Ionflow: network and enrichment analysis for ionomics data

Wanchang Lin

11-12-2020

This vignette explains how to perform ionomics data analysis including gene network and enrichment analysis by using a modification of the R package, [ionflow](https://github.com/AlinaPeluso/MetaboFlow). The modification(ionflow\_funcs) was made by Wanchang Lin ([w.lin@imperial.ac.uk](mailto:w.lin@imperial.ac.uk)) and Jacopo Iacovacci([j.iacovacci@imperial.ac.uk](mailto: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 : Raw data set

| Knockout | Batch\_ID | Ca | Cd | Co | Cu | Fe | K | Mg | Mn | Mo | Na | Ni | P | S | Zn |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| YBR182C | 4 | 36.99520 | 0.74339 | 0.13959 | 1.19377 | 4.43660 | 2461.762 | 348.3296 | 1.03651 | 0.42294 | 82.09721 | 0.61259 | 2379.384 | 456.1653 | 12.05238 |
| YGL202W | 87 | 42.22396 | 0.87045 | 0.13585 | 1.59650 | 8.33314 | 2849.723 | 720.0473 | 1.18075 | 0.80928 | 213.39464 | 0.84394 | 4731.093 | 641.2027 | 13.07261 |
| YDL227C | 15 | 54.62109 | 0.90417 | 0.14917 | 1.69544 | 9.32855 | 3167.710 | 706.5190 | 1.50680 | 1.46482 | 264.19360 | 1.37324 | 4517.556 | 528.1594 | 19.89572 |
| YDL227C | 86 | 34.58320 | 0.97531 | 0.13316 | 1.63095 | 7.42064 | 3052.041 | 722.4106 | 1.34508 | 0.68742 | 419.32539 | 1.21114 | 5005.570 | 718.2416 | 13.22052 |
| YDR344C | 10 | 11.45636 | 0.99586 | 0.20203 | 1.82212 | 2.47614 | 2254.528 | 360.7966 | 1.04301 | 1.07686 | 54.96136 | 1.22636 | 2736.286 | 223.5428 | 13.83899 |
| YDR057W | 7 | 38.85431 | 0.96496 | 0.18536 | 1.29088 | 5.69300 | 2663.081 | 528.6942 | 1.09963 | 0.81794 | 129.84004 | 0.90016 | 3534.702 | 537.9017 | 20.30433 |
| YJL129C | 64 | 14.55434 | 1.00229 | 0.15487 | 1.16946 | 7.08721 | 4063.749 | 746.0223 | 1.12801 | 0.76191 | 138.35950 | 1.12249 | 5089.530 | 701.1228 | 16.05329 |
| YDL227C | 2 | 32.44068 | 0.75940 | 0.15578 | 1.80467 | 10.90018 | 2497.658 | 340.4322 | 1.06547 | 1.29936 | 172.23733 | 1.08352 | 2891.797 | 360.5723 | 15.97490 |
| YOR253W | 26 | 46.84940 | 1.88677 | 0.16709 | 1.79883 | 9.11405 | 2470.386 | 815.4674 | 1.58297 | 0.87943 | 75.55065 | 1.89476 | 5652.308 | 665.1655 | 17.98825 |
| YDL227C | 7 | 64.90008 | 0.95890 | 0.18957 | 1.69918 | 5.74065 | 2964.090 | 724.4207 | 1.41358 | 0.58513 | 322.08494 | 0.97527 | 4541.059 | 767.5384 | 23.62677 |

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

Table : User STD file

| Ion | sd |
| --- | --- |
| Ca | 0.1508 |
| Cd | 0.0573 |
| Co | 0.0580 |
| Cu | 0.0735 |
| Fe | 0.1639 |
| K | 0.0940 |
| Mg | 0.0597 |
| Mn | 0.0771 |
| Mo | 0.1142 |
| Na | 0.1075 |
| Ni | 0.0784 |
| P | 0.0597 |
| S | 0.0801 |
| 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

Table : Processed data summary

| Ion | Min. | 1st Qu. | Median | Mean | 3rd Qu. | Max. | Variance |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Ca | -4.448 | -0.277 | -0.133 | -0.120 | 0.022 | 2.347 | 0.109 |
| Cd | -1.704 | 0.029 | 0.105 | 0.108 | 0.174 | 0.929 | 0.028 |
| Co | -2.800 | 0.019 | 0.085 | 0.061 | 0.146 | 1.603 | 0.049 |
| Cu | -0.665 | -0.098 | -0.029 | -0.015 | 0.035 | 5.278 | 0.039 |
| Fe | -7.476 | -0.174 | -0.058 | -0.020 | 0.066 | 6.885 | 0.139 |
| K | -2.212 | -0.170 | -0.008 | -0.078 | 0.091 | 1.832 | 0.078 |
| Mg | -1.844 | -0.062 | 0.006 | -0.015 | 0.067 | 1.693 | 0.033 |
| Mn | -4.108 | -0.236 | -0.080 | -0.131 | 0.007 | 1.784 | 0.060 |
| Mo | -2.029 | -0.262 | -0.076 | -0.083 | 0.087 | 4.445 | 0.130 |
| Na | -7.414 | -0.530 | -0.223 | -0.328 | -0.036 | 1.246 | 0.237 |
| Ni | -2.400 | -0.013 | 0.086 | 0.119 | 0.214 | 7.901 | 0.120 |
| P | -1.178 | -0.056 | 0.003 | -0.010 | 0.058 | 1.448 | 0.020 |
| S | -2.384 | -0.032 | 0.053 | 0.063 | 0.165 | 2.375 | 0.045 |
| Zn | -0.462 | -0.080 | -0.029 | -0.010 | 0.029 | 4.599 | 0.020 |

