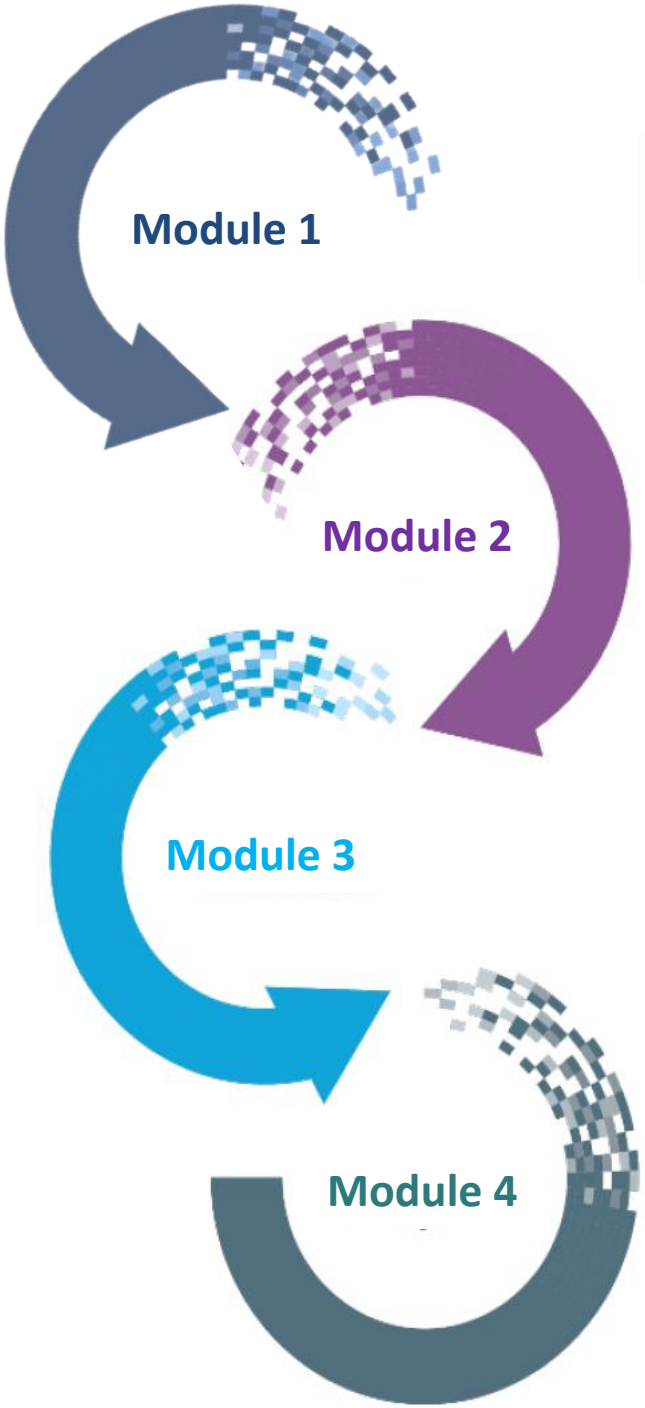
Module 2:
Network Analysis



Exploratory Data Analysis

- Importing data
- Pre-processing phase
- Filtering phase

Network Analysis

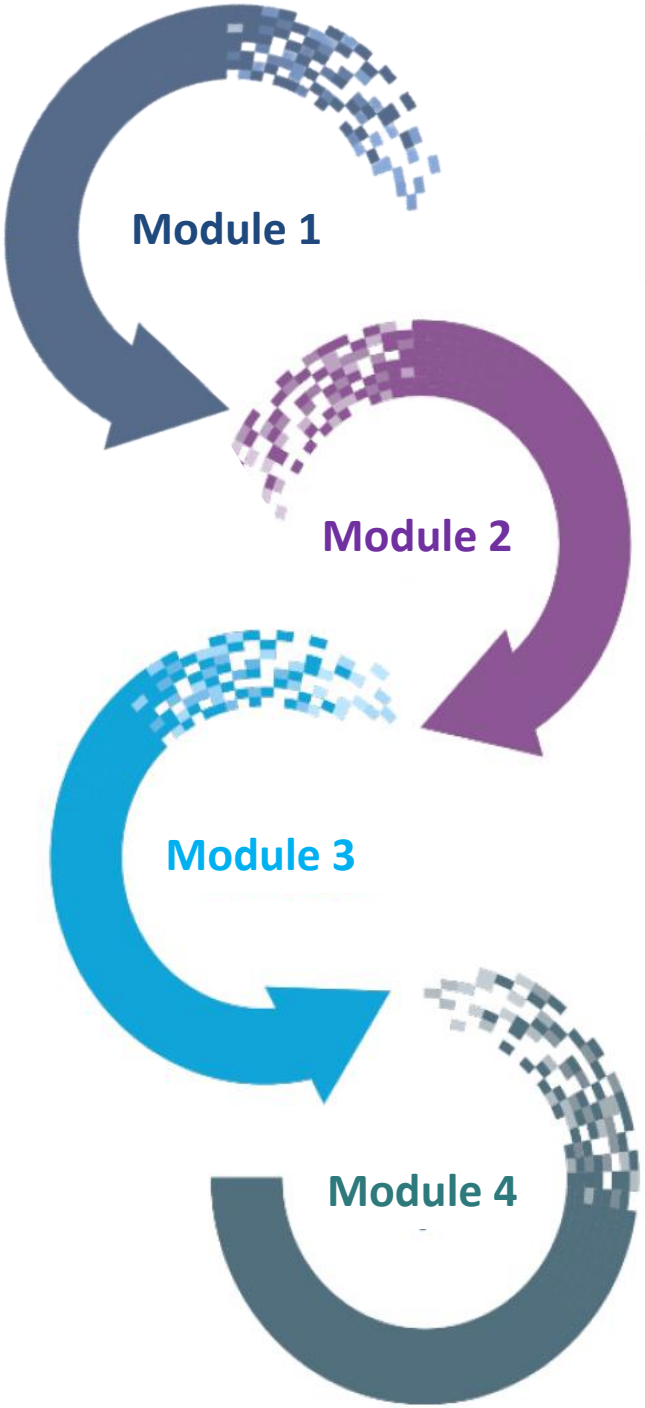
- Building the correlation network
- Checking network integrity
- Finding communities in the network

Switch Mining

- Building the heat cartography map
- Identification of switch genes

Resilience Analysis

- Evaluation of network resilience



Exploratory Data Analysis

- Importing data
- Pre-processing phase
- Filtering phase

Network Analysis

- Building the correlation network
- Checking network integrity
- Finding communities in the network

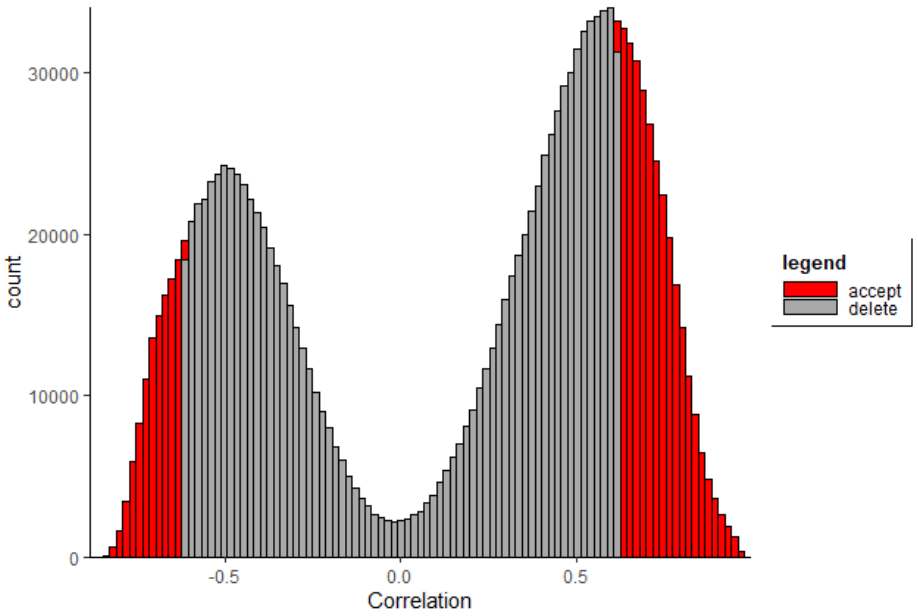
Switch Mining

- Building the heat cartography map
- Identification of switch genes

Resilience Analysis

- Evaluation of network resilience

Pearson correlation coefficient distribution



- x-axis represents the Person correlation coefficient between the expression profiles of all pairs of genes
- y-axis represents the frequency
- red bars are the selected highly correlated pairs
- grey bars are the deleted poorly correlated pairs

