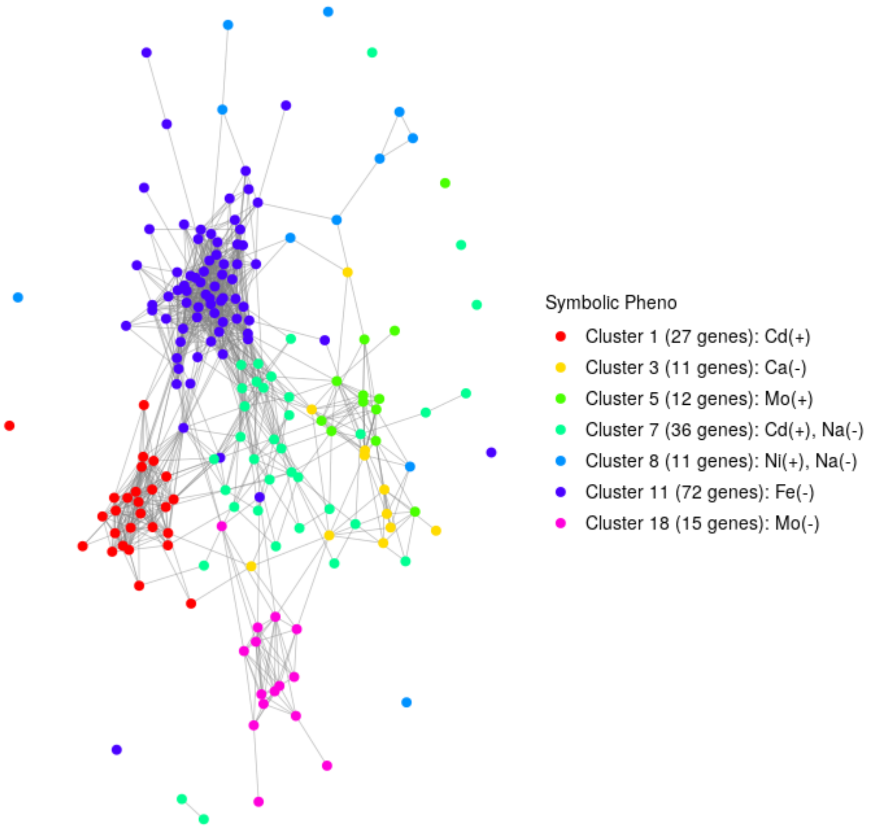

Figure 7: Network with Mahalanobis Cosine

```
net_2$plot.pnet2
```

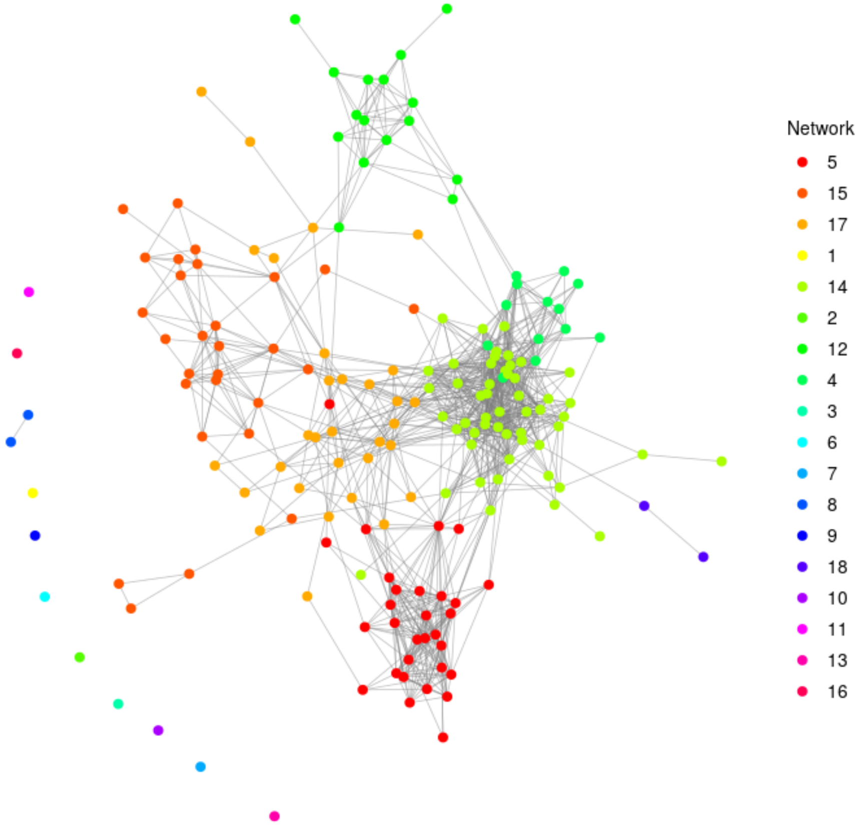

Figure 8: Network with Mahalanobis Cosine

Again, we use *Hybrid Mahalanobis Cosine*:

```
net_3 <- GeneNetwork(data = dat,
                     data_symb = dat_symb,
                     min_clust_size = 10,
                     thres_corr = 0.75,
                     method_corr = "hybrid_mahal_cosine")
net_3$plot.pnet1
```