The pre-processed data are like this:

pre$data.gene.zscores %>% head()

Table : Processed data

| Line | Ca | Cd | Co | Cu | Fe | K | Mg | Mn | Mo | Na | Ni | P | S | Zn |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| YAL004W | -1.162359 | 0.7524514 | 1.1859842 | -0.4726908 | 0.04034395 | 0.6119824 | 0.51347974 | -0.8412151 | -0.07712908 | -1.8431339 | 1.710947233 | 0.51921694 | 0.33115731 | -0.091006640 |
| YAL005C | -1.673185 | 0.8365862 | 0.5506768 | 0.5833309 | -2.79443362 | 0.5945925 | 0.30972925 | -1.1588944 | -1.41957485 | -0.1161533 | 1.478781153 | 0.72989597 | 0.12577956 | -0.130834509 |
| YAL007C | -2.120494 | 0.6404126 | 0.2331309 | -0.5266910 | -0.24406106 | 0.7946849 | -0.09149687 | -0.1361174 | 1.21825939 | -0.9170735 | 0.004805424 | 0.08745428 | -0.29378038 | -0.649004917 |
| YAL008W | -2.341054 | 1.1262364 | 0.2124286 | -0.7265787 | -2.15578745 | 0.5222401 | -0.02090207 | -0.8663923 | 0.92837151 | -0.5750513 | 0.016280970 | -0.09175176 | -0.72749574 | -0.473533507 |
| YAL009W | -1.184548 | 0.6593268 | 0.5485459 | -1.1132635 | -3.91312743 | 0.2178748 | 0.08980504 | -0.1803018 | 1.49568583 | -0.8370121 | -0.087715257 | 0.14454347 | 0.00870707 | -0.361045566 |
| YAL010C | -1.278552 | 1.4259143 | 2.2683778 | 0.4610008 | 1.52953958 | -2.7538515 | 0.03635921 | -0.7358068 | -9.70996958 | -4.3041286 | 2.422126014 | -0.98365280 | -0.04538613 | -0.009771027 |

And the symbolic data are like this:

pre$data.gene.symb %>% head()

Table : 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()

Table : 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 |

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:

pre$plot.hist

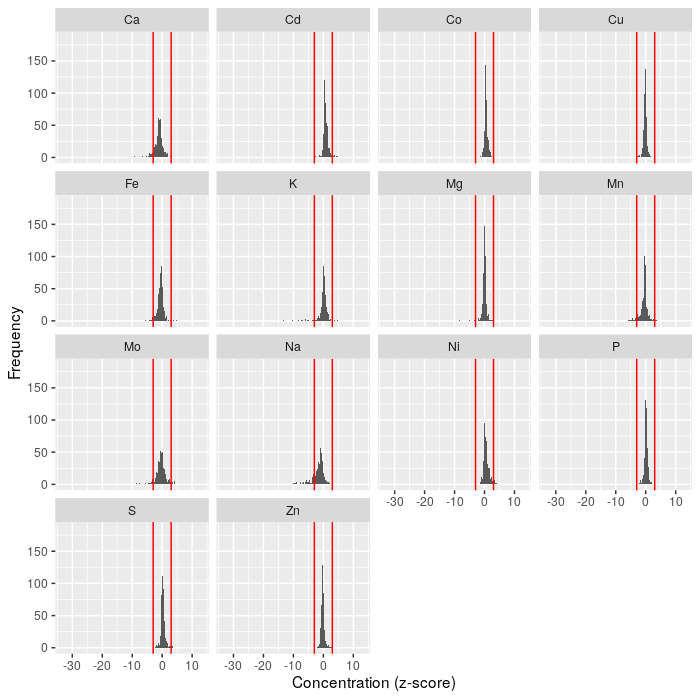


Figure : Ionomcs data distribution plot

## 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

Table : Cluster table

| cluster | nGenes |
| --- | --- |
| 11 | 72 |
| 7 | 36 |
| 1 | 27 |
| 18 | 15 |
| 5 | 12 |
| 3 | 11 |
| 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](https://bit.ly/3oED6hj). For the last two methods, see publication: [Extraction and Integration of Genetic Networks from Short-Profile Omic Data Sets](https://www.mdpi.com/2218-1989/10/11/435/htm) for details.