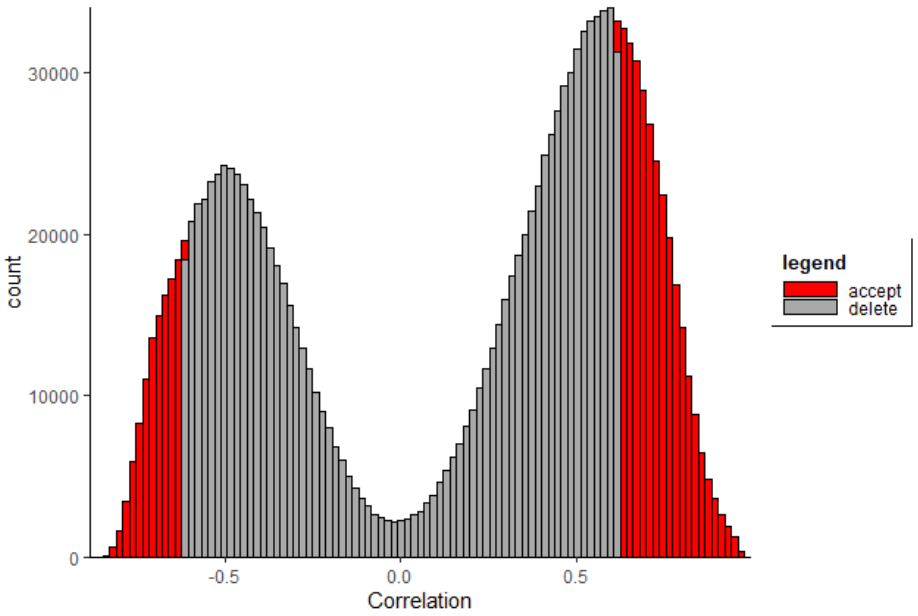
Correlation between two variables

- The existence of a linear relationship between two normally distributed continuous variables (e.g., gene expression values of gene X and gene Y) can be expressed by the **Pearson correlation coefficient ρ**

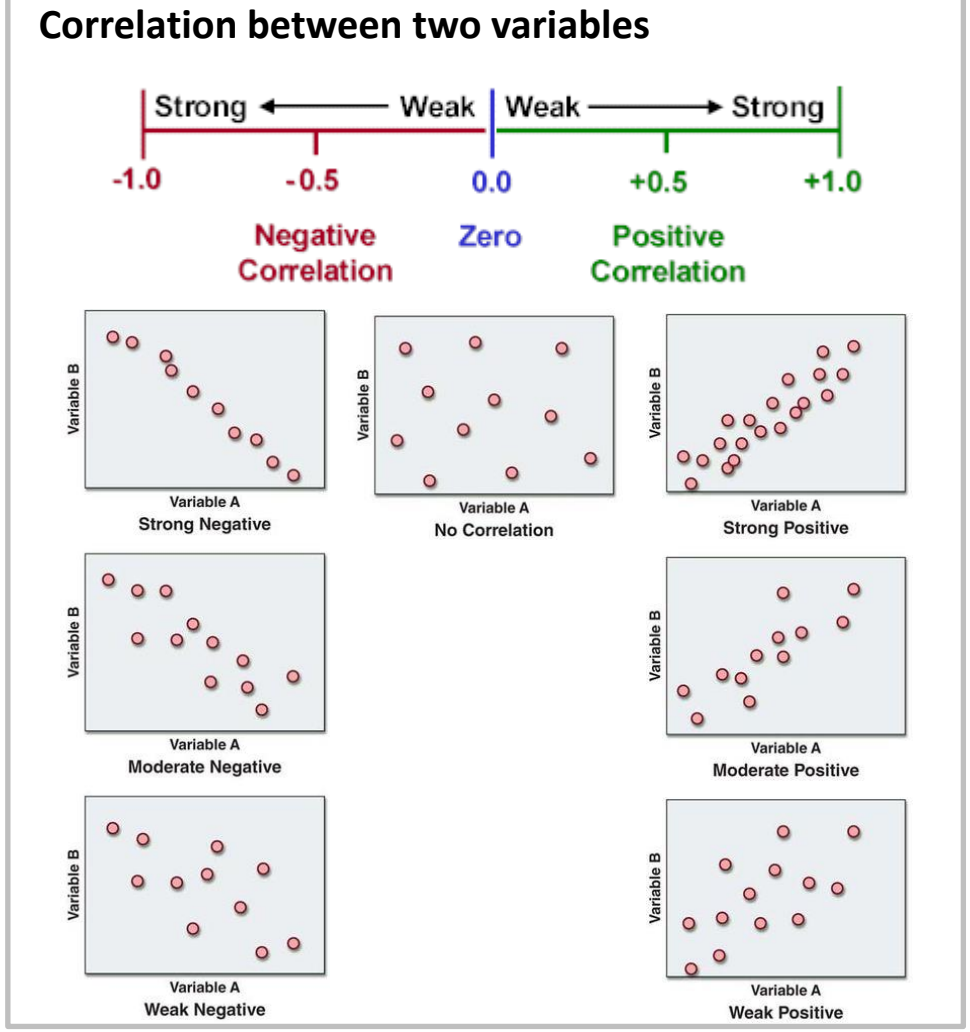
$$\rho_{xy} = \frac{\sum_{i=1}^{col}\left(x_i - \bar{x}\right)\left(y_i - \bar{y}\right)}{\sqrt{\sum_{i=1}^{col}\left(x_i - \bar{x}\right)^2 \sum_{i=1}^{col}\left(y_i - \bar{y}\right)^2}}$$

- ρ coefficient varies between -1 and 1 and shows strength (value) and direction (sign) of correlation

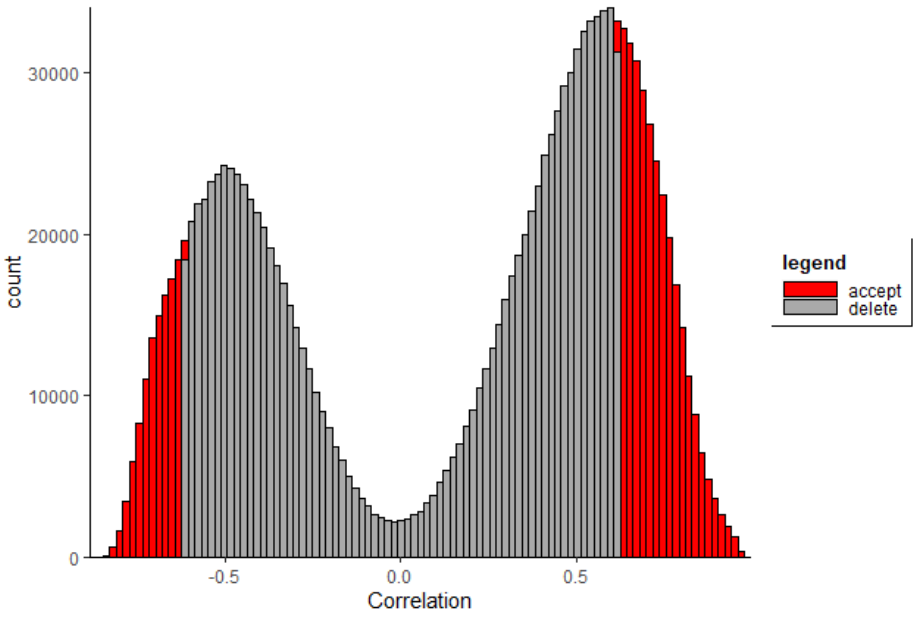
Pearson correlation coefficient distribution



- x-axis represents the Person correlation coefficient between the expression profiles of all pairs of genes
- y-axis represents the frequency
- red bars are the selected highly correlated pairs
- grey bars are the deleted poorly correlated pairs



