Hipathia Package

***Marta R. Hidalgo*[*1], *Francisco Salavert*[2], *Alicia Amadoz*[3], *Çankut Cubuk*[4], *José Carbonell-Caballero*[5], and *Joaquín Dopazo*[4,6,7]**

[1]Unidad de Bioinformática y Bioestadística, Centro de Investigación Príncipe Felipe (CIPF), Valencia, 46012, Spain
[2]BioBam Bioinformatics S.L., Valencia, 46012, Spain
[3]Department of Bioinformatics, Igenomix S.A., Valencia, 46980, Spain
[4]Clinical Bioinformatics Area, Fundación Progreso y Salud (FPS), Hospital Virgen del Rocio, Sevilla, 41013, Spain
[5]Chromatin and Gene expression Lab, Gene Regulation, Stem Cells and Cancer Program, Centre de Regulació Genòmica (CRG), The Barcelona Institute of Science and Technology, PRBB, Barcelona, 08003, Spain
[6]Functional Genomics Node (INB), FPS, Hospital Virgen del Rocio, Sevilla, 41013, Spain.
[7]Bioinformatics in Rare Diseases (BiER), Centro de Investigación Biomédica en Red de Enfermedades Raras (CIBERER), FPS, Hospital Virgen del Rocio, Sevilla, 41013, Spain

*marta.hidalgo@outlook.es

2018-05-01

Abstract

Hipathia is a method for the computation of signal transduction along signaling pathways from transcriptomic data. The method is based on an iterative algorithm which is able to compute the signal intensity passing through the nodes of a network by taking into account the level of expression of each gene and the intensity of the signal arriving to it. It also provides a new approach to functional analysis allowing to compute the signal arriving to the functions annotated to each pathway.

Package

hipathia 1.0.0

Contents

1 Introduction

Hipathia package implements the Canonical Circuit Activity Analysis method for the quantification of the signaling pathways activity presented in Hidalgo et al. This method has been implemented in the webtool http://hipathia.babelomics.org, allowing the user to compare signal propagation in an experiment, and train and use a predictor based on the activation of the canonical circuits or subpathways. The package *hipathia* has been conceived as a functional tool for R users which allows more control on the analysis pipeline than the web implementation does.

This document will introduce you to the *hipathia* package and how to use it to analyze your data.

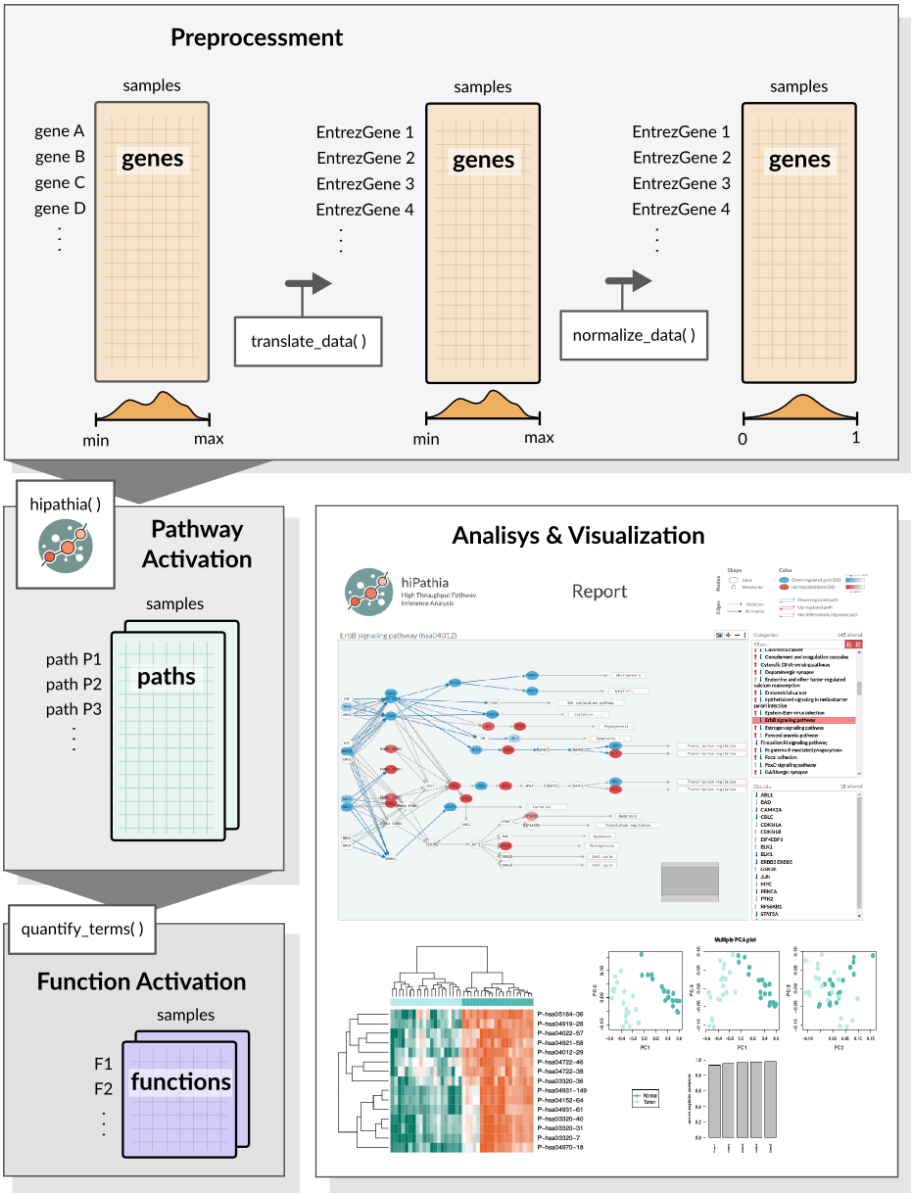

Figure 1: Visual representation of the package functionalities

2 Previous considerations

Hipathia is a method for the computation of signal transduction along signaling pathways taking as input transcriptomics data. The method is independent on the pathways database, it only needs information about the topology of the graph and the genes included in each node.

However, due to computational cost, *hipathia* needs to preprocess the graphs to be fully efficient. In the current implementation we have developed a module which has preprocessed 145 KEGG pathway KGML files, which are ready to be analyzed.

Further versions of the package will allow the user to preprocess their own graph pathways to be analyzed with *hipathia*.

2.1 Instalation

In order to install the *hipathia* package, type on your R console

```
## try http:// if https:// URLs are not supported
source("https://bioconductor.org/biocLite.R")
biocLite("hipathia")
```

2.2 Example data

In order to illustrate the *hipathia* package functionalities an example dataset has been prepared. Data has been downloaded from The Cancer Genome Atlas data repository, from the BRCA-US project, release 20. 20 tumor and 20 normal samples of RNA-Seq data have been randomly selected and normalized.