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

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:

net$plot.pnet1

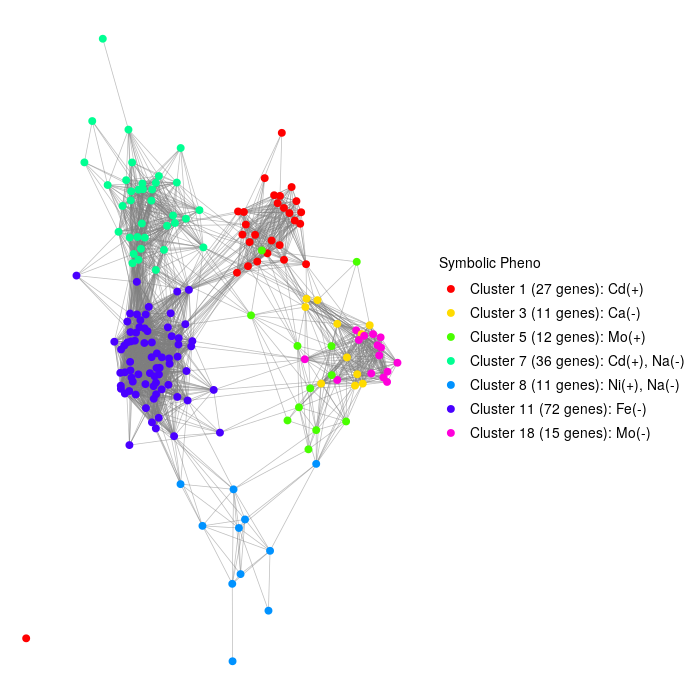


Figure : Network with Pearson correlation: symbolic clustering

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

net$plot.pnet2

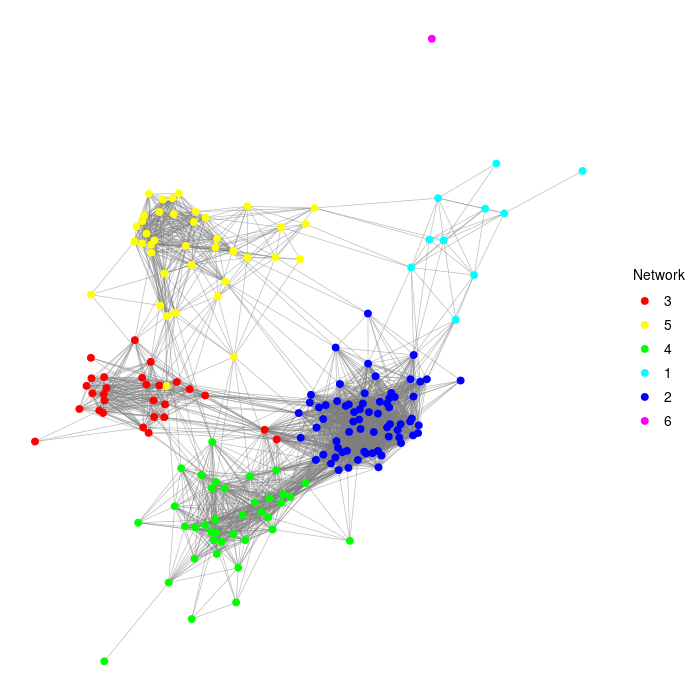


Figure : Network with Pearson correlation: community detction

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

net$plot.impact\_betweenness

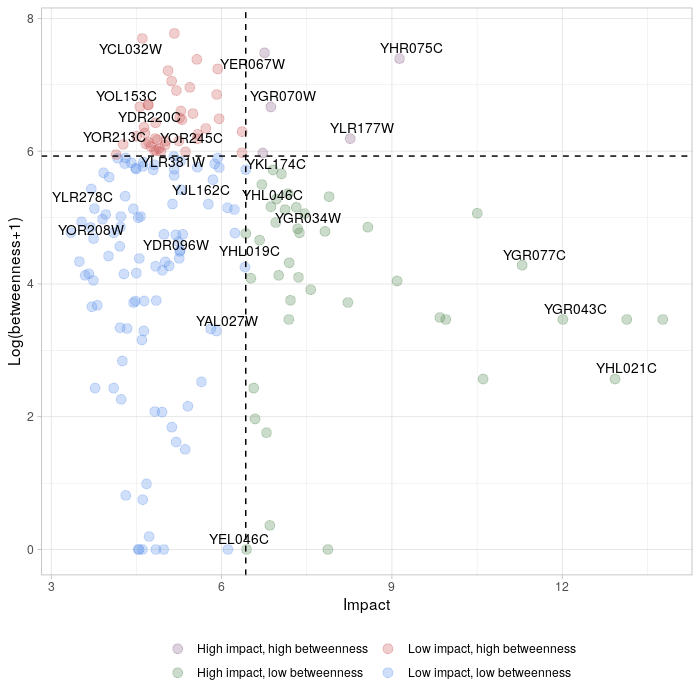


Figure : Network with Pearson correlation: impact and betweenness

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

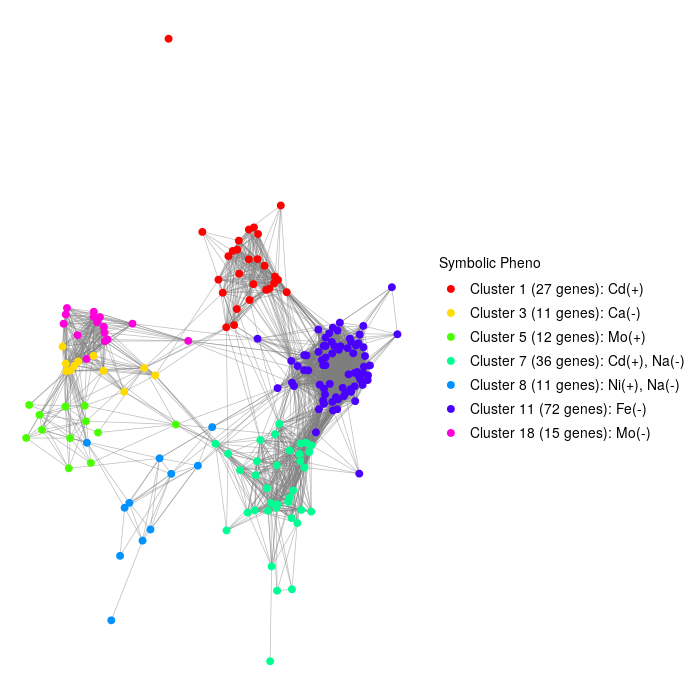


Figure : Network analysis based on Cosine

net\_1$plot.pnet2

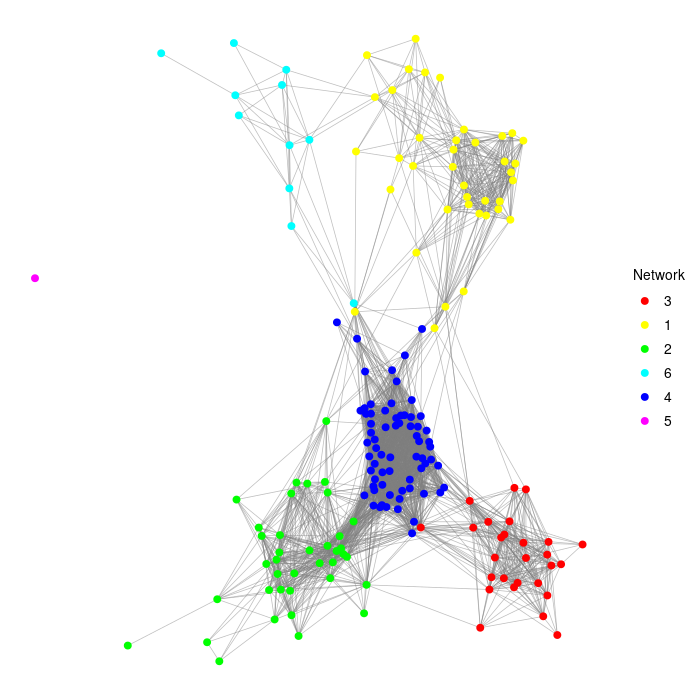


Figure : Network analysis based on Cosine

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

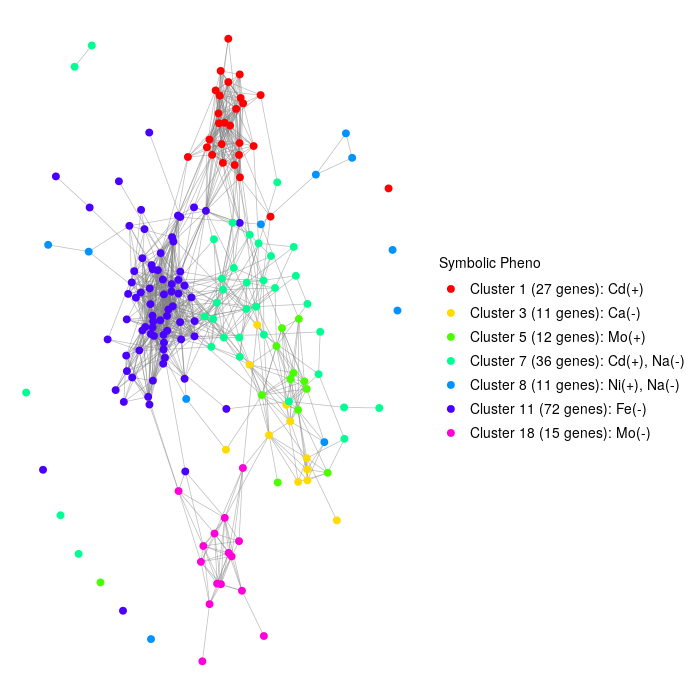


Figure : Network with Mahalanobis Cosine