Pearson correlation coefficient distribution

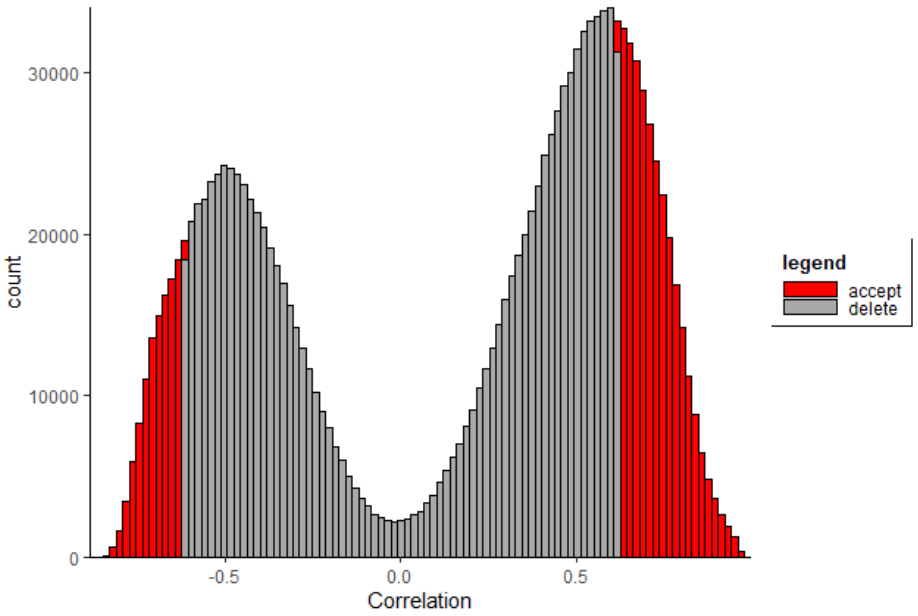


- x-axis represents the Person correlation coefficient between the expression profiles of all pairs of genes
- y-axis represents the frequency
- red bars are the selected highly correlated pairs
- grey bars are the deleted poorly correlated pairs

Correlation between two variables

```
computeCorrelation <- function(data,type,method){

  res <- rcorr(t(data), type = type)

  rho <- res$r
  pval <- res$P

  pval_adj <- p.adjust(pval, method = method)

  pval_adj <- matrix(pval_adj, ncol = ncol(pval),
                     dimnames = list(row = row.names(rho), col = colnames(rho)))

  ut <- upper.tri(rho)

  df <- data.frame(source = rownames(rho)[row(rho)[ut]],
                   target = rownames(rho)[col(rho)[ut]],
                   correlation  = rho[ut],
                   pval = pval[ut],
                   pval_adj = pval_adj[ut])

  return(df)

}
```

Pearson correlation coefficient distribution



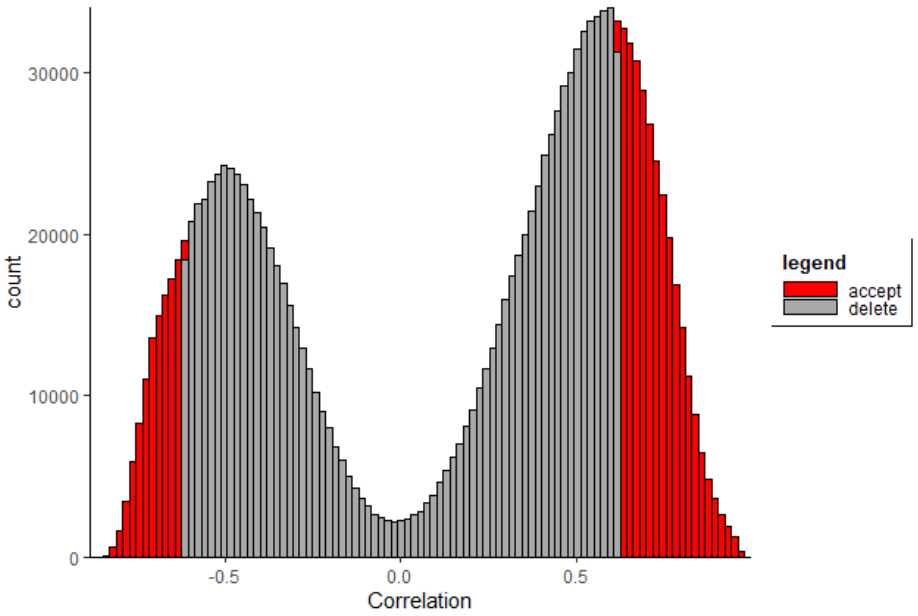
- x-axis represents the Person correlation coefficient between the expression profiles of all pairs of genes
- y-axis represents the frequency
- red bars are the selected highly correlated pairs
- grey bars are the deleted poorly correlated pairs

Correlation between two variables

```
computeCorrelation <- function(data,type,method){

  res <- rcorr(t(data), type = type)

  rho <- res$r
  pval <- res$P

  pval_adj <- p.adjust(pval, method = method)

  pval_adj <- matrix(pval_adj, ncol = ncol(pval),
                     dimnames = list(row = row.names(rho), col = colnames(rho)))

  ut <- upper.tri(rho)

  df <- data.frame(source = rownames(rho)[row(rho)[ut]],
                   target = rownames(rho)[col(rho)[ut]],
                   correlation  = rho[ut],
                   pval = pval[ut],
                   pval_adj = pval_adj[ut])

  return(df)

}
```

⚠ **Caveat:** the user can choose even other correlation methods (e.g. type = Spearman)

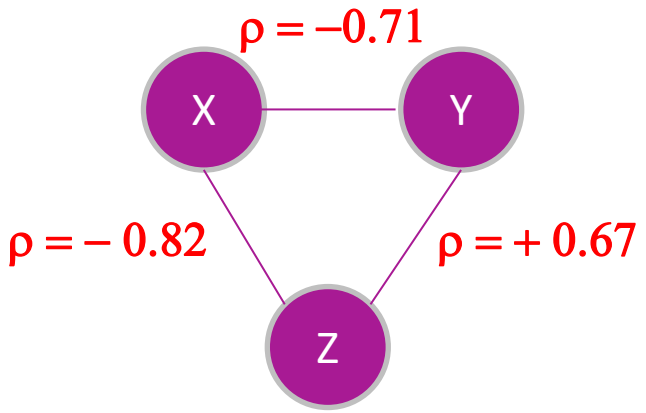
Pearson correlation coefficient distribution



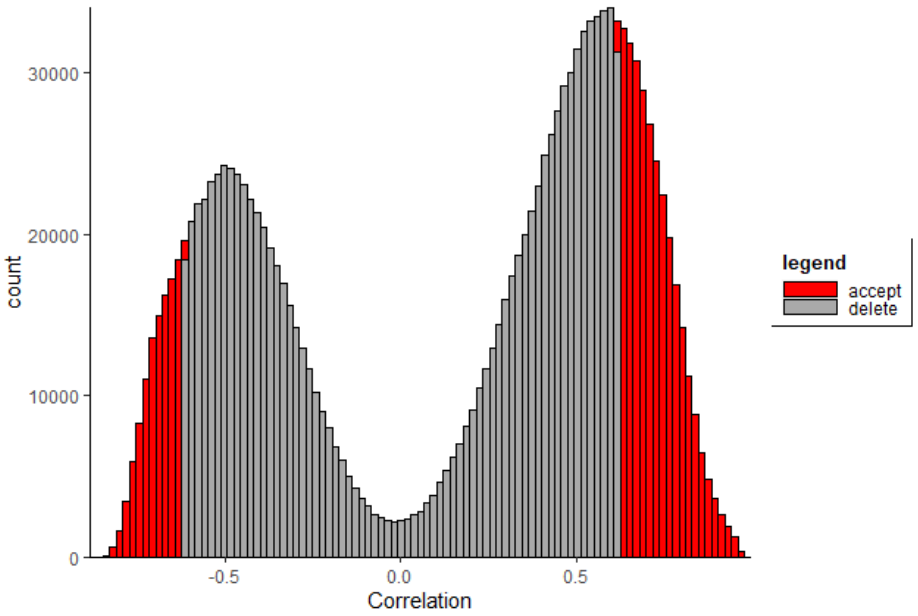
- x-axis represents the Person correlation coefficient between the expression profiles of all pairs of genes
- y-axis represents the frequency
- red bars are the selected highly correlated pairs
- grey bars are the deleted poorly correlated pairs

Correlation network

Nodes are genes (including RNAs and miRNAs if available) and a link occurs between two nodes if the absolute value of the Pearson correlation between their expression profiles is greater than the selected threshold (e.g., 80th percentile → 0.63)