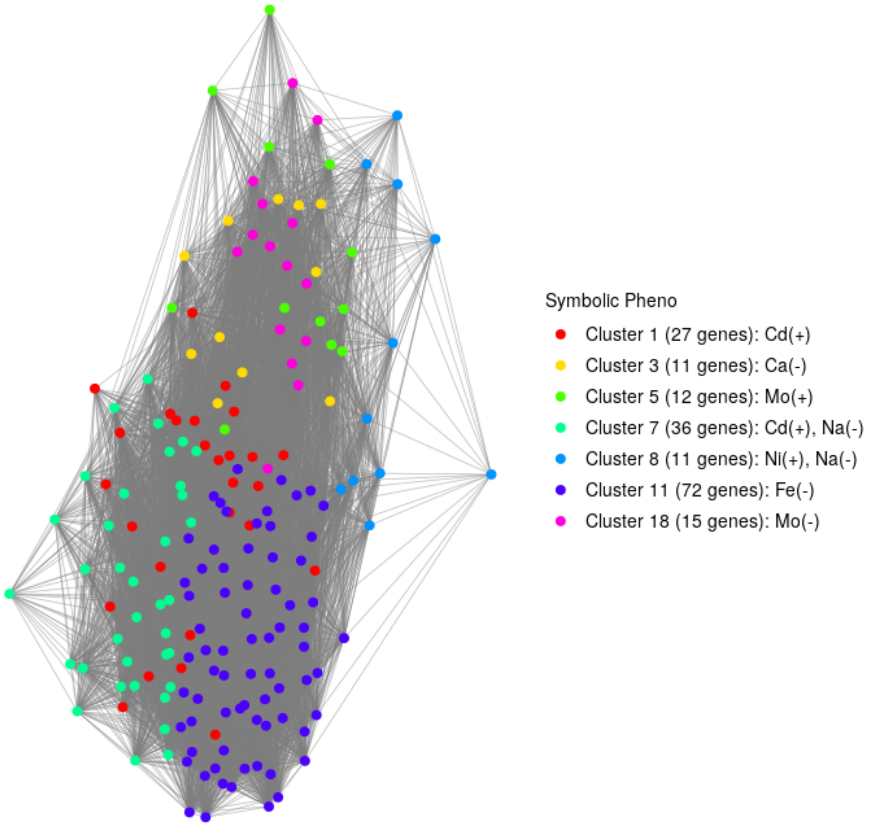
Symbolic Pheno
● Cluster 1 (27 genes): Cd(+)
● Cluster 3 (11 genes): Ca(-)
● Cluster 5 (12 genes): Mo(+)
● Cluster 7 (36 genes): Cd(+), Na(-)
● Cluster 8 (11 genes): Ni(+), Na(-)
● Cluster 11 (72 genes): Fe(-)
● Cluster 18 (15 genes): Mo(-)