net\_2$plot.pnet2

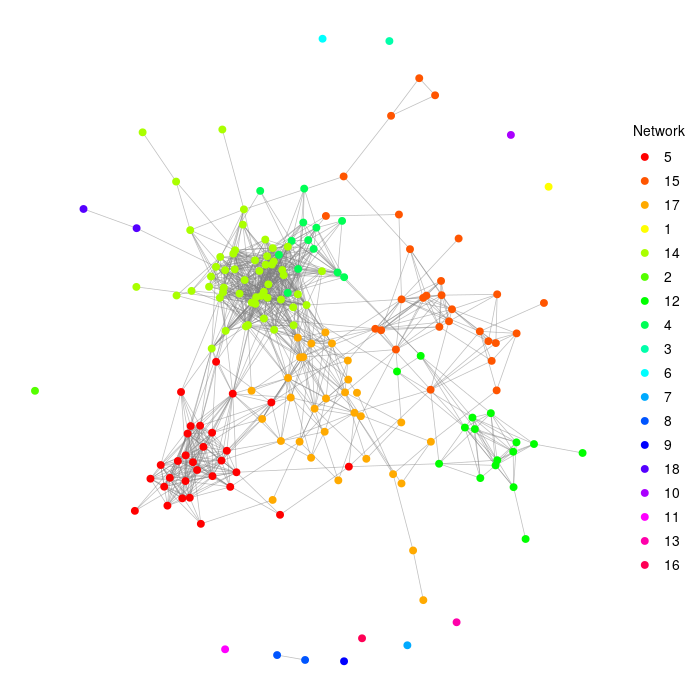


Figure : Network with Mahalanobis Cosine

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

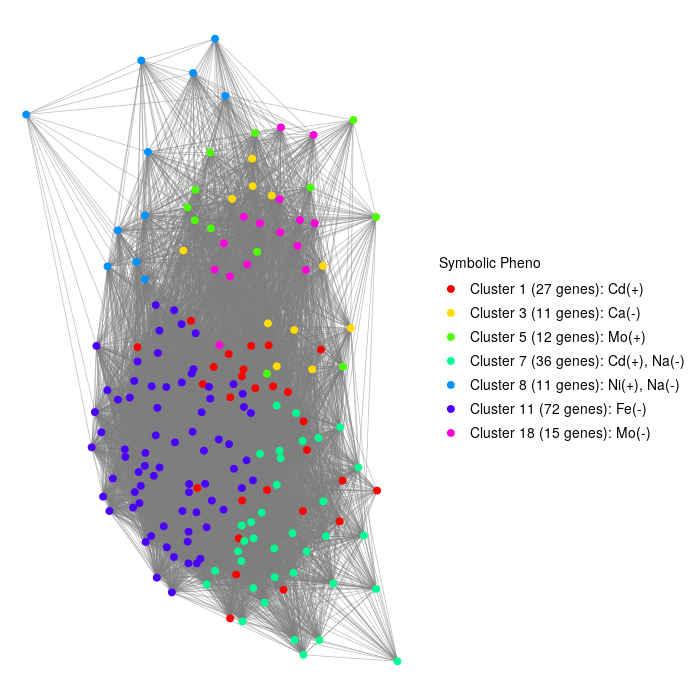


Figure : Network with Hybrid Mahalanobis Cosine

net\_3$plot.pnet2

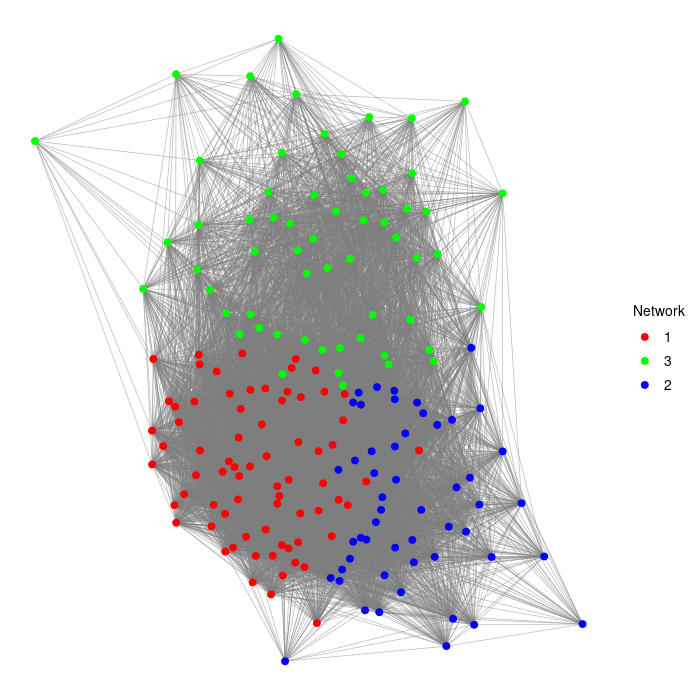


Figure : Network with Hybrid Mahalanobis Cosine

## 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](http://bioconductor.org/packages/release//bioc/html/GOstats.html) 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](https://bioconductor.org/packages/release/data/annotation/html/org.Sc.sgd.db.html):

kegg <- kegg\_enrich(data = dat\_symb, min\_clust\_size = 10, pval = 0.05,  
 annot\_pkg = "org.Sc.sgd.db")  
kegg

Table : KEGG enrichment analysis

| Cluster | KEGGID | Pvalue | Count | Size | Term |
| --- | --- | --- | --- | --- | --- |
| Cluster 18 (15 genes) | 00290 | 0.008516129 | 2 | 2 | Valine, leucine and isoleucine biosynthesis |
| Cluster 18 (15 genes) | 00520 | 0.008516129 | 2 | 2 | Amino sugar and nucleotide sugar metabolism |
| Cluster 18 (15 genes) | 00260 | 0.011668529 | 3 | 6 | Glycine, serine and threonine metabolism |
| Cluster 18 (15 genes) | 00010 | 0.024163651 | 2 | 3 | Glycolysis / Gluconeogenesis |
| Cluster 18 (15 genes) | 01110 | 0.036608564 | 5 | 22 | Biosynthesis of secondary metabolites |
| Cluster 3 (11 genes) | 00400 | 0.008516129 | 2 | 2 | Phenylalanine, tyrosine and tryptophan biosynthesis |
| Cluster 8 (11 genes) | 01100 | 0.006180343 | 6 | 55 | Metabolic pathways |
| Cluster 8 (11 genes) | 00564 | 0.026583592 | 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*):

go <- go\_enrich(data = dat\_symb, min\_clust\_size = 10, pval = 0.05,  
 ont = "BP", annot\_pkg = "org.Sc.sgd.db")  
head(go)

Table : GP Terms enrichment 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 exploratory 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 exploratory 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