Pearson correlation coefficient distribution

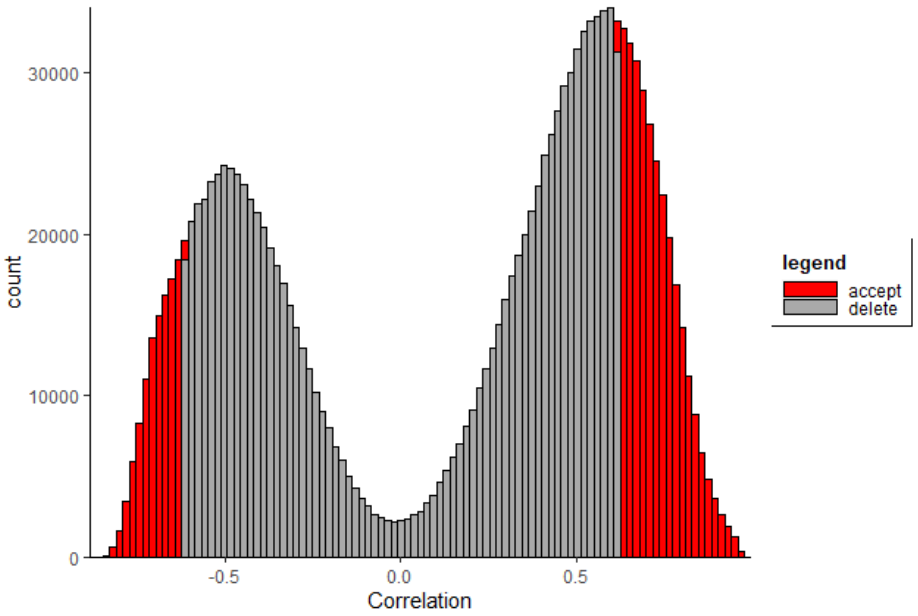


- x-axis represents the Person correlation coefficient between the expression profiles of all pairs of genes
- y-axis represents the frequency
- red bars are the selected highly correlated pairs
- grey bars are the deleted poorly correlated pairs

Correlation network

```
buildCorrelationNetwork <- function(network,threshold_corr,threshold_pval_adj,
                                    output_file_CorrelationNetwork,
                                    output_file_CorrelationNetwork_R,save=F){

  ind <- which((abs(network$correlation) >= threshold_corr) &
                 (network$pval_adj <= threshold_pval_adj))

  network <- network[ind,]

  if(save){

    write.table(network, output_file_CorrelationNetwork, sep="\t",
                row.names = F, col.names = T, quote = F)

    save(network, file = output_file_CorrelationNetwork_R)

  }

  return(network)

}
```

Pearson correlation coefficient distribution

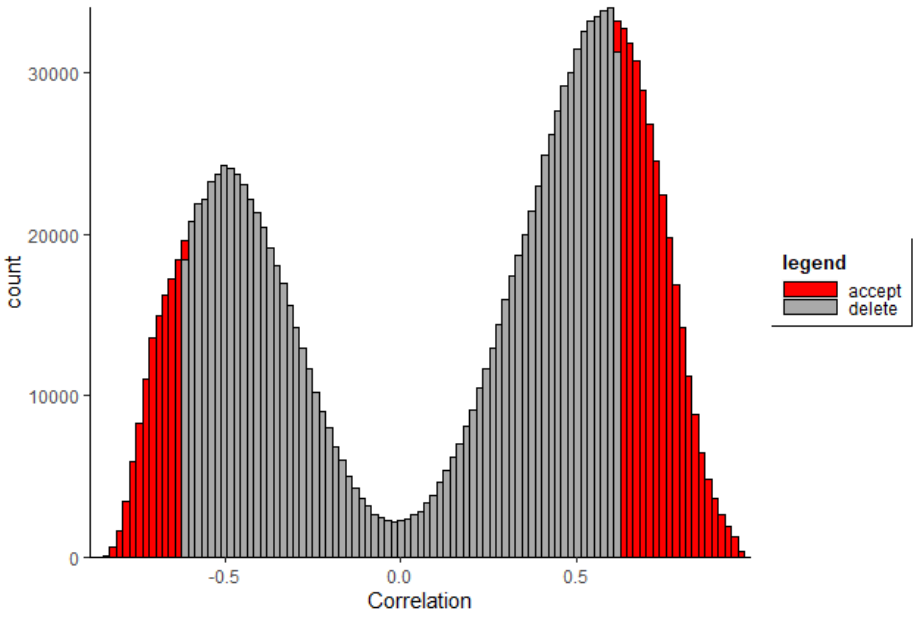


- x-axis represents the Person correlation coefficient between the expression profiles of all pairs of genes
- y-axis represents the frequency
- red bars are the selected highly correlated pairs
- grey bars are the deleted poorly correlated pairs

Histogram of Pearson correlation

```
getHistogram <- function(x,threshold,title,xlabel){

  df <- data.frame(variable = x)

  df$legend <- ifelse( (abs(x) <= threshold), "delete", "accept")

  w <- (max(x) - min(x)) / 100

  p = ggplot(df, aes(variable, fill = legend)) + geom_histogram
(binwidth = w, colour='black') +
    scale_x_continuous(expand = c(0, 0)) + scale_y_continuous(expand
= c(0, 0)) +
    scale_fill_manual(values = c("delete" = "darkgrey", "accept" =
"red")) +
    theme(panel.grid.major = element_blank(), panel.grid.minor =
element_blank(),
          panel.background = element_blank(), axis.line = element_li
ne(colour = "black"),
          plot.title = element_text(hjust = 0.5, face = "bold"),
          legend.title = element_text(colour = "black", size=10,
face="bold"),
          legend.key.height = unit(0.2, "cm"),legend.key.width =
unit(1, "cm"),
          legend.box.background = element_rect(colour = "black")) +
    labs(title = title, x = xlabel)

  print(p)

}
```

Pearson correlation coefficient distribution



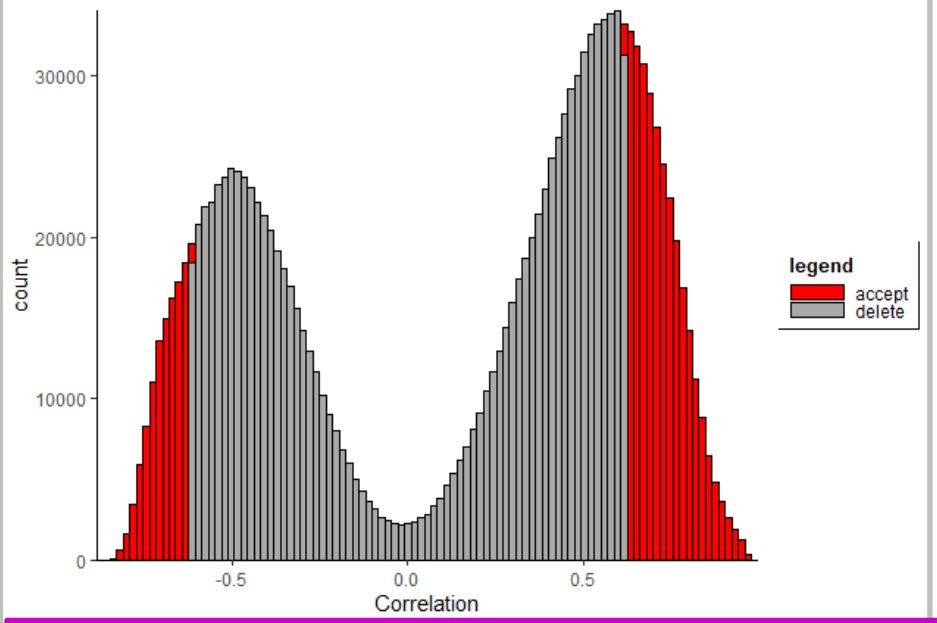
- x-axis represents the Person correlation coefficient between the expression profiles of all pairs of genes
- y-axis represents the frequency
- red
- grey
pairs

Histogram of Pearson correlation

```
getHistogram <- function(x,threshold,title,xlabel){

  df <- data.frame(variable = x)

  df$legend <- ifelse( (abs(x) <= threshold), "delete", "accept")

  w <- (max(x) - min(x)) / 100

  p = ggplot(df, aes(variable, fill = legend)) + geom_histogram
(binwidth = w, colour='black') +
    scale_x_continuous(expand = c(0, 0)) + scale_y_continuous(expand
= c(0, 0)) +
    scale_fill_manual(values = c("delete" = "darkgrey", "accept" =
"red")) +
    theme(panel.grid.major = element_blank(), panel.grid.minor =
element_blank(),
          panel.background = element_blank(), axis.line = element_li
ne(colour = "black"),
          plot.title = element_text(hjust = 0.5, face = "bold"),
          legend.title = element_text(colour = "black", size=10,
face="bold"),
          legend.key.height = unit(0.2, "cm"),legend.key.width =
unit(1, "cm"),
          legend.box.background = element_rect(colour = "black")) +
    labs(title = title, x = xlabel)

  print(p)
```

⚠ **Caveat:** Only the highly correlated pairs (red bars) according to the selected threshold will be used to build the correlation network