Figure 9: Network with Hybrid Mahalanobis Cosine

```
net_3$plot.pnet2
```

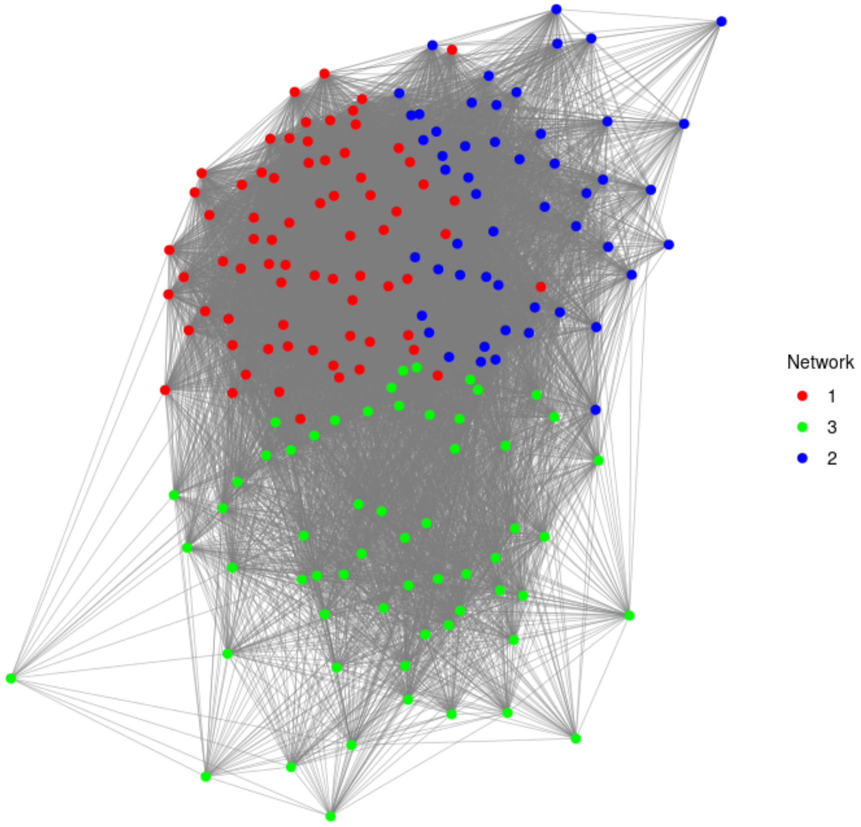

Figure 10: Network with Hybrid Mahalanobis Cosine

Enrichment analysis

The enrichment analysis is based on symbolic data clustering. The genes in clusters are considered target gene sets while genes in the whole data set is the universal gene set. The Bioconductor R package GOstats is used for the enrichment analysis.

The KEGG enrichment analysis, using a p-values of 0.05 and Genome wide annotation for Yeast, org.Sc.sgd.db:

```r
kegg <- kegg_enrich(data = dat_symb, min_clust_size = 10, pval = 0.05,
                    annot_pkg =  "org.Sc.sgd.db")

#' kegg
kegg %>%
  kable(caption = 'KEGG enrichmenat analysis',
        digits = 3, booktabs = T) %>%
  kable_styling(full_width = F, font_size = 10,
                latex_options = c("striped", "scale_down"))
```

Table 6: KEGG enrichmenat analysis

Cluster	KEGGID	Pvalue	Count	Size	Term
Cluster 18 (15 genes)	00290	0.009	2	2	Valine, leucine and isoleucine biosynthesis
Cluster 18 (15 genes)	00520	0.009	2	2	Amino sugar and nucleotide sugar metabolism
Cluster 18 (15 genes)	00260	0.012	3	6	Glycine, serine and threonine metabolism
Cluster 18 (15 genes)	00010	0.024	2	3	Glycolysis / Gluconeogenesis
Cluster 18 (15 genes)	01110	0.037	5	22	Biosynthesis of secondary metabolites
Cluster 3 (11 genes)	00400	0.009	2	2	Phenylalanine, tyrosine and tryptophan biosynthesis
Cluster 8 (11 genes)	01100	0.006	6	55	Metabolic pathways
Cluster 8 (11 genes)	00564	0.027	2	6	Glycerophospholipid metabolism

Note that there could be no results returned for KEGG enrichment analysis. Arguments such as `min_clust_size` can be changed as appropriate.

The GO Terms enrichment analysis with ontology of *BP* (other two are *MF* and *CC*):

```r
go <- go_enrich(data = dat_symb, min_clust_size = 10, pval = 0.05,
                ont = "BP", annot_pkg =  "org.Sc.sgd.db")
#' go
go %>% head() %>%
  kable(caption = 'GO Terms enrichmenat analysis',
        digits = 3, booktabs = T) %>%
  kable_styling(full_width = F, font_size = 10,
                latex_options = c("striped", "scale_down"))
```

Table 7: GO Terms enrichmenat analysis

Cluster	ID	Description	Pvalue	Count	CountUniverse	Ontology
Cluster 11 (72 genes)	GO:0051336	regulation of hydrolase activity	0.0018	4	12	BP
Cluster 11 (72 genes)	GO:0043085	positive regulation of catalytic activity	0.0044	4	15	BP
Cluster 11 (72 genes)	GO:0035303	regulation of dephosphorylation	0.0068	2	3	BP
Cluster 11 (72 genes)	GO:0046889	positive regulation of lipid biosynthetic process	0.0068	2	3	BP
Cluster 11 (72 genes)	GO:1903727	positive regulation of phospholipid metabolic process	0.0068	2	3	BP
Cluster 11 (72 genes)	GO:0044764	multi-organism cellular process	0.0074	3	9	BP

Exploratory analysis

The explanatory analysis performs PCA and correlation analysis for ions in terms of genes. Note that this analysis treats ions as samples/replicates while genes are treated as variables/features. The explanatory analysis is initially employed at an early stage of the analysis.

For example, we apply it to the pre-processed data `dat` before any other analysis:

```
expl <- ExploratoryAnalysis(data = dat)
names(expl)
#> [1] "plot.pca"        "data.pca.load"  "plot.corr"       "plot.corr.heat"
#> [5] "plot.heat"       "plot.net"
```