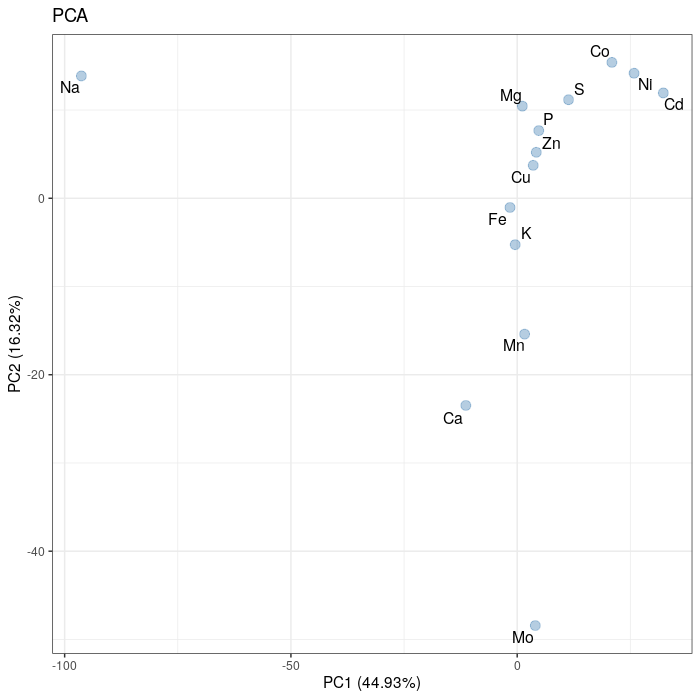


Figure : Ion PCA plot on pre-processed data

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

expl$plot.corr

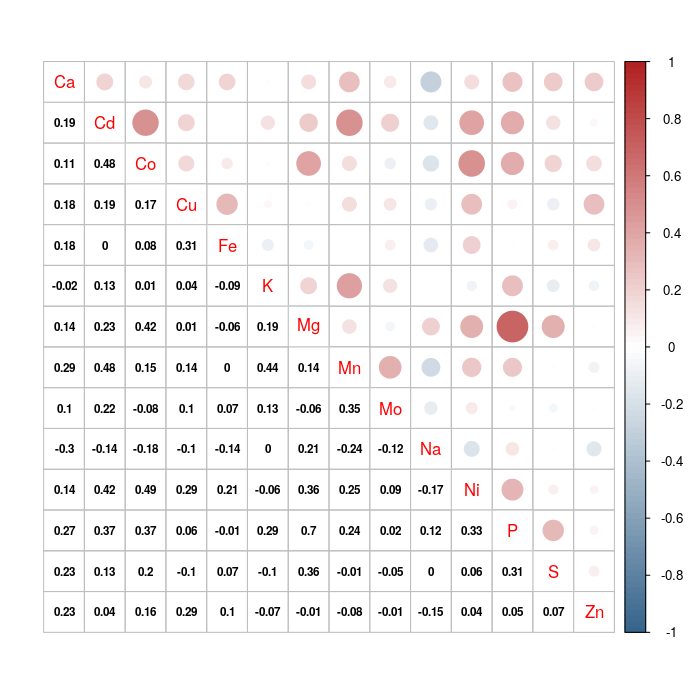


Figure : Ion correlation plots on pre-processed data

expl$plot.corr.heat

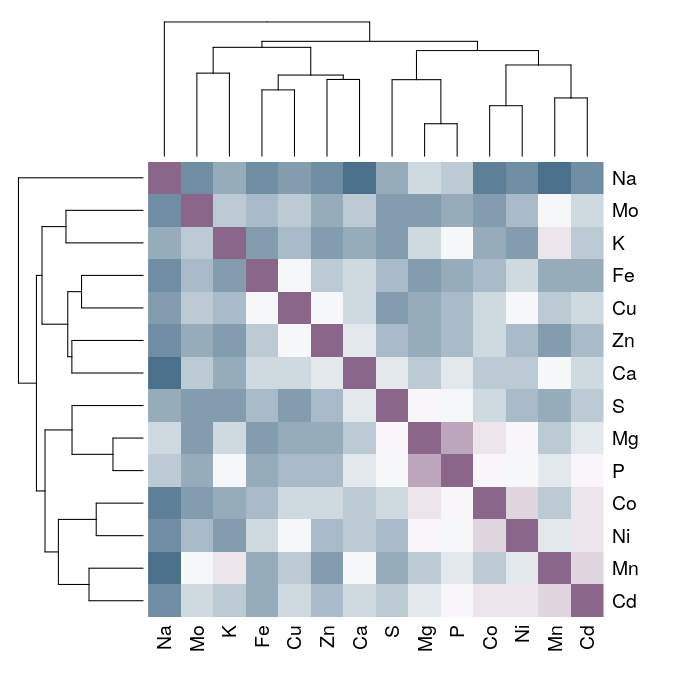


Figure : Ion correlation plots on pre-processed data