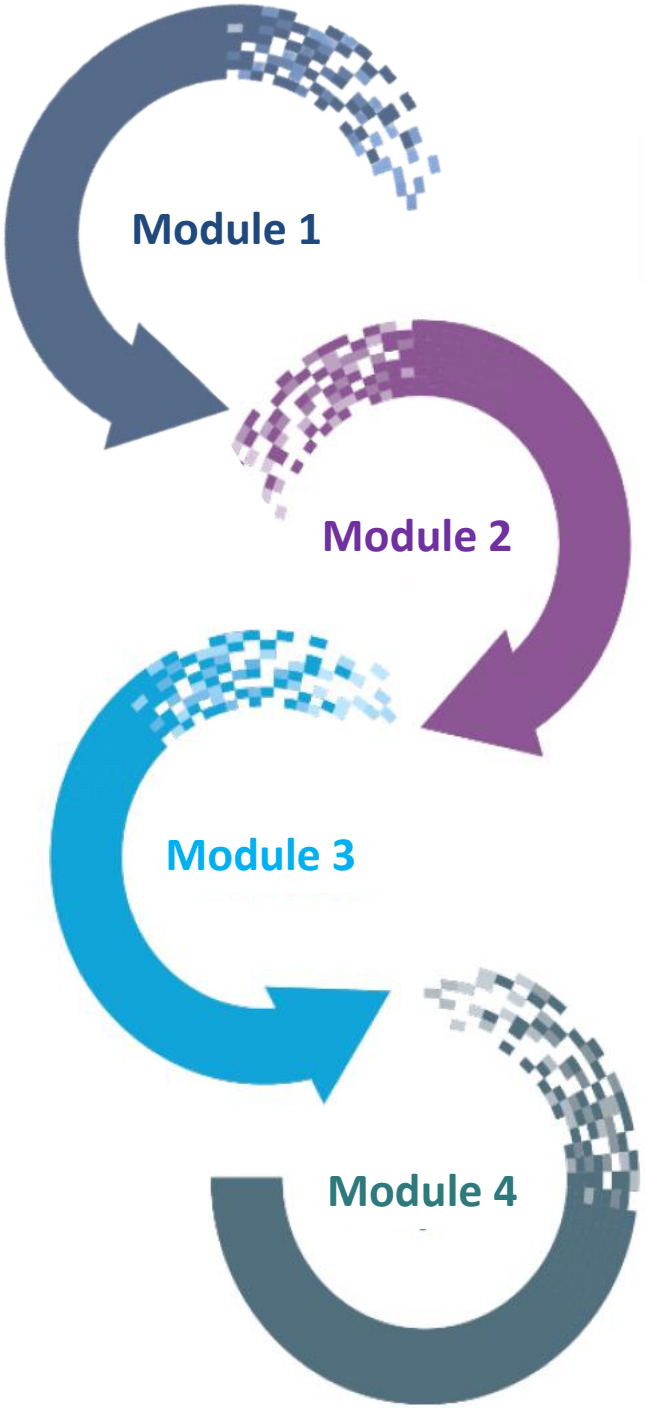
Pearson correlation coefficient distribution



Histogram of Pearson correlation

```
getHistogram <- function(x,threshold,title,xlabel){

  df <- data.frame(variable = x)

  df$legend <- ifelse( (abs(x) <= threshold), "delete", "accept")

  w <- (max(x) - min(x)) / 100

  p = ggplot(df, aes(variable, fill = legend)) + geom_histogram
(binwidth = w, colour='black') +
    scale_x_continuous(expand = c(0, 0)) + scale_y_continuous(expand
= c(0, 0)) +
    scale_fill_manual(values = c("delete" = "darkgrey", "accept" =
"red")) +
    theme(panel.grid.major = element_blank(), panel.grid.minor =
element_blank(),
          panel.background = element_blank(), axis.line = element_li
ne(colour = "black"),
          plot.title = element_text(hjust = 0.5, face = "bold"),
```

Correlation threshold should reflect a right balance between the number of edges and the number of connected components of the network: the number of **edges** should be **as small as possible** in order to have a manageable network (**high** threshold) and the number of **connected components** should be **as small as possible** in order to preserve the integrity of the network (**small** threshold).



Exploratory Data Analysis

- Importing data
- Pre-processing phase
- Filtering phase

Network Analysis

- Building the correlation network
- Checking network integrity
- Finding communities in the network

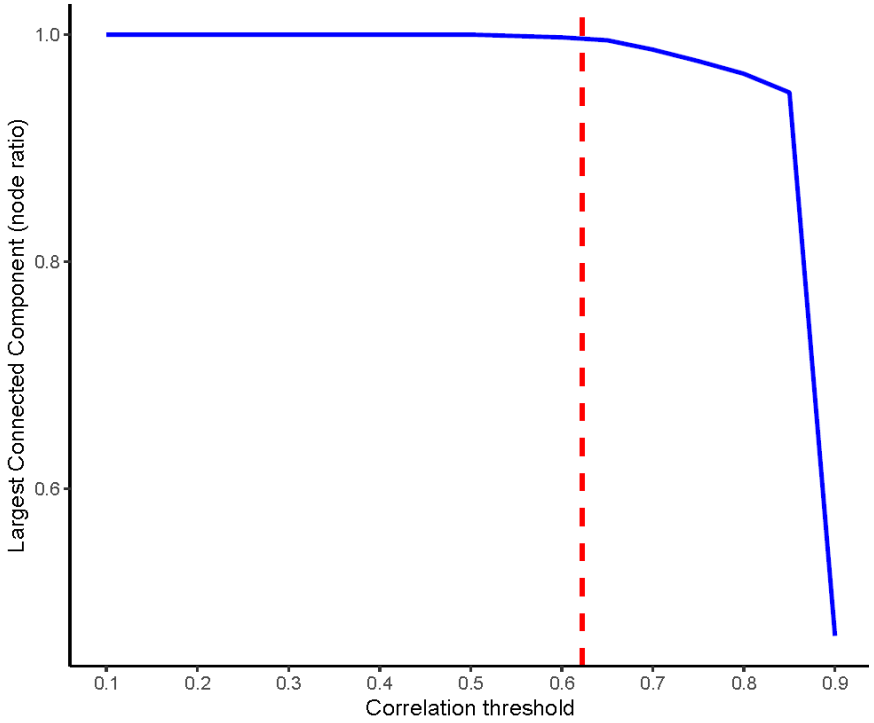
Switch Mining

- Building the heat cartography map
- Identification of switch genes

Resilience Analysis

- Evaluation of network resilience

Network integrity plot



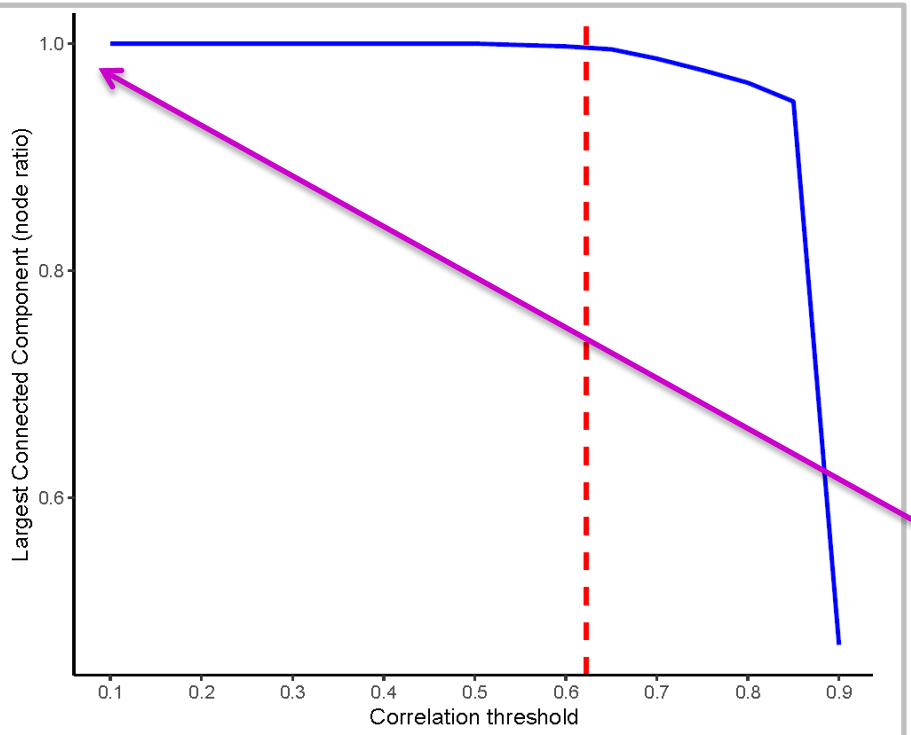
- x-axis represents the Pearson correlation threshold varying in the chosen range
- y-axis represents the fraction of nodes populating the largest connected component
- dashed red line correspond to the selected threshold