The PCA plot is:

```
expl$plot.pca
```

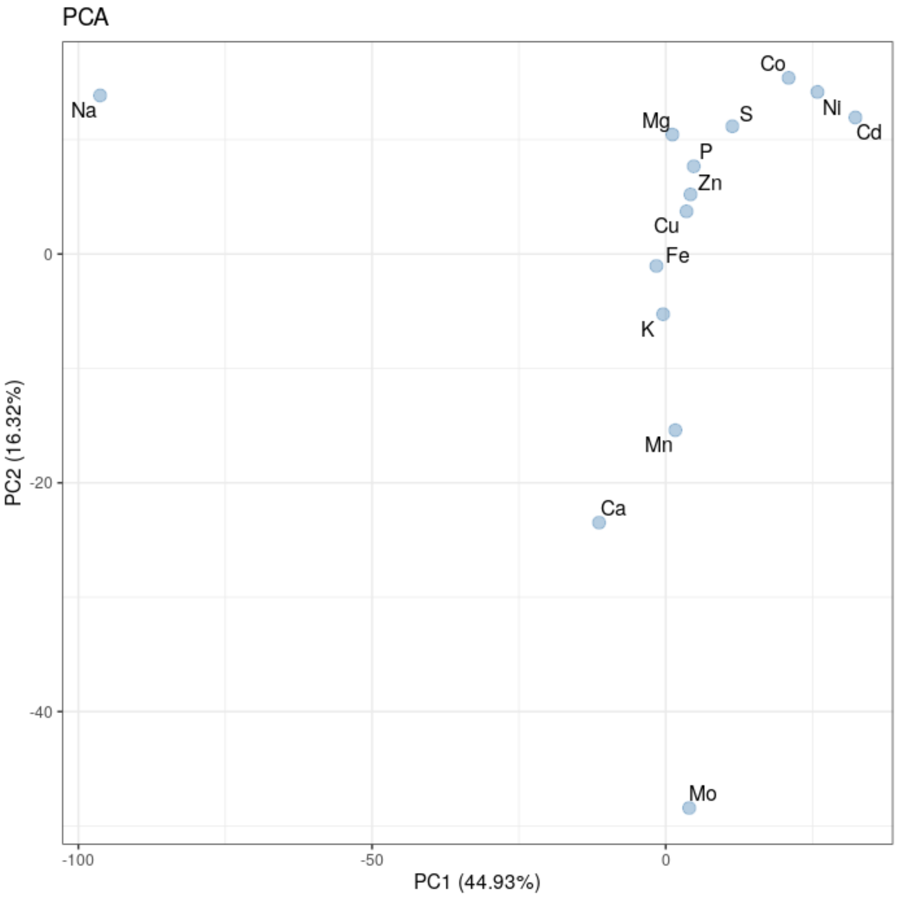

Figure 11: Ion PCA plot on pre-processed data

The Person correlation of ions are shown in correlation plot, heatmap and network plot:

```
expl$plot.corr
```

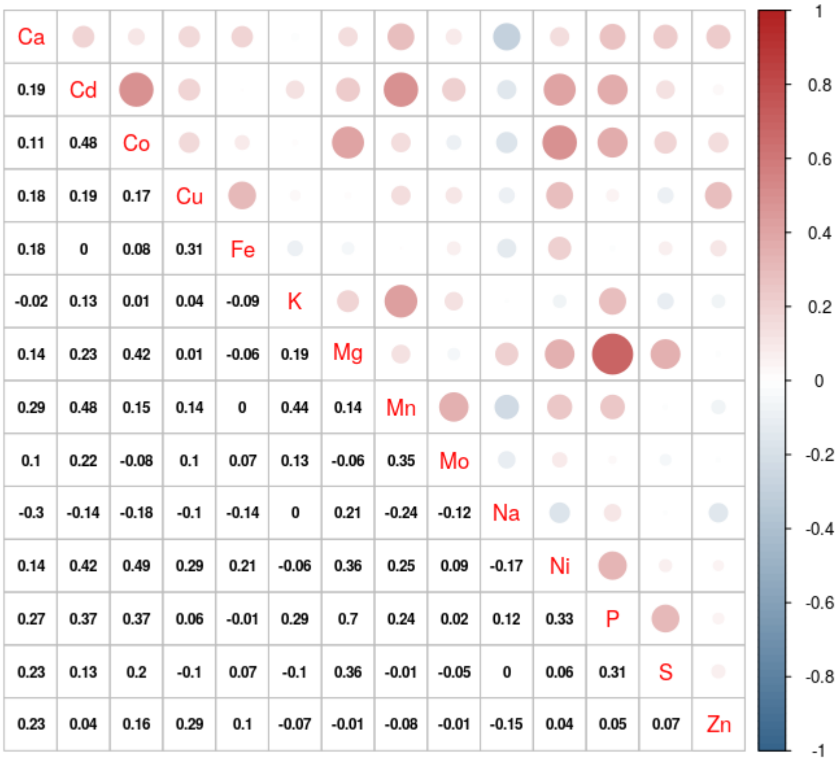

Figure 12: Ion correlation plots on pre-processed data

```
expl$plot.corr.heat
```

```
expl$plot.net
```

The correlation between ions and genes are shown in heatmap with dendrogram:

```
expl$plot.heat
```