expl$plot.net

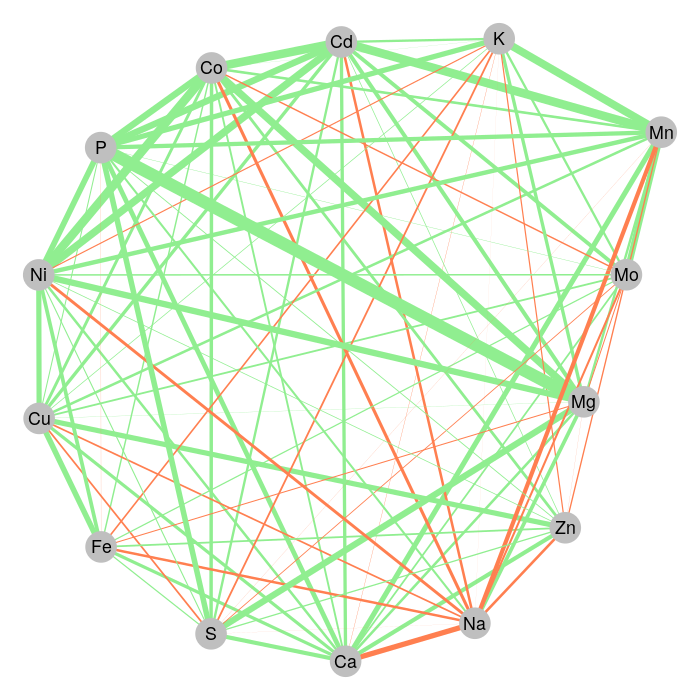


Figure : Ion correlation plots on pre-processed data

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

expl$plot.heat

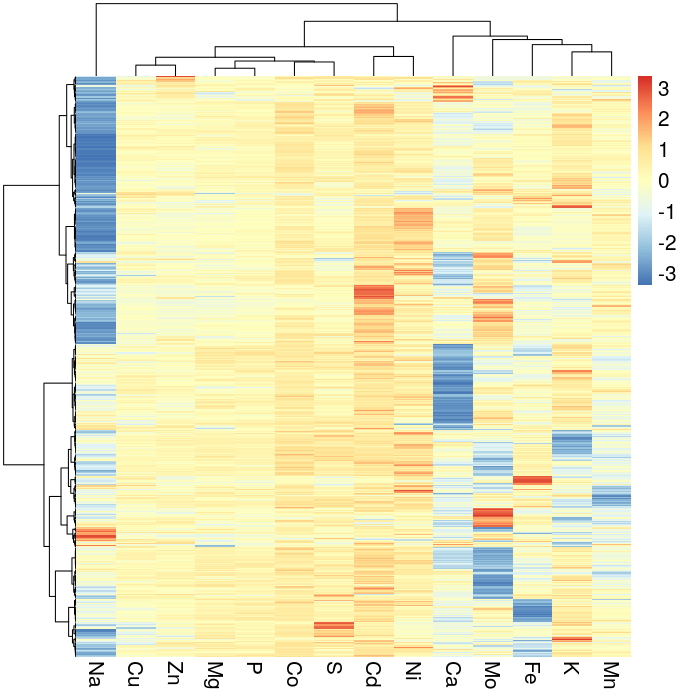


Figure : Correlation between ions and genes on pre-processed data

The exploratory 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)