Network integrity

- This step evaluates the effect on the **network connectivity** of varying the Pearson correlation threshold
- It is **optional**: set "T" on main.R

```
network <- NetworkAnalysis(data, checkNetIntegrity = T,
screePlot = T)
```

Network integrity plot



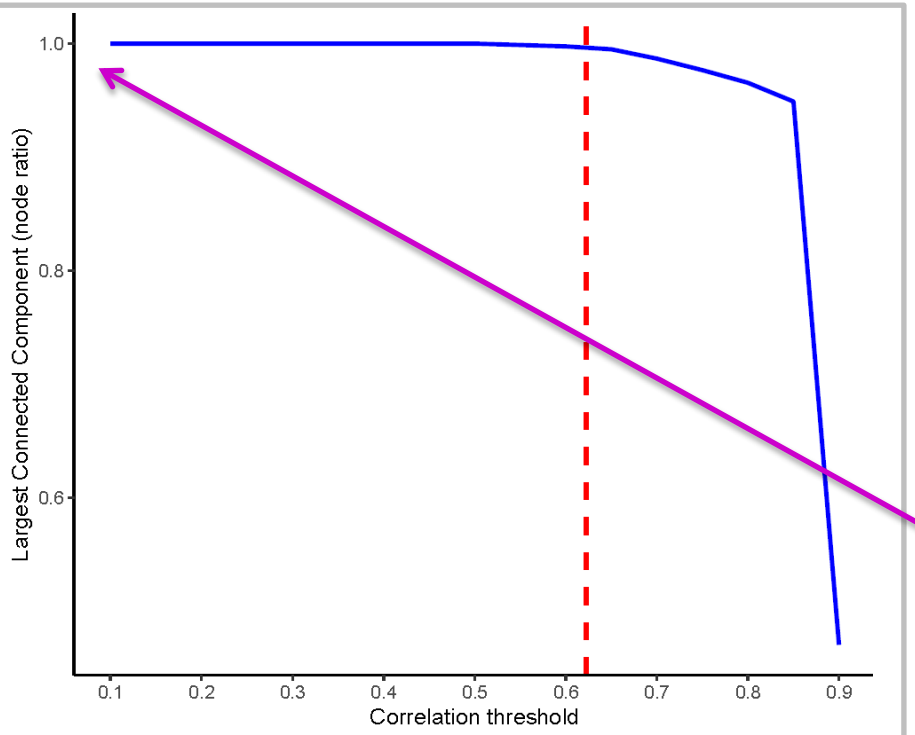
- x-axis represents the Pearson correlation threshold varying in the chosen range
- y-axis represents the fraction of nodes populating the largest connected component
- dashed red line correspond to the selected threshold

Network integrity

- This step evaluates the effect on the **network connectivity** of varying the Pearson correlation threshold
- It is **optional**: set "T" on main.R

y=1 means that all nodes fall in the largest component and thus the network is **fully connected**; otherwise more components exist

Network integrity plot



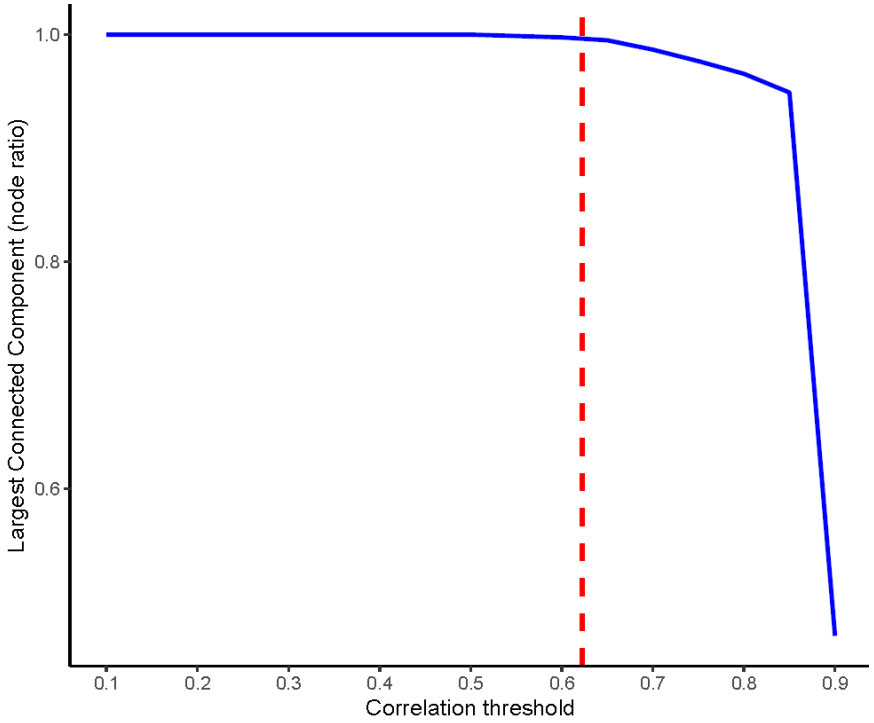
Network integrity

- This step evaluates the effect on the **network connectivity** of varying the Pearson correlation threshold
- It is **optional**: set "T" on main.R

y=1 means that all nodes fall in the largest component and thus the network is **fully connected**; otherwise more components exist

A reasonable choice for the correlation threshold should be the largest one (corresponding to the smallest number of edges) for which the fraction of nodes of the largest connected component is equal to 1. Here, we could have increased the threshold up to 0.7.

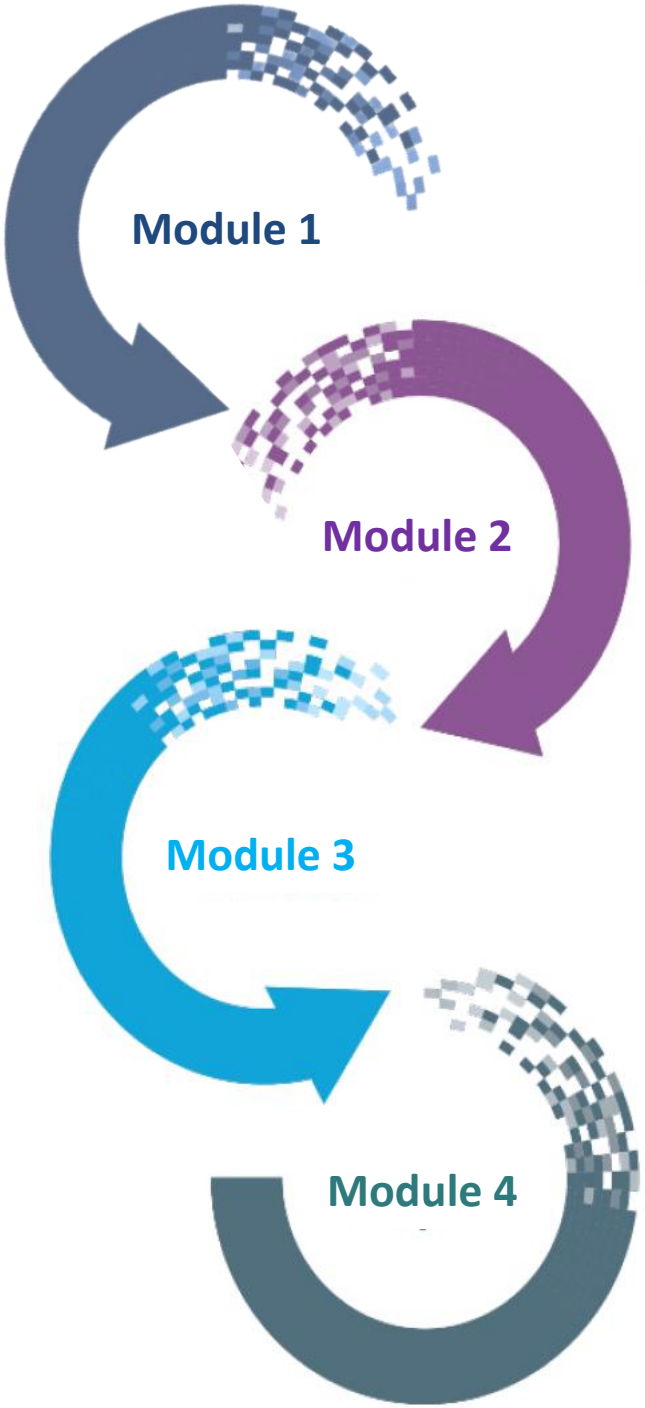
Network integrity plot



- x-axis represents the Pearson correlation threshold varying in the chosen range
- y-axis represents the fraction of nodes populating the largest component
- dashed red line correspond to the selected threshold