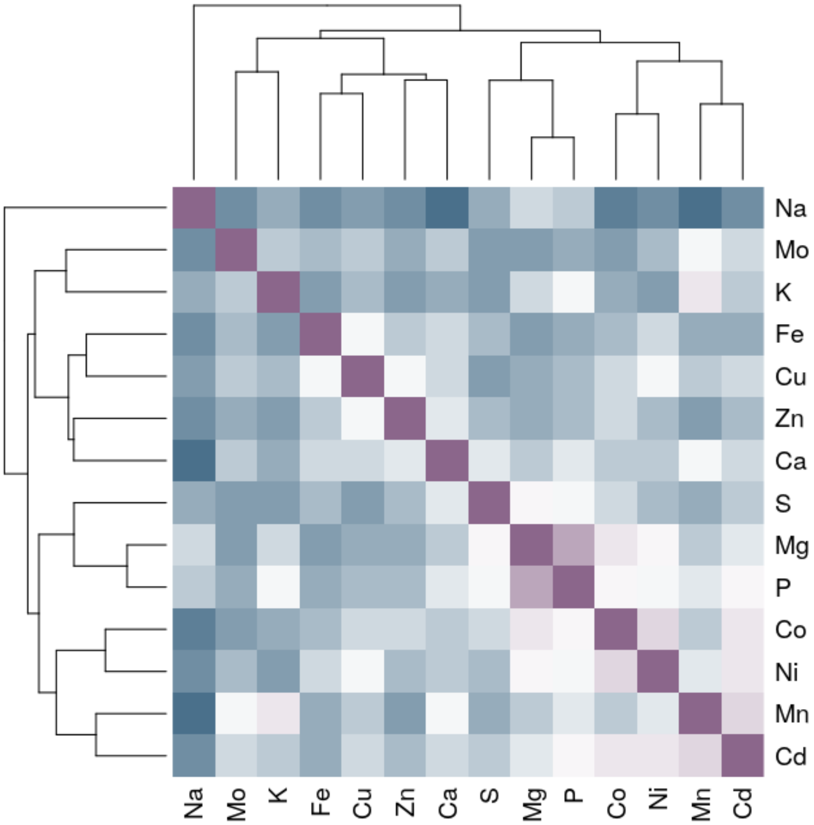

Figure 13: Ion correlation plots on pre-processed data

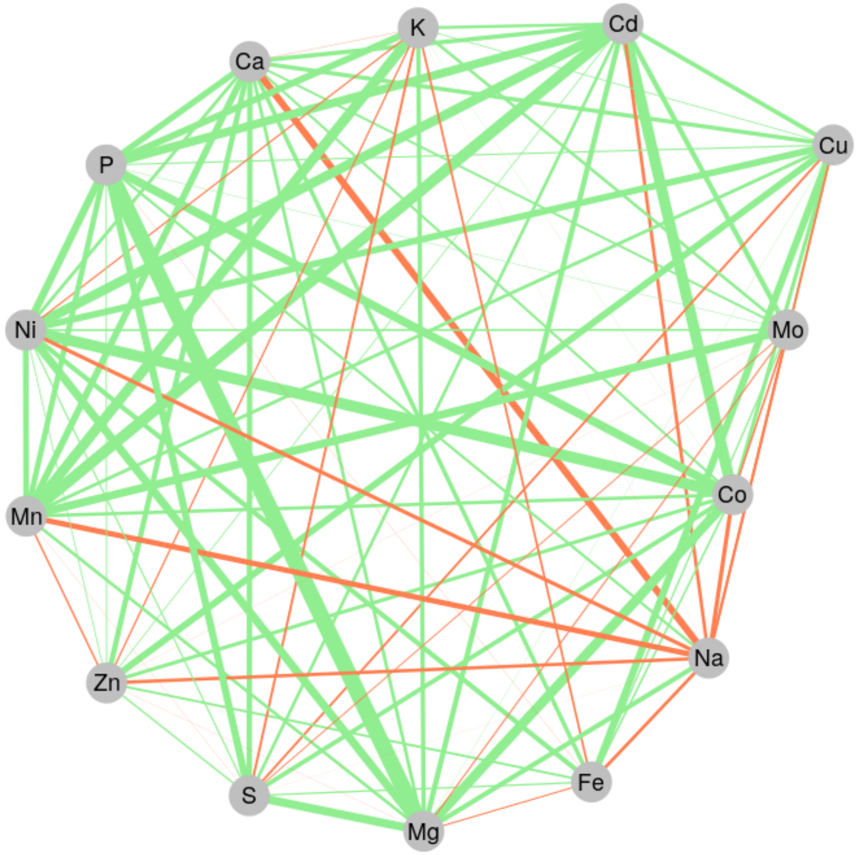

Figure 14: Ion correlation plots on pre-processed data

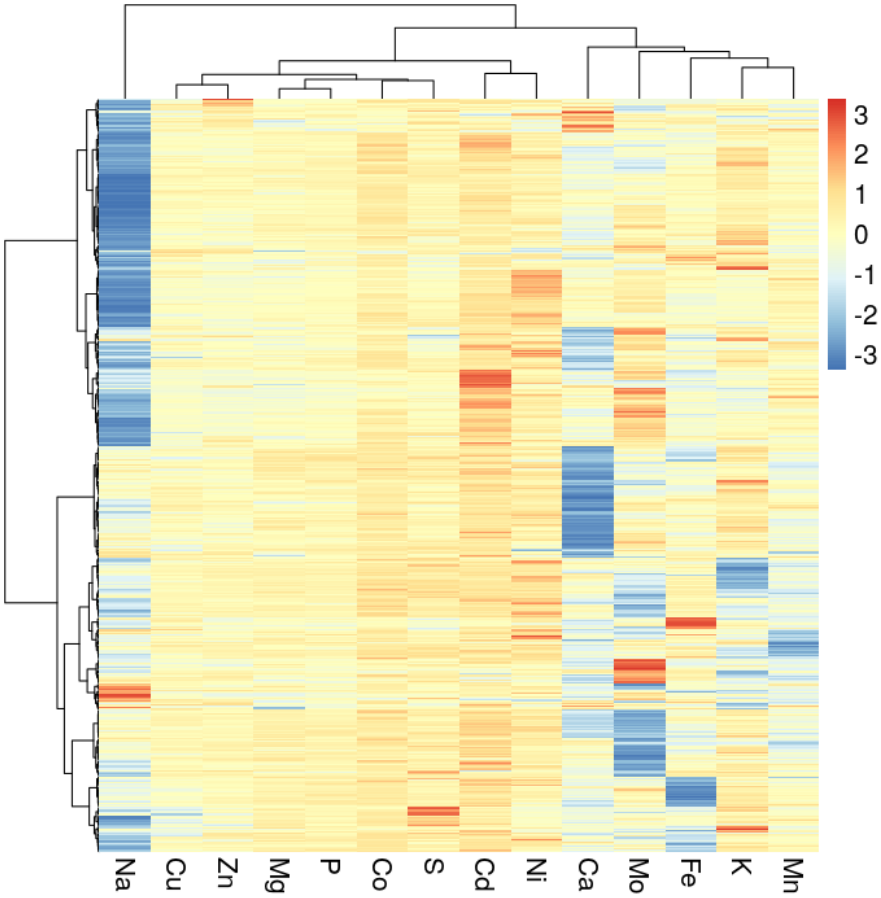

Figure 15: Correlation between ions and genes on pre-processed data

The explanatory analysis can also be used at other stages of the