Specifically, raw data has been corrected for batch effect using the `ComBat` function from package *sva*, then corrected for RNA composition bias applying TMM normalization from package *edgeR*, and finally log-transformed.

```
library(hipathia)
data("brca")
brca
## class: SummarizedExperiment
## dim: 3187 40
## metadata(0):
## assays(1): raw
## rownames(3187): 2 8647 ... 3925 219699
## rowData names(0):
## colnames(40): TCGA.BH.A1FM.11B.23R.A13Q.07
##   TCGA.E2.A1LB.11A.22R.A144.07 ... TCGA.A2.A0CT.01A.31R.A056.07
##   TCGA.BH.A18U.01A.21R.A12D.07
## colData names(1): group
```

The dataset `brca` is a *SummarizedExperiment* object, including the gene expression of the 40 samples in the assay `raw`, and the information about whether each sample comes from *Tumor* or *Normal* tissues in the `group` columns of the `colData` dataFrame.

```
hhead(assay(brca), 4)
##      TCGA.BH.A1FM.11B.23R.A13Q.07 TCGA.E2.A1LB.11A.22R.A144.07
## 2                       10.5317320                     9.732938
## 8647                    -3.3266788                    -3.457515
## 5244                    -0.3600828                    -1.139309
## 1244                     2.2876961                     1.724625
##      TCGA.BH.A208.11A.51R.A157.07 TCGA.BH.A18K.11A.13R.A12D.07
## 2                       9.7958036                    10.868669
## 8647                    -2.5261155                    -3.584934
## 5244                    -0.7368491                    -1.257797
## 1244                     1.0217356                     1.467979
```

```
colData(brca)
## DataFrame with 40 rows and 1 column
##                                    group
##                                <character>
## TCGA.BH.A1FM.11B.23R.A13Q.07      Normal
## TCGA.E2.A1LB.11A.22R.A144.07      Normal
## TCGA.BH.A208.11A.51R.A157.07      Normal
## TCGA.BH.A18K.11A.13R.A12D.07      Normal
## TCGA.E9.A1RC.11A.33R.A157.07      Normal
## ...                                  ...
## TCGA.AO.A12A.01A.21R.A115.07       Tumor
## TCGA.AR.A0TR.01A.11R.A084.07       Tumor
## TCGA.A8.A07E.01A.11R.A034.07       Tumor
## TCGA.A2.A0CT.01A.31R.A056.07       Tumor
## TCGA.BH.A18U.01A.21R.A12D.07       Tumor
```

2.3 Accepted objects

Hipathia has been designed to work with matrices encapsulated as *SummarizedExperiment* objects, in which also the experimental design has been included. However, it is also possible to work in *hipathia* with matrix objects, as long as the experimental design is provided when needed.

Imagine we have the expression data stored in a matrix object called `brca_data` and the experimental design stored in a data frame with one column called `brca_design`. Then, in order to summarize this data in a `SummarizedExperiment` object we should only run:

```
brca <- SummarizedExperiment(assays=SimpleList(raw=brca_data),
                             colData=brca_design)
```

Note that the data frame object provided as `colData` parameter should be ordered as the columns in the matrix provided as assay. For further information on this kind of objects please refer to *SummarizedExperiment*.

When executing a function which needs as input parameter the experimental design (such as the Wilcoxon test in `do_wilcoxon`, or `heatmap_plot`), parameter `group` may take two different objects. In case parameter `data` is a matrix, `group` should be a vector giving the class to

which each sample belongs, in the same order than the data matrix. In case parameter `data` is a `SummarizedExperiment`, `group` may be either a vector as above, or the name of the column in the `colData` `dataFrame` of the `SummarizedExperiment` storing this information.

In general, functions accepting both SummarizedExperiment and matrix objects as input data and returning a data matrix object, will give as output the same kind of object received. That is, if we apply function `translate_data` to a SummarizedExperiment object, we will obtain a SummarizedExperiment, while applying the same function to a matrix object will result in a matrix object as output.

2.4 How to cite

Hipathia is a free open-source software implementing the result of a research work. If you use it, please support the research project by citing:

Hidalgo, M. R., Cubuk, C., Amadoz, A., Salavert, F., Carbonell-Caballero, J., & Dopazo, J. (2017). High throughput estimation of functional cell activities reveals disease mechanisms and predicts relevant clinical outcomes. Oncotarget, 8(3), 5160–5178. http://doi.org/10.18632/oncotarget.14107

3 Preprocessment

Hipathia accepts as input data a gene expression matrix. Expression may have been measured with any available sequencing technique. However, *hipathia* assumes that data has been already normalized for correcting any possible sequencing bias (which includes also batch effect correction).

3.1 Gene IDs translation

The gene expression matrix must include samples as columns and genes as rows, as shown in the `brca` dataset example. Rownames must be the Entrez IDs of the genes in the rows. In order to transform other gene IDs to Entrez IDs, function `translate_data` can be used. Accepted IDs to be transformed to Entrez IDs include:

Human

- Affy HG U133A probeset
- Affy HG U133B probeset
- Affy HG U133-PLUS_2 probeset
- Agilent SurePrint G3 GE 8x60k
- Agilent SurePrint G3 GE 8x60k v2
- Agilent Whole Genome 4x44k
- Agilent Whole Genome 4x44k v2
- CCDS
- Ensembl gene
- Ensembl transcript
- Entrez ID
- GenBank EMBL
- GenBank PID
- HGNC symbol
- RefSeq mRNA
- RefSeq mRNA PRED
- RefSeq ncRNA
- RefSeq ncRNA PRED

Mouse

- Affy Mouse 430 2
- Ensembl gene
- Gene name
- Mouse Gene 1.0

Rat

- Ensembl gene
- Gene name

The parameters needed by this function are the data matrix and the species of the experiment.

```
data(brca_data)
trans_data <- translate_data(brca_data, "hsa")
## snapshotDate(): 2018-04-23
## translated ids = 3184 (1)
## untranslated ids = 3 (0.00094)
## multihit ids = 0 (0)
```

3.2 Data scaling & normalization

Apart from the necessary bias corrections, the expression data matrix must be scaled between 0 and 1 before computing the subpaths activation values. Function `normalize_data` is designed to this purpouse.

```
exp_data <- normalize_data(trans_data)
```

```
boxplot(trans_data)
```

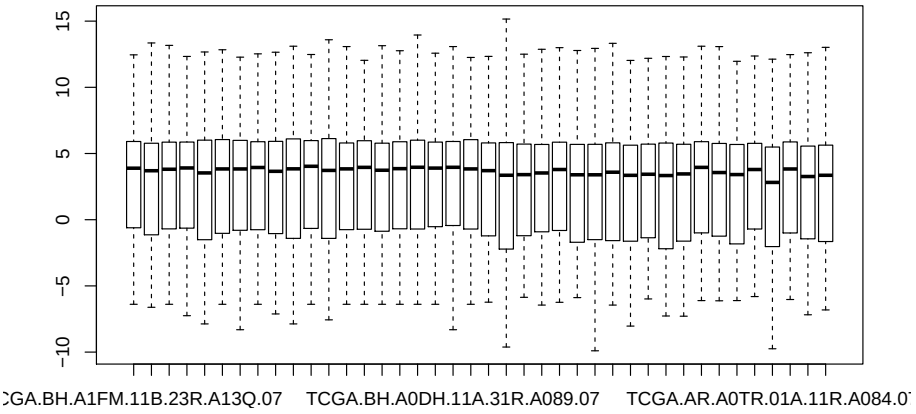

Figure 2: BRCA data before scaling

```
boxplot(exp_data)
```

Function `normalize_data` includes different parameters for normalization. If option `by_quantiles` is `TRUE`, a previous normalization by quantiles is performed.

```
exp_data <- normalize_data(trans_data, by_quantiles = TRUE)
boxplot(exp_data)
```