Network integrity plot

```
getNetworkIntegrityPlot <- function(df,min_rho,max_rho
,threshold_corr,output_file){

  p <- ggplot(df, aes(x = threshold_corr, y = frac_node_LCC))
+
    geom_line(color = "blue", size = 1) + scale_x_continuous
(breaks=seq(min_rho,max_rho,0.1)) +
    theme(panel.grid.major = element_blank(), panel.grid.minor
= element_blank(),
          panel.background = element_blank(), #axis.title =
element_text(face = "bold"),
          axis.line = element_line(colour = "black")) +
    labs(x = "Correlation threshold", y = "Largest Connected
Component (node ratio)") +
    geom_vline(xintercept = threshold_corr, linetype =
"dashed", color = "red", size = 1)

  print(p)

  savePDF(p,output_file)

}
```



Exploratory Data Analysis

- Importing data
- Pre-processing phase
- Filtering phase

Network Analysis

- Building the correlation network
- Checking network integrity
- Finding communities in the network

Switch Mining

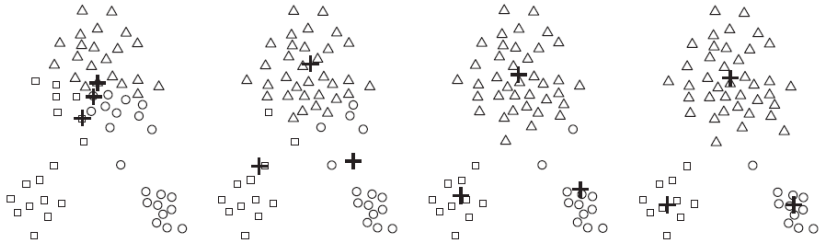
- Building the heat cartography map
- Identification of switch genes

Resilience Analysis

- Evaluation of network resilience

K-means

1: Select K points as initial centroids.
2: **repeat**
3: Form K clusters by assigning each point to its closest centroid.
4: Recompute the centroid of each cluster.
5: **until** Centroids do not change.



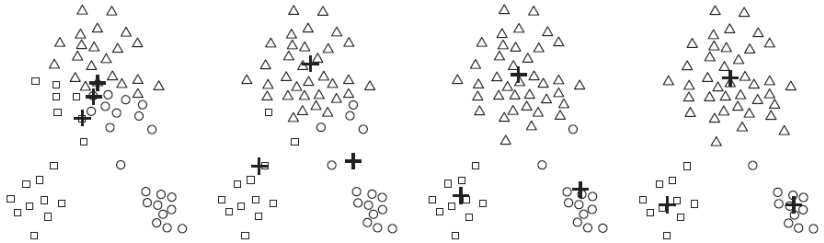
(a) Iteration 1. (b) Iteration 2. (c) Iteration 3. (d) Iteration 4.

K-means algorithm

- This step searches for communities in the correlation network by a **k-means algorithm**
- **k-means** is a clustering algorithm whose aim is to partition a set of *n* objects in *N* groups (clusters) so that each object belongs to the cluster with the nearest centroid

K-means

1: Select K points as initial centroids.
2: **repeat**
3: Form K clusters by assigning each point to its closest centroid.
4: Recompute the centroid of each cluster.
5: **until** Centroids do not change.



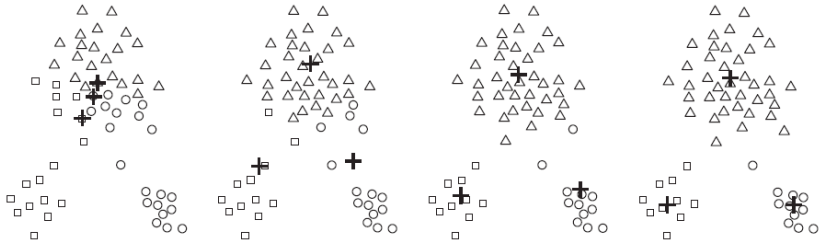
(a) Iteration 1. (b) Iteration 2. (c) Iteration 3. (d) Iteration 4.

K-means algorithm

- It requires to know the **number of clusters** in advance
- The k-means algorithm performs **iterations** until the minimum of the Sum of the Squared Error (SSE) function is reached
- SWIMmer repeats the clustering many times (**replicates**), each with a new set of initial cluster centroid positions, randomly chosen, and selects the clusters configuration corresponding the minimum of SSE among all replicates

K-means

1: Select K points as initial centroids.
2: **repeat**
3: Form K clusters by assigning each point to its closest centroid.
4: Recompute the centroid of each cluster.
5: **until** Centroids do not change.



(a) Iteration 1. (b) Iteration 2. (c) Iteration 3. (d) Iteration 4.

K-means algorithm

- The **Sum of the Squared Error** function is

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(c_i, x)$$

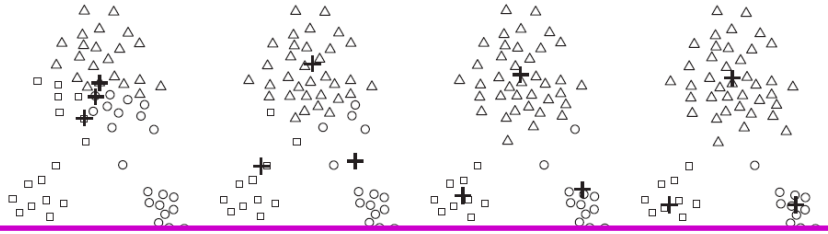
- where *dist* is the Euclidean distance, K is the number of the clusters, C_i is the i^{th} cluster, x is a node in the i^{th} cluster, c_i is the centroid of the i^{th} cluster given by:

$$c_i = \frac{1}{m_i} \sum_{x \in C_i} x_i$$

- where m_i is the number of nodes in the i^{th} cluster. There are as many centroids as the number of the clusters.

K-means

1: Select K points as initial centroids.
2: **repeat**
3: Form K clusters by assigning each point to its closest centroid.
4: Recompute the centroid of each cluster.
5: **until** Centroids do not change.



K-means algorithm

- The **Sum of the Squared Error** function is

$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(c_i, x)$$

- where *dist* is the Euclidean distance, K is the number of the clusters, C_i is the i^{th} cluster, x is a node in the i^{th} cluster, c_i is the centroid of the i^{th} cluster given by:

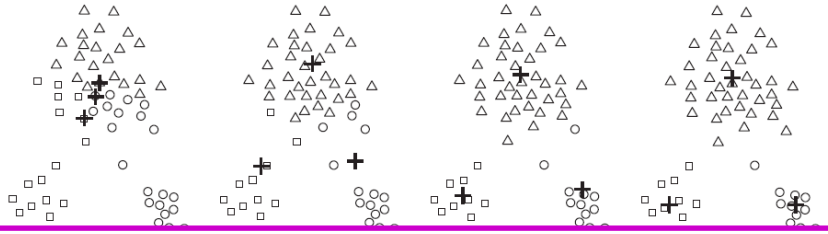
The choice of the Euclidean distance is intended to emphasize the differences between correlated and anti-correlated genes.

$$dist^2(A,B) = \left(x_1^A - x_1^B\right)^2 + \left(x_2^A - x_2^B\right)^2 + \cdots + \left(x_N^A - x_N^B\right)^2$$

$x_1^A = a_{A,1}$ where $a_{N,N}$ is the weighted adjacency matrix and the weights are the Pearson correlation coefficients

K-means

1: Select K points as initial centroids.
2: **repeat**
3: Form K clusters by assigning each point to its closest centroid.
4: Recompute the centroid of each cluster.
5: **until** Centroids do not change.



K-means algorithm

- The **Sum of the Squared Error** function is

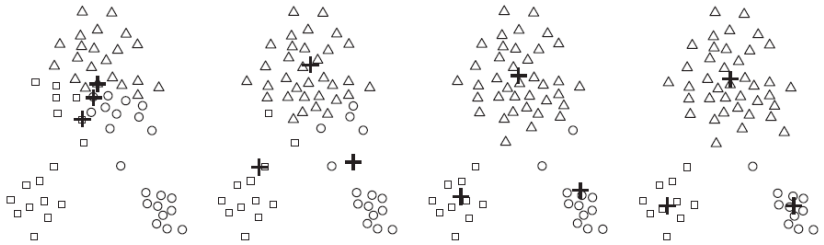
$$SSE = \sum_{i=1}^{K} \sum_{x \in C_i} dist^2(c_i, x)$$

- where *dist* is the Euclidean distance, K is the number of the clusters, C_i is the i^{th} cluster, x is a node in the i^{th} cluster, c_i is the centroid of the i^{th} cluster given by:

Thus, two nodes A, B are close in the network ($dist = 0$) if they are highly correlated ($\rho = 1$) and they are far apart in the network ($dist = dist_{max}$) if they are highly anti-correlated ($\rho = -1$).