analysis. Here for example after gene clustering analysis:

```
#' update data set with results of gene clustering
dat_clus <- dat[clust$idx, ]
dim(dat_clus)
#> [1] 184  15

expl.1 <- ExploratoryAnalysis(data = dat_clus)
```

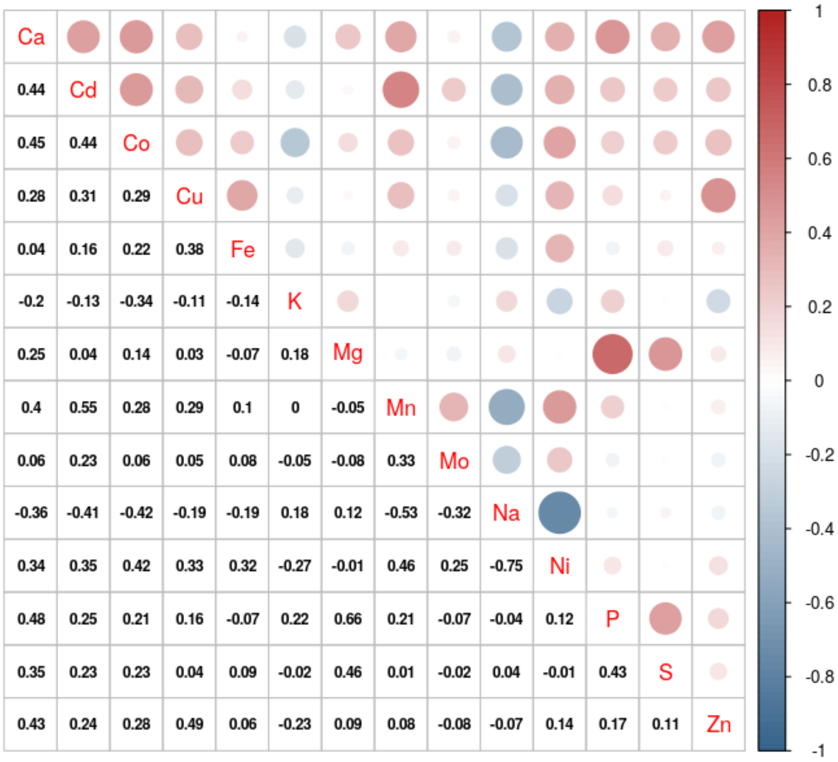

Figure 16: Explanotory analysis after gene clustering

```
expl.1$plot.pca
```

```
expl.1$plot.net
```