Other parameters of this function affect the way in which scaling to the interval [0,1] is performed. Parameter `by_gene` indicates whether to perform the scaling to [0,1] to each row of the matrix. If the option `by_gene` is set to `TRUE`, the normalization between 0 and 1 is done for each row of the matrix, meaning that the expression of each gene will have a range

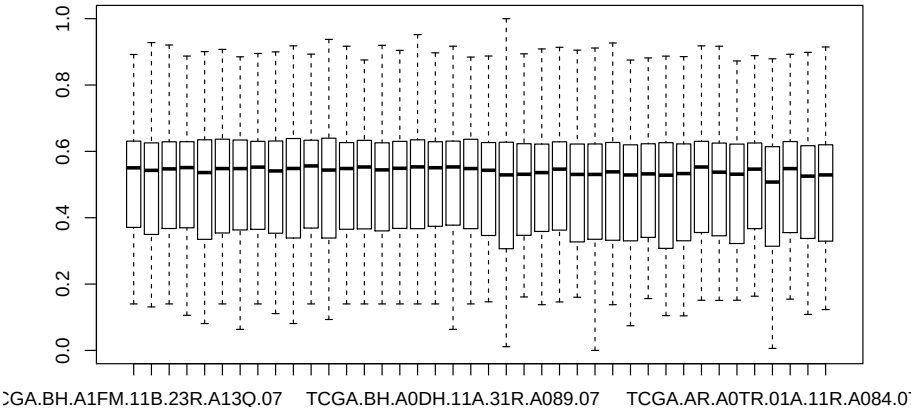

Figure 3: BRCA data after scaling

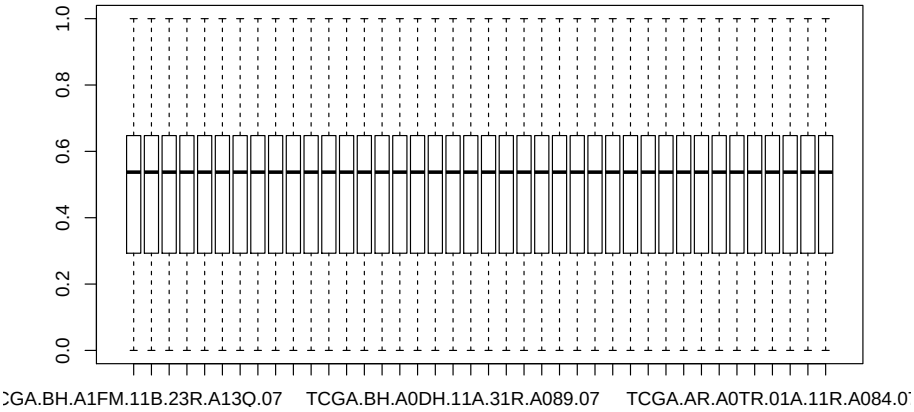

Figure 4: BRCA data after a Quantiles normalization

between 0 and 1. If it is set to `FALSE`, the normalization is done for the whole matrix, meaning that only the genes with the maximum value of the matrix will have a normalized value of 1. It is recommended to keep it set to `FALSE`, as the default value.

Parameter `percentil` indicates whether to use the percentil to compute the normalized value between 0 and 1. If it is set to `TRUE`, the function takes as a value for the position `(i,j)` of the matrix the percentil of sample `j` in the ditribution of gene `i`. If it is set to `FALSE`, the function applies a direct transformation from the original interval to [0,1]. It is recommended to keep it set to `FALSE` except for heavy-tailed distributions of the genes.

```
exp_data <- normalize_data(trans_data, percentil = TRUE)
boxplot(exp_data)
```

Parameter `truncation_percentil` gives the value of percentil `p` from which all further values are truncated to percentil `p`. Symmetrically, values beyond percentil `1-p` are also truncated to `1-p`.

```
exp_data <- normalize_data(trans_data, truncation_percentil = 0.95)
boxplot(exp_data)
```

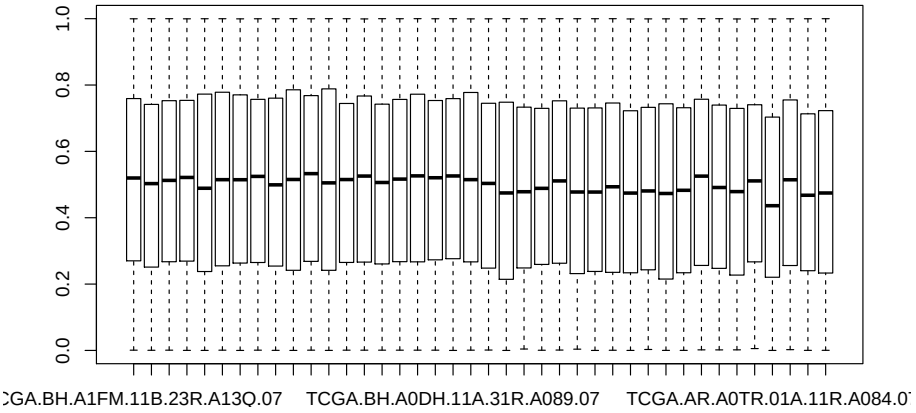

Figure 5: BRCA data after normalizing by percentil

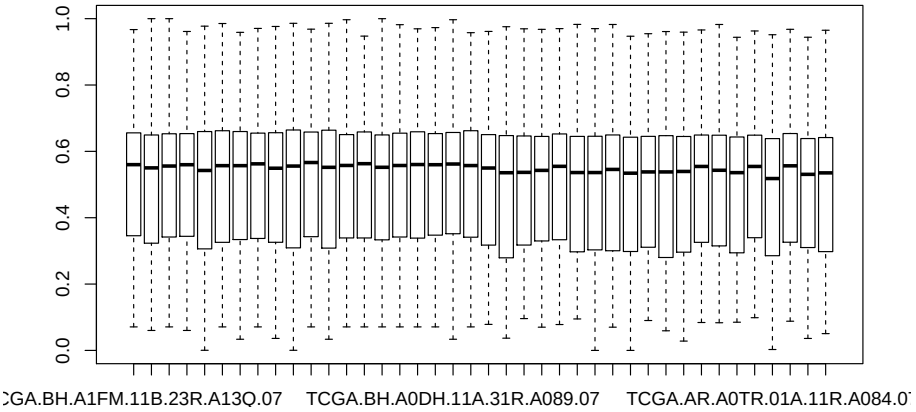

Figure 6: BRCA data after truncating by percentil 0.95

4 Pathway activation computation

Hipathia aims to compute the level of activation of each subpathway in a pathway for each of the samples from the experiment. This is done by function `hipathia`, which takes as inputs the matrix of gene expression, the pathways object and some additional parameters.

In this section we will see how to load the pathways object, how the *hipathia* method works and how to apply function `hipathia` to the computation of the values of activation of the loaded pathways.

4.1 Loading pathways

Hipathia package is currently implemented to use preprocessed KEGG pathways. The pathways have been processed and stored in a pathways object. This object includes all the information that the different functions in the package need. In order to load this object, use function `load_pathways` and select the species to be analyzed. Available species include human (*hsa*), mouse (*mmu*) and rat (*rno*).

```
pathways <- load_pathways(species = "hsa")
## Loaded 145 pathways
```

Parameter `pathways_list` allows the user to specify the pathways to be loaded. The different functions of the package will use all the pathways in the pathways object for its computations. In order to restrict the analysis to a particular set of pathways, load only the required pathways to the pathway object. By default, all pathways available for the specified species are loaded.

```
pathways_only2 <- load_pathways(species = "hsa", pathways_list = c("hsa03320",
                                                                   "hsa04014"))
## Loaded 2 pathways
```

In order to know which pathways are included in each pathways object, function `get_pathways_list` can be used.

```
length(get_pathways_list(pathways))
## [1] 145
get_pathways_list(pathways)[1:10]
##  [1] "hsa03320" "hsa03460" "hsa04010" "hsa04012" "hsa04014" "hsa04015"
##  [7] "hsa04020" "hsa04022" "hsa04024" "hsa04062"
```

```
length(get_pathways_list(pathways_only2))
## [1] 2
get_pathways_list(pathways_only2)
## [1] "hsa03320" "hsa04014"
```

4.2 Computing the signal

In order for a protein to pass the signal, there are two important factors: first, the protein must be present, and second, some other protein must activate it. Therefore, *hipathia* is a method to compute signal transduction based on two steps. First, it quantifies the presence of a particular gene as a normalized value between 0 and 1. Then, it computes the signal value passing through a node taking into account the level of expression of each gene inside the node and the intensity of the signal arriving to it. The signal value of the pathway is the signal value through the last node of the pathway.

4.2.1 Subpathways

Pathways are represented by directed graphs, which include different input and output nodes. The signal arrives to an initial node and is transmited along the pathway following the direction of the interactions up to an output node. Thus, the signal may follow many different paths along the pathway. *Hipathia* computes the intensity of this signal up to each output node of a pathway separately.

Genes in the output nodes are also called *effector proteins*, since they are the ones responsibles for performing the action the signal is seeking. We define the *effector subpathway* ending in node G as the subgraph including any node in a path leading to G. When applied to effector subpathways, *hipathia* returns the intensity of the signal arriving to the effector protein G.

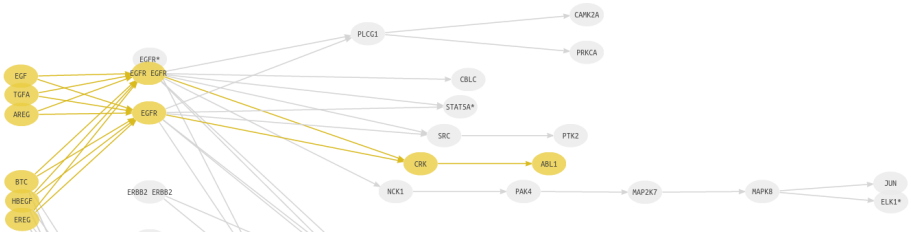

Figure 7: Effector subpathway depicted in yellow

Effector subpathways may have many different input nodes. In order to analyze in detail which of the possible paths leading to node *G* is responsible for the observed change, effector subpathays can be decomposed into several subpathways including only one input node. We define the *decomposed subpathway* from *H* to *G* as the subgraph including any node in a path from *H* to *G*.

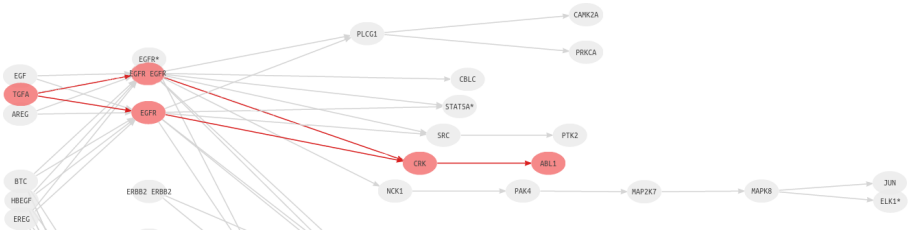

Figure 8: Decomposed subpathway depicted in red

4.2.2 Node expression

Pathways are represented by graphs and composed by nodes and relations among them. Some nodes may contain multiple genes representing different isoforms of the protein or members of the same gene familiy, among others. Since each gene has its own level of expression, the first step of the method is to summarize this information into a score representing the expression of the node as a whole.

4.2.3 Signal transduction

The computation of the signal intensity across the pathway is performed by means of an iterative algorithm beginning in the input nodes of the subpathway. In order to initialize the pathway signal we assume an incoming signal value of 1 in the input nodes of the subpathway. Then, for each node n of the network, the signal value S_n is propagated along the nodes according to the following rule

$$S_n = v_n \cdot (1 - \prod_{s_i \in A_n} (1 - s_i)) \cdot \prod_{s_j \in I_n} (1 - s_j) \quad (1)$$

where A_n is the set of signals arriving to the current node from an activation edge, I_n is the set of signals arriving to the current node from an inhibition edge, and v_n is the normalized value of expression of the current node.

4.3 Using *Hipathia* to compute the signal

Function `hipathia` computes the level of activation of the subpathways, taking as inputs the matrix of gene expression, the pathways object and some additional parameters.

Parameter `decompose` indicates whether to use effector subpathways or decomposed subpathways. Option `decompose=FALSE` uses effector subpathways while option `decompose=TRUE` uses decomposed subpathways. For further information on this, see Section 4.2.1. For further information on the method used to compute the level of signal activity in the pathways, see Section 4.2 or refer to Hidalgo et al..

```
results <- hipathia(exp_data, pathways, decompose = FALSE, verbose=FALSE)
## Added missing genes: 144 (4.33%)
```

The genes which are needed by `hipathia` to compute the signal and are not present in the provided matrix are added by the function, assigning to each sample the median of the matrix. The number and percentage of added genes is shown by the function. A high level of *added missing genes* may indicate that the results are not representative of the actual analysis.

The resulting object is a *MultiArrayExperiment* object, which includes two different *SummarizedExperiment* objects: `paths` and `nodes`.

```
results
## A MultiAssayExperiment object of 2 listed
##  experiments with user-defined names and respective classes.
##  Containing an ExperimentList class object of length 2:
##  [1] paths: SummarizedExperiment with 1868 rows and 40 columns
##  [2] nodes: SummarizedExperiment with 6752 rows and 40 columns
## Features:
##  experiments() - obtain the ExperimentList instance
##  colData() - the primary/phenotype DataFrame
##  sampleMap() - the sample availability DataFrame
##  `$`, `[`, `[[` - extract colData columns, subset, or experiment
##  *Format() - convert into a long or wide DataFrame
##  assays() - convert ExperimentList to a SimpleList of matrices
```

The `paths` object includes as assay a matrix with the level of activity of the signal in each subpathway. In order to extract the object of signal activity values from this object use function `get_paths_data`. By default, this function returns a *SummarizedExperiment* object, but it can return just the matrix of subpaths values if parameter `matrix` is set to `TRUE`.

```
path_vals <- get_paths_data(results, matrix = TRUE)
path_vals <- get_paths_data(results)
hhead(path_vals, 4)
##               TCGA.BH.A1FM.11B.23R.A13Q.07 TCGA.E2.A1LB.11A.22R.A144.07
## P-hsa03320-37                   0.42896593                   0.37775624
## P-hsa03320-61                   0.19398922                   0.19805778
## P-hsa03320-46                   0.09120405                   0.08985045
## P-hsa03320-57                   0.10329939                   0.10176627
##               TCGA.BH.A208.11A.51R.A157.07 TCGA.BH.A18K.11A.13R.A12D.07
## P-hsa03320-37                    0.4024025                   0.41396505
## P-hsa03320-61                    0.1655702                   0.18774026
## P-hsa03320-46                    0.0905164                   0.09280719
```

```
## P-hsa03320-57                  0.1025205                  0.13953972
```

Rownames of the matrix of pathway results are the IDs of the subpathways. The object `results` stores also the comprehensive names of the subpathways as `rowData` of the assay `paths`. However, in case we need to transform subpath IDs to comprehensive subpath names, we can use `get_path_names` function, see Section 8.1.2 for further information on this. However, it is not recommended to change the row names of the matrix of subpath values.

Notice that the matrix of subpathway activity values will include a value of activity for each sample and for each possible subpathway of the pathways in the pathway object. Depending on whether parameter `decompose` is set to `TRUE` or `FALSE`, and on the number of pathways included in the object of pathways given as attribute, the number of analyzed subpathways may vary. Currently *hipathia* includes up to the following number of pathways, effector subpathways and decomposed subpathways per species:

	Pathways	Effector subpathways	Decomposed subpathways
hsa	145	1868	8409
mmu	141	1849	8409
rno	141	1845	8220

It is recommended to perform an initial *hipathia* analysis with effector subpathways, and use decomposed subpathways only for specific pathways in which the user is highly interested.

5 Function activation computation

Each effector protein of a pathway is responsible for performing a particular function. Thus, from the matrix of effector subpathways we can infer the functions matrix with the function `quantify_terms`, which computes an intensity value for each molecular function and for each sample.

Different effector subpathways of different pathways may end in the same effector protein, and also different effector proteins may have the same molecular function. Therefore, for a particular function f, `quantify_terms` summarizes the values of all the subpathways ending in an effector protein related to f with a mean value.

Different function activity matrices can be computed depending on the functional annotation given to the effector nodes. Function `quantify_terms`, through parameter `dbannot`, accepts any annotation defined by the user and it has also two default annotations: Gene Ontology functions and Uniprot keywords.

```
uniprot_vals <- quantify_terms(results, pathways, dbannot = "uniprot")
## Quantified functions:142
go_vals <- quantify_terms(results, pathways, dbannot = "GO")
## Quantified functions:1654
```

The result of this function is a data object with the level of activity of each annotated function for each sample. As before, the returned object is a *SummarizedExperiment*, unless parameter `matrix` is set to `TRUE` in which case a matrix is returned.

Notice that functions annotated to genes which are not included in any effector node will be not computed.

6 Pathway/Function activation analysis

6.1 Two classes comparison

Once the object data of desired features has been computed, either subpath values or function values, any kind of analysis may be performed on it, in the same way as if it were the matrix of gene expression. Specifically, comparison of the features across different groups of samples is one of the keys. We can perform a comparison of two groups applying the Wilcoxon test using function `do_wilcoxon`.

```
sample_group <- brca_design[colnames(path_vals),"group"]
comp <- do_wilcoxon(path_vals, sample_group, g1 = "Tumor", g2 = "Normal")
hhead(comp)
##                                         name UP/DOWN  statistic    p.value
## P-hsa03320-37 PPAR signaling pathway: HMGCS2    DOWN -2.2451574 0.04654349
## P-hsa03320-61  PPAR signaling pathway: APOA1    DOWN -0.8926529 0.46628647
## P-hsa03320-46  PPAR signaling pathway: APOA2    DOWN -2.5427084 0.02155103
## P-hsa03320-57  PPAR signaling pathway: APOC3    DOWN -2.5697585 0.02010826
## P-hsa03320-64  PPAR signaling pathway: APOA5    DOWN -0.3246011 0.81227041
##                FDRp.value
## P-hsa03320-37 0.024467723
## P-hsa03320-61 0.383413282
## P-hsa03320-46 0.010314038
## P-hsa03320-57 0.009483607
## P-hsa03320-64 0.758350956
```

Function `get_pathways_summary` returns a summary by pathway of the results from the Wilcoxon test, summaryzing the number of significant up- or down-activated features.

```
pathways_summary <- get_pathways_summary(comp, pathways)
head(pathways_summary, 4)
##                                                          id_pathways
## mTOR signaling pathway                                      hsa04150
## Notch signaling pathway                                     hsa04330
## Signaling pathways regulating pluripotency of stem cells    hsa04550
## Jak-STAT signaling pathway                                  hsa04630
##                                                          num_total_paths
## mTOR signaling pathway                                                 3
## Notch signaling pathway                                                3
## Signaling pathways regulating pluripotency of stem cells              14
## Jak-STAT signaling pathway                                            11
##                                                          num_significant_paths
## mTOR signaling pathway                                                       3
## Notch signaling pathway                                                      3
## Signaling pathways regulating pluripotency of stem cells                    14
## Jak-STAT signaling pathway                                                  11
##                                                          percent_significant_paths
## mTOR signaling pathway                                                         100
## Notch signaling pathway                                                        100
## Signaling pathways regulating pluripotency of stem cells                       100
## Jak-STAT signaling pathway                                                     100
```

```
##                                                          num_up_paths
## mTOR signaling pathway                                              1
## Notch signaling pathway                                             1
## Signaling pathways regulating pluripotency of stem cells            2
## Jak-STAT signaling pathway                                          8
##                                                          percent_up_paths
## mTOR signaling pathway                                              33.33
## Notch signaling pathway                                             33.33
## Signaling pathways regulating pluripotency of stem cells            14.29
## Jak-STAT signaling pathway                                          72.73
##                                                          num_down_paths
## mTOR signaling pathway                                                2
## Notch signaling pathway                                               2
## Signaling pathways regulating pluripotency of stem cells             12
## Jak-STAT signaling pathway                                            3
##                                                          percent_down_paths
## mTOR signaling pathway                                                66.67
## Notch signaling pathway                                               66.67
## Signaling pathways regulating pluripotency of stem cells              85.71
## Jak-STAT signaling pathway                                            27.27
```

In order to visualize the results of the comparison, see Section 7.

6.2 Principal Components Analysis

Principal Components Analysis can be also performed by using function `do_pca`. Notice that the number of rows must not exceed the number of columns of the input matrix.

```
ranked_path_vals <- path_vals[order(comp$p.value, decreasing = FALSE),]
pca_model <- do_pca(ranked_path_vals[1:ncol(ranked_path_vals),])
```

PCA models can be visualized with a specific function called `pca_plot`. See Section 7 for further information.

7 Visualization

7.1 Heatmap

Function `heatmap_plot` plots a heatmap with the values of the given data object. This object may be a SummarizedExperiment object or a matrix. The experimental design can be provided to assign a class to each sample by means of the parameter `group`. Notice that the classes must be in the same order as the columns of the provided matrix. One can select whether to cluster samples or variables setting parameters `variable_clust` and `sample_clust` to `TRUE`.

The colors of the different classes of samples can be selected through parameter `sample_colors` with a vector of colors named after the classes. The colors inside the heatmap can be also selected with parameter `colors`. Personalized colors can be provided as a vector, or preselected color schemes *classic* (default), *hipathia* or *redgreen* may be chosen.

```
heatmap_plot(path_vals, group = sample_group)
```

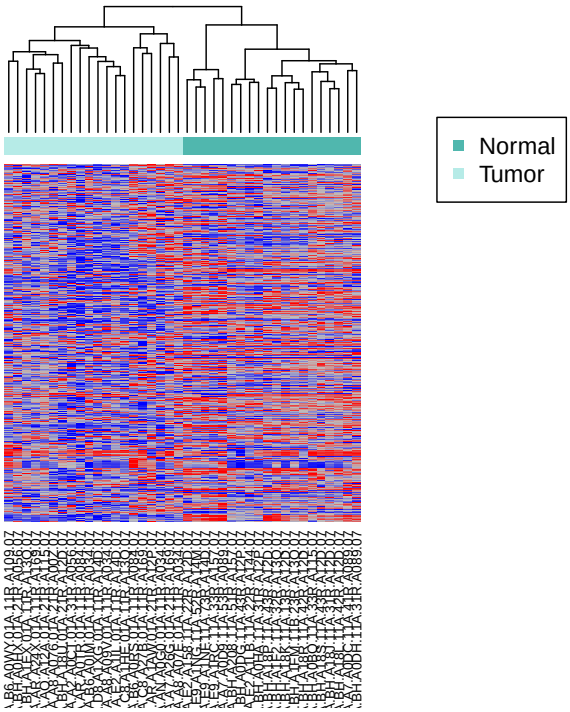

Figure 9: Heatmap plot

```
heatmap_plot(uniprot_vals, group = sample_group, colors="hipathia",
          variable_clust = TRUE)
```

```
heatmap_plot(go_vals, group = sample_group, colors="redgreen",
          variable_clust = TRUE)
```

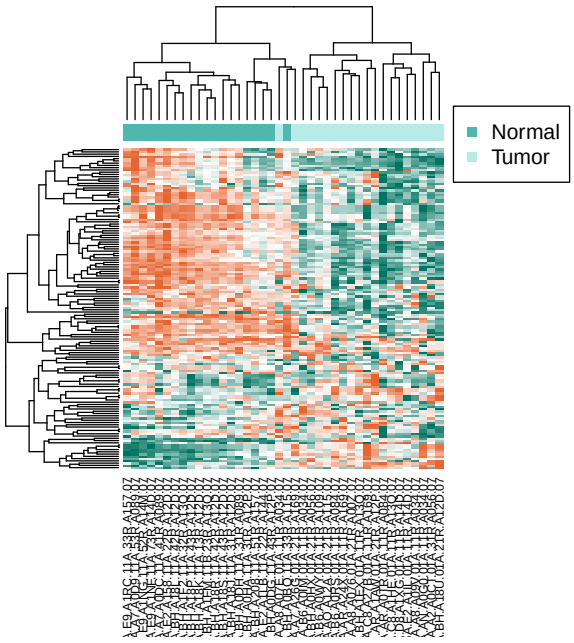

Figure 10: Heatmap plots with variable clustering

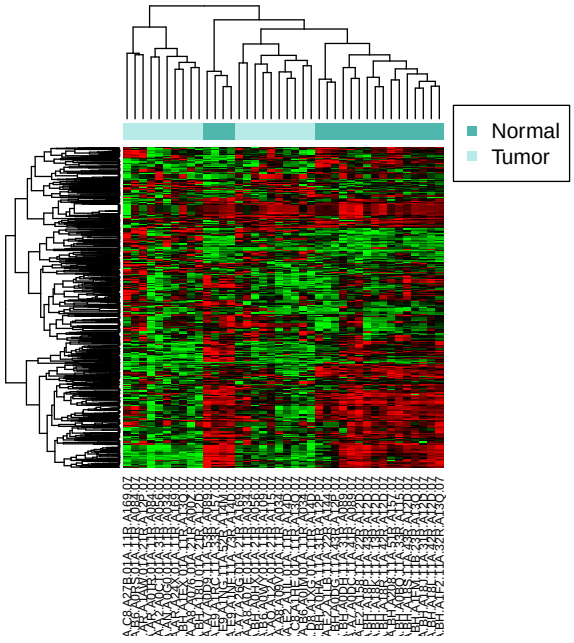

Figure 11: Different colors of heatmaps: 'redgreen'

7.2 PCA

Function `pca_plot` plots two components of a PCA model computed with function `do_pca` (see Section 6). The experimental design can be provided to assign a class to each sample by means of the parameter `group`. Notice that the classes must be in the same order as the columns of the matrix provided to the PCA model. The colors of the different classes of

samples can be selected through parameter `sample_colors` with a vector of colors named after the classes. If no such parameter is provided, a predefined set of colors will be assigned. A main title may be given to the plot through parameter `main`. The components to be plotted can be selected through parameters `cp1` and `cp2` giving integer number. If parameter `legend` is set to TRUE, the legend will be plotted.

```
pca_plot(pca_model, sample_group, legend = TRUE)
```

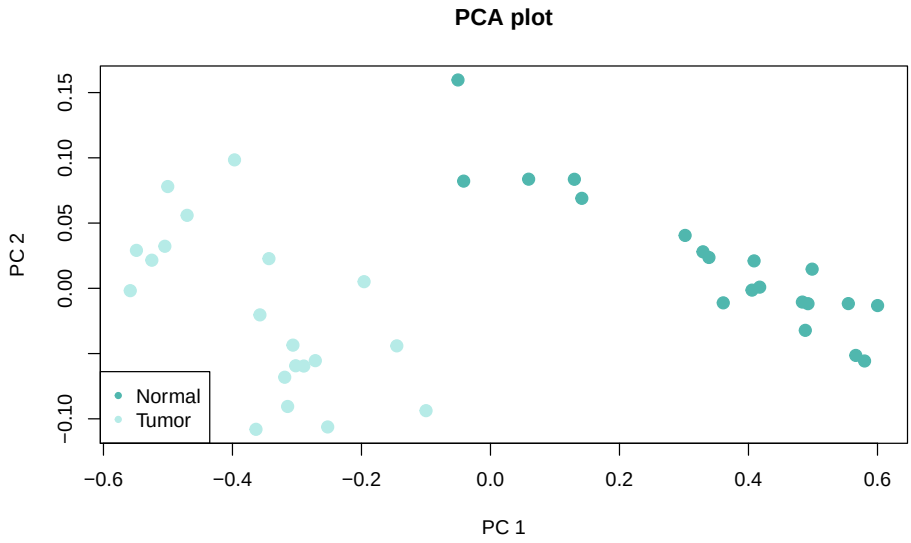

Figure 12: PCA plot

```
pca_plot(pca_model, group = rep(1:5, 8), main = "Random types",
      legend = TRUE)
```

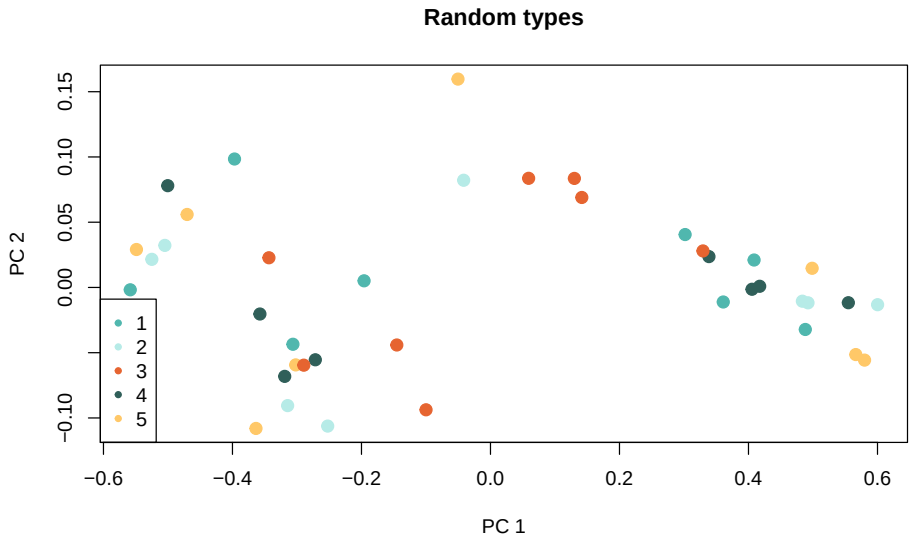

Figure 13: PCA plot with 5 random colors

Function `multiple_pca_plot` plots n PCA components given by parameter `comps=n` as an integer vector. By default, $n = 3$. As before, the experimental design can be provided to assign a class to each sample by means of the parameter `group`. Notice that the classes must be in the same order as the columns of the matrix provided to the PCA model. The colors of the different classes of samples can be selected through parameter `sample_colors` with a vector of colors named after the classes. If no such parameter is provided, a predefined set of colors will be assigned. The cumulative explained variance can be represented by setting `plot_variance` parameter to TRUE. If parameter `legend` is set to TRUE, the legend will be plotted. A main title may be given to the plot through parameter `main`.

```
multiple_pca_plot(pca_model, sample_group, cex=3, plot_variance = TRUE)
```

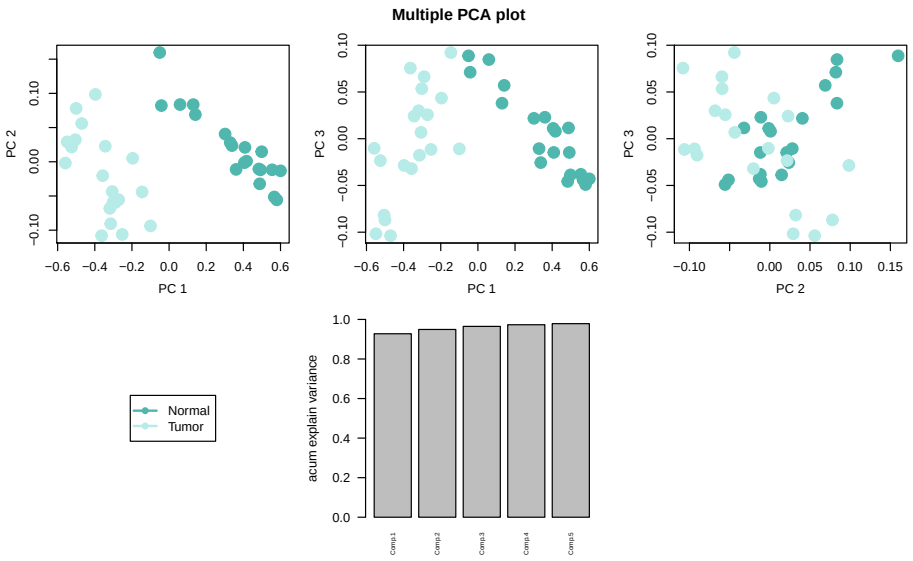

Figure 14: **Multiple PCA plot with acumulated explained variance**

7.3 Pathway comparison

The results of a comparison are sometimes difficult to summarize. An easy way to understand these results is to visualize them as an image. Function `pathway_comparison_plot` creates an image of a pathway, with the same layout from KEGG, including a color code representing the significant up- and down-activated subpathways, and, if desired, the significant up- and down-regulated nodes.

```
pathway_comparison_plot(comp, metaginfo = pathways, pathway = "hsa03320")
```

In these plots, colored edges represent significant subpathways. Edges belonging to subpathways which are significantly down-activated will be depicted in blue and those belonging to subpathways which are significantly up-activated will be depicted in red (as default). The *up* and *down* colors may be changed by the user through the parameter `colors` by giving a vector with three colors (representing down-activation, non-significance and up-activation respectively) or a color scheme (either *classic* or *hipathia*).

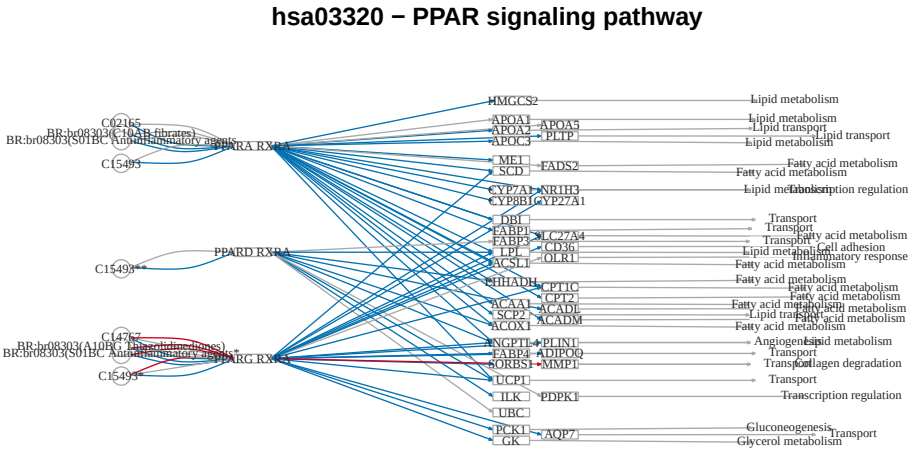

Figure 15: Pathway comparison plot without node colors

In order to visualize the effect of the nodes expression differences in the pathways, nodes can be colored by its differential expression. The color of each node with respect to its differential expression must be previously computed using function `node_color_per_de`. Note that this fucntion computes differential expression on the nodes, not on the genes. It uses function `eBayes` from package *limma*, see the package vignette for further information.

When computed, the resulting object must be provided to the `pathway_comparison_plot` function as parameter `node_colors`.

```
colors_de <- node_color_per_de(results, pathways, sample_group, "Tumor",
                               "Normal")
pathway_comparison_plot(comp, metaginfo = pathways, pathway = "hsa03320",
                        node_colors = colors_de)
```

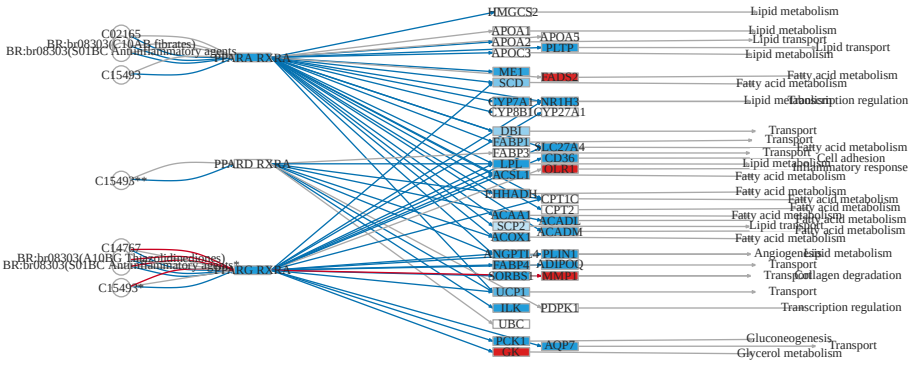

Figure 16: Pathway comparison plot with node colors: 'classic'

```
colors_de_hipathia <- node_color_per_de(results, pathways, sample_group,
                                        "Tumor", "Normal", colors = "hipathia")
pathway_comparison_plot(comp, metaginfo = pathways, pathway = "hsa03320",
                        node_colors = colors_de_hipathia, colors = "hipathia")
```

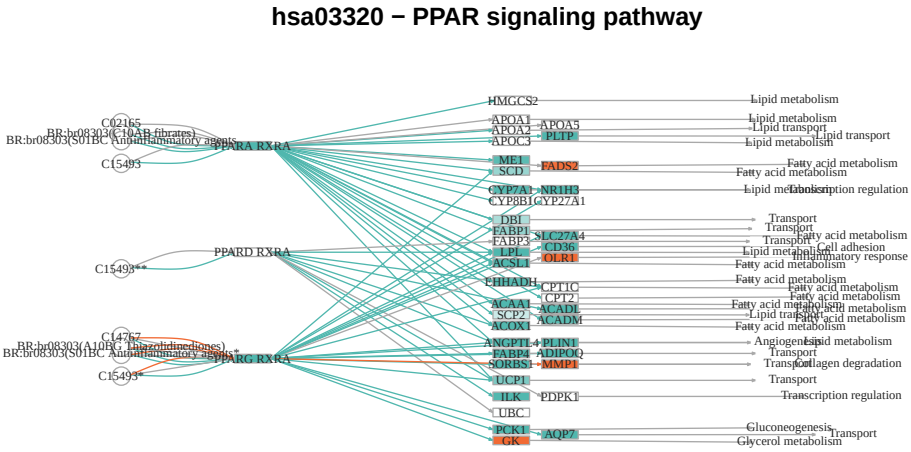

Figure 17: Pathway comparison plot with node colors: 'hipathia'

7.4 Visualization through a local server

Hipathia results can be viewed on a web browser interactivelly. In order to save the files for their visualization, use function `create_report`. To visualize the created report, use function `visualize_report`. For the interpretation of the results in this visualization, see Section 7.6.

The parameters to be provided to function `create_report` are the object of results, the Wilcoxon comparison, the pathways object and the path to the folder in which the files must be saved. Optionally, the colors of the nodes showing their differential expression can be also provided using an object returned by function `node_color_per_de` or a similar data structure.

```
report <- create_report(comp, pathways, "save_noColors")
## Creating report folders...
## Creating pathways folder...
## Creating HTML index...
report_colors <- create_report(comp, pathways, "save_colors",
                               node_colors = colors_de)
## Creating report folders...
## Creating pathways folder...
## Creating HTML index...
```

Due to cross-origin security restrictions (CORS), a web server is needed to serve the result files correctly. The easiest way to run a web server to show the result files is with the *hipathia* function `visualize_report`. The user must specify the folder where the report has been stored by function `create_report`. A web server developed in R will be executed, serving the result files to the default URL http://127.0.0.1:4000. Port 4000 may be changed through parameter `port`.

```
visualize_report(report_colors)
## Serving the directory C:\Users\biocbuild\bbs-3.7-bioc\tmpdir\Rtmpa6OlYE\save_colors\pathway-viewer at http
## Open a web browser and go to URL http://127.0.0.1: 4000
```

```
visualize_report(report, port = 4001)
## Serving the directory C:\Users\biocbuild\bbs-3.7-bioc\tmpdir\Rtmpa6OlYE\save_noColors\pathway-viewer at ht
## Open a web browser and go to URL http://127.0.0.1: 4001
```

The function `visualize_report` uses the *servr* package, please refer to the package documentation for further information.

The servers will be active until function `daemon_stop` from package `servr` is executed. Information about how to stop each server individually is given as an output of each `visualize_report` function. To stop all servers at a time, use

```
servr::daemon_stop()
```

Alternatively, if you have already a web server installed in your computer, just link or move the output folder to your web server http document root and then open it on your web browser.

7.5 Visualization through a local server with different groupings

Effector subpathway results are shown by default grouped by the pathway to which they belong. However, if our interest is to see the comparison of all the subpathways arriving to a particular function, we can group the subpathways by Uniprot or GO functions. Moreover, if we want to see the results of all the subpathways containing a particular gene, we can group the subpathways by genes.

In order to do that, we must use the `group_by` parameter of functions `node_color_per_de` and `create_report`. Available `group_by` parameter values include: `uniprot`, to group subpathways arriving to the same Uniprot functions, `GO`, to group subpathways arriving to the same GO terms, and `genes`, to group subpathways containing each particular gene.

```
colors_de_uni <- node