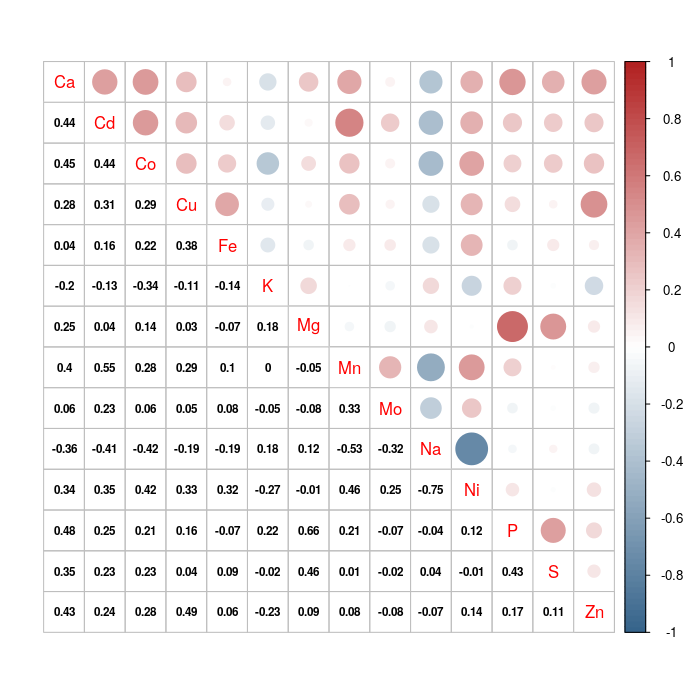


Figure : Exploratory analysis after gene clustering

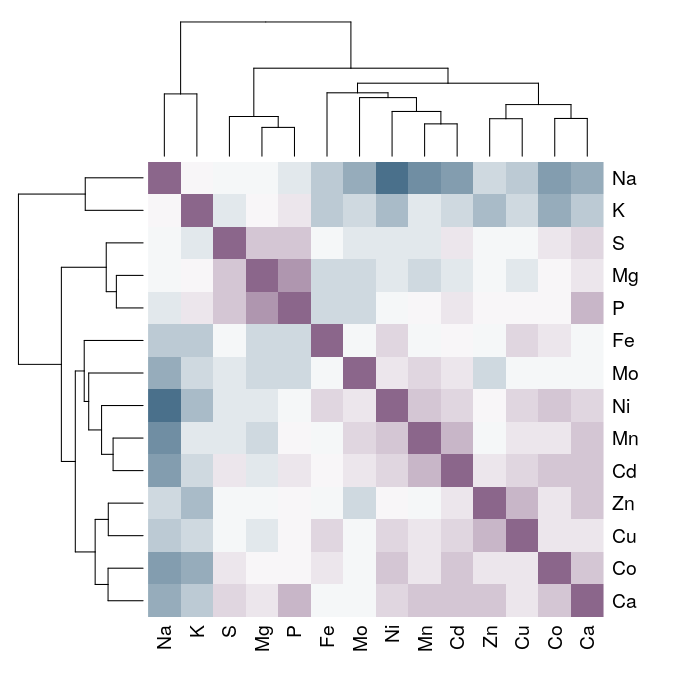


Figure : Exploratory analysis after gene clustering

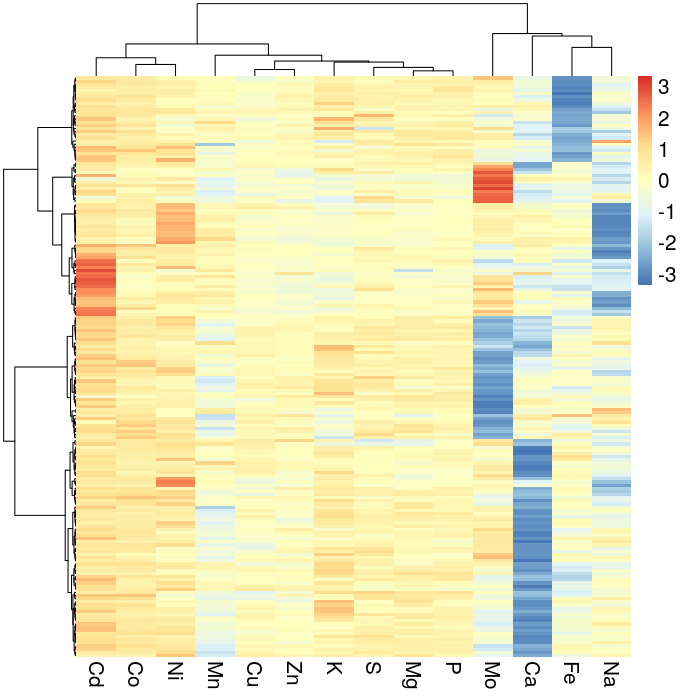


Figure : Exploratory analysis after gene clustering

expl.1$plot.pca

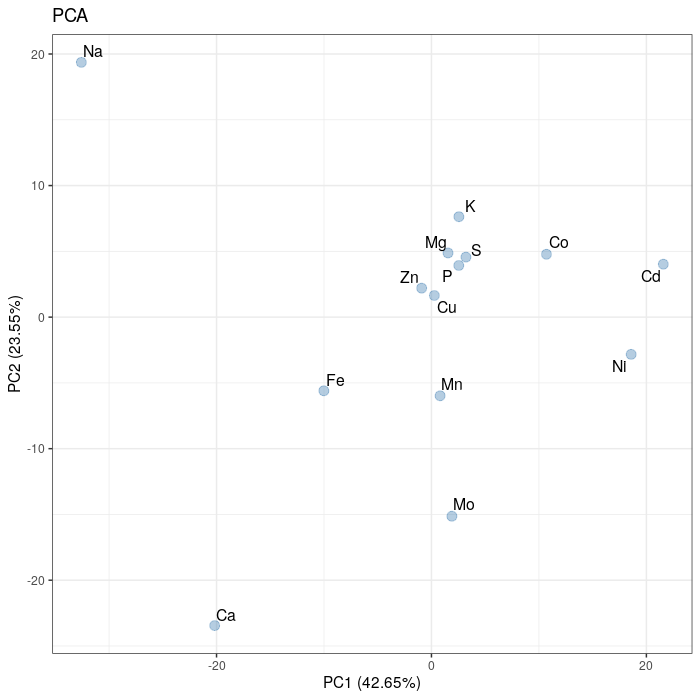


Figure : Exploratory analysis after gene clustering

expl.1$plot.net

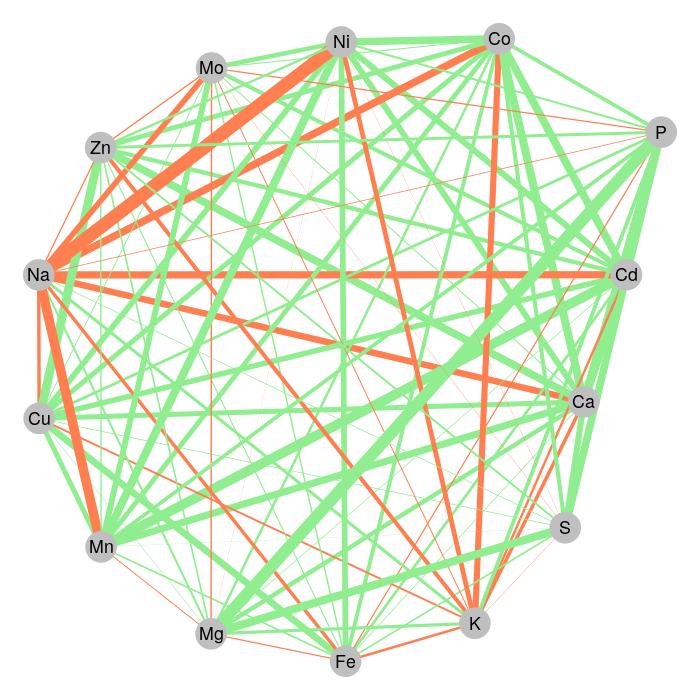


Figure : Exploratory analysis after gene clustering