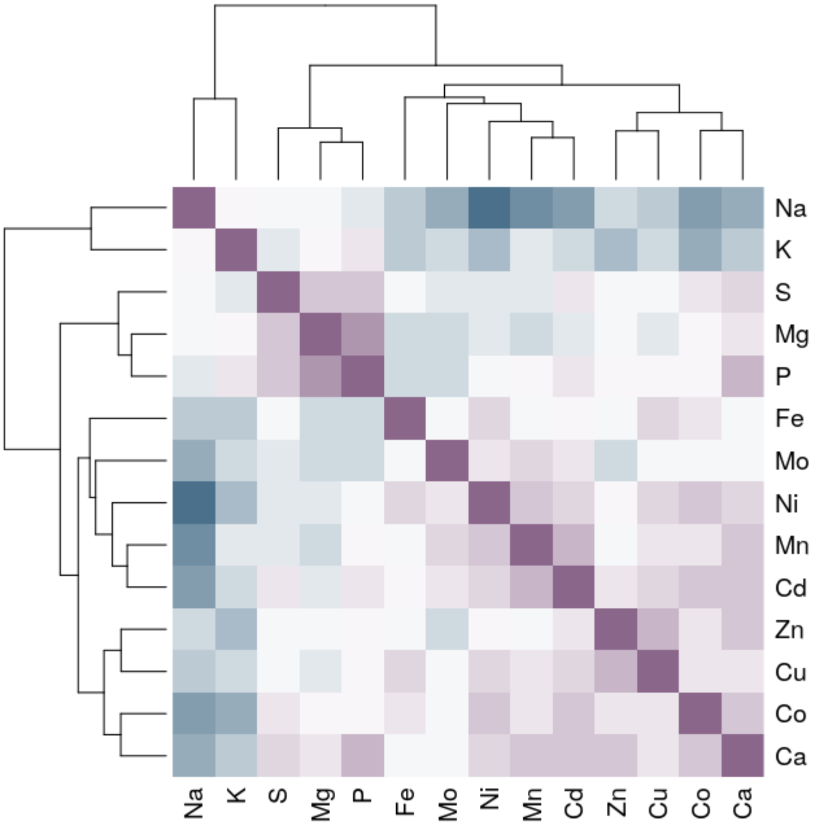

Figure 17: Explanotory analysis after gene clustering

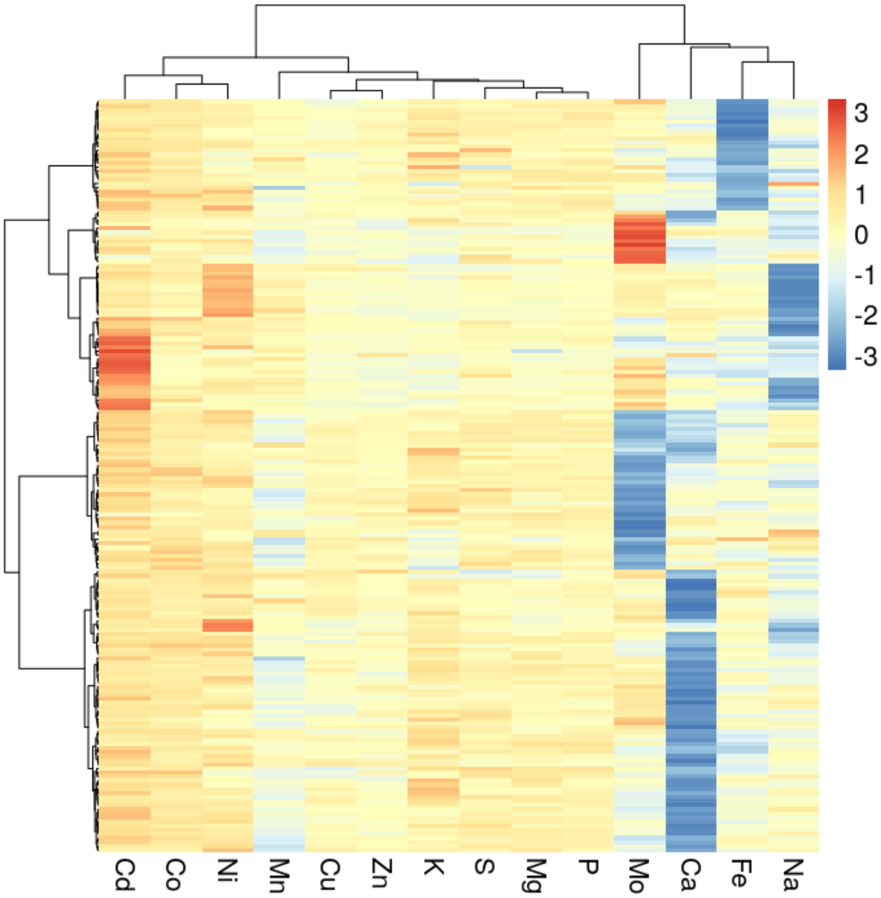

Figure 18: Explanotory analysis after gene clustering

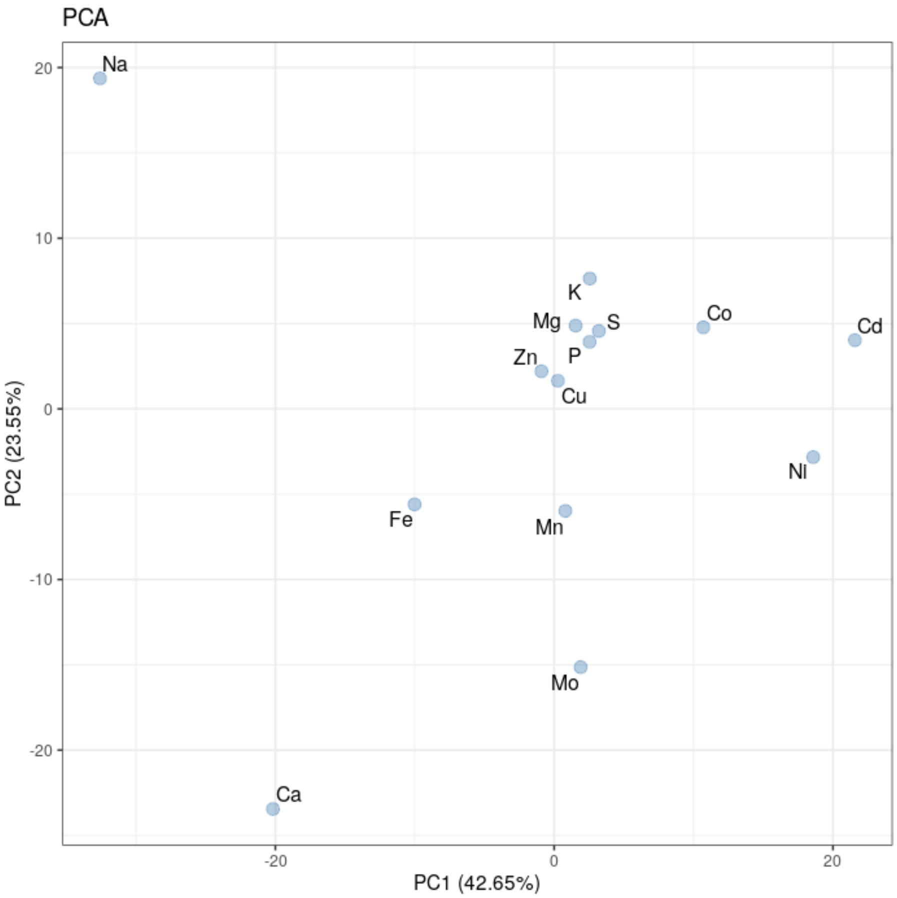

Figure 19: Explanotory analysis after gene clustering

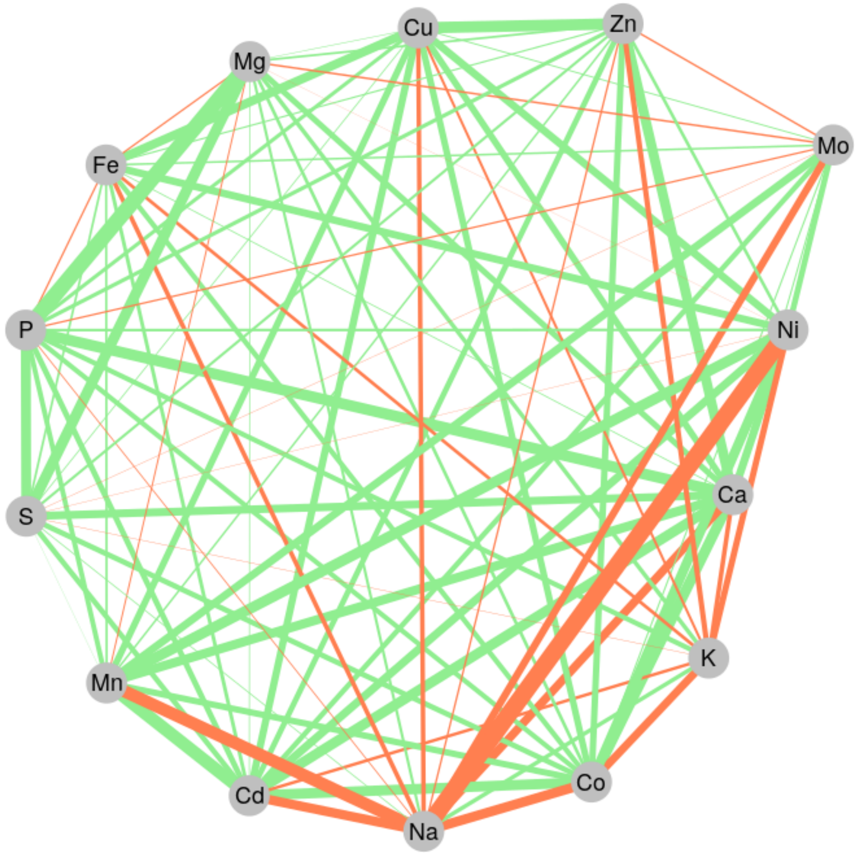

Figure 20: Explanotory analysis after gene clustering