K-means

1: Select K points as initial centroids.
2: **repeat**
3: Form K clusters by assigning each point to its closest centroid.
4: Recompute the centroid of each cluster.
5: **until** Centroids do not change.



(a) Iteration 1. (b) Iteration 2. (c) Iteration 3. (d) Iteration 4.

K-means algorithm

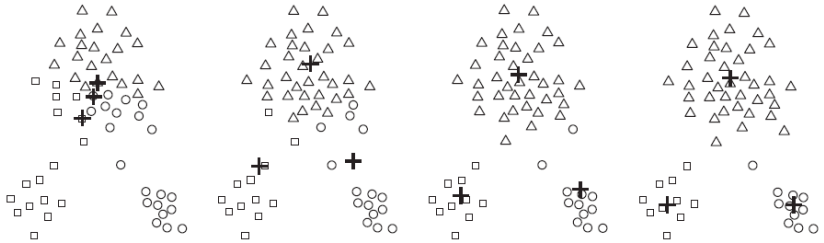
weighted adjacency matrix

parameters

```
getClustering <- function(w_adj,num_clusters,iter_max,num_repeats,output_file_idx){

  ########################################

  model <- kmeans(w_adj, centers = num_clusters, iter.max = iter_max, nstart = num_repeats)

  WSS <- model$withinss        # Within-cluster Sum of Squares (sum of squared distances from each centroid)
  TWSS <- model$tot.withinss   # Total Within-cluster Sum of Squares (SSE = TWSS = sum(WSS))
  TSS <- model$totss           # the Total Sum of Squares (sum of squared distances from the network centroid)
  BSS <- model$betweenss       # the Between Sum of Squares (BSS = TSS - TWSS)
  size <- model$size
  idx <- data.frame(cluster = as.factor(model$cluster))

  # The smaller WSS, the more compact the cluster, the higher the score
  cluster_score <- (1 - WSS/TSS) * 100
  cluster_score <- signif(cluster_score,3)

  # The smaller TWSS (or the larger BSS), the better the k-means partition, the higher the score
  partition_score <- BSS / TSS * 100  # (1 - TWSS/TSS) * 100
  partition_score <- signif(partition_score,3)
  ########################################

  write.table(idx, output_file_idx, sep="\t", row.names = T, col.names = NA, quote = F)

  res <- list(size = size, idx = idx,
              cluster_score = cluster_score,
              partition_score = partition_score)

  return(res)

}
```

K-means

1: Select K points as initial centroids.
2: **repeat**
3: Form K clusters by assigning each point to its closest centroid.
4: Recompute the centroid of each cluster.
5: **until** Centroids do not change.



(a) Iteration 1. (b) Iteration 2. (c) Iteration 3. (d) Iteration 4.

K-means parameters

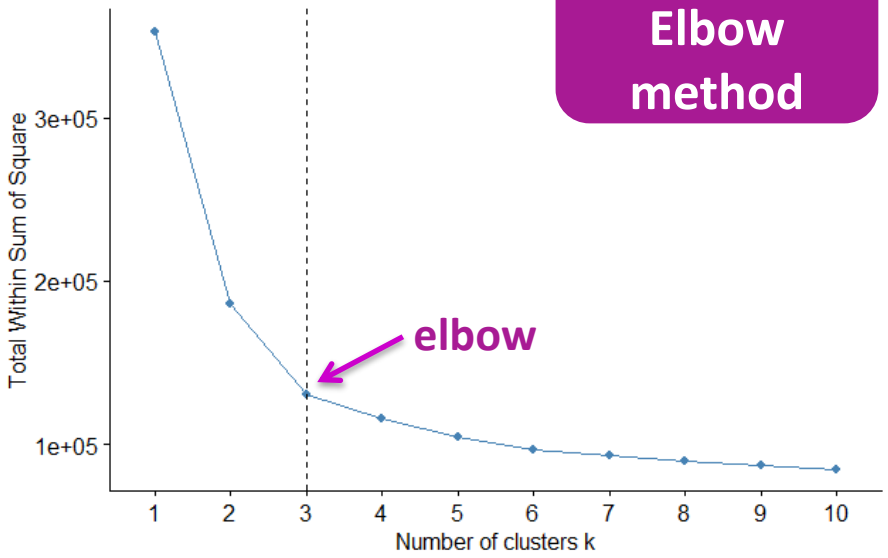
- the **number of clusters**
- the maximum number of **iterations** allowed for each replicate of k-means
- the number of **replicates** for a given number of clusters

⚠ **Caveat:** maximum number of **iterations** should be large enough to guarantee convergence

⚠ **Caveat:** more **replicates** get you more confident to be far from a local minimum

Scree plot

Scree plot: Elbow method



- x-axis refers to the number of clusters
- y-axis refers to the value of the error function corresponding to the best clusters configuration among all replicates for that number of clusters

Scree plot computation

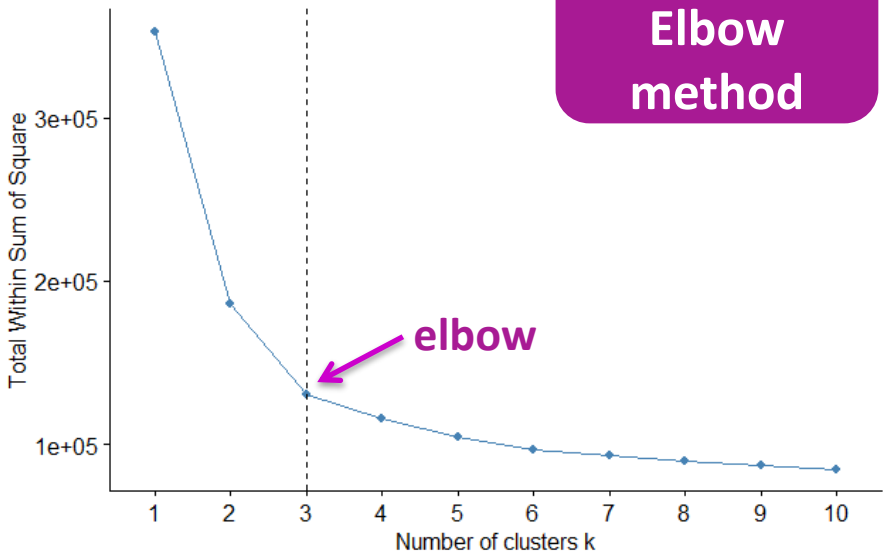
- Scree plot will be produced to suggest the best choice for the number of **clusters**
- It is **optional**: set "T" on main.R

```
network <- NetworkAnalysis(data, checkNetIntegrity = T,
screePlot = T)
```

- It shows the value of the error function (**SSE**) for each number of clusters
- a good choice for the optimal number of clusters is in correspondence of the **elbow**

Scree plot

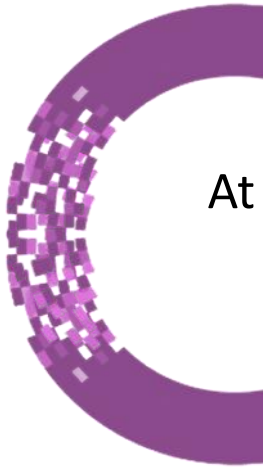
Scree plot: Elbow method



- x-axis refers to the number of clusters
- y-axis refers to the value of the error function corresponding to the best clusters configuration among all replicates for that number of clusters

Scree plot computation

```
getScreePlot <- function(w_adj){

  ########################################
  # input parameters

  num_clusters <- input_parameter$num_clusters
  iter_max <- input_parameter$iter_max
  num_repeats <- input_parameter$num_repeats
  ########################################

  # Elbow method
  p <- fviz_nbclust(w_adj, kmeans, method = "wss", iter
.max = iter_max, nstart = num_repeats) +
    geom_vline(xintercept = num_clusters, linetype = 2) +
# add line for better visualisation
    labs(subtitle = "Elbow method") # add subtitle

  print(p)

}
```



At the end of module 2, you will obtain the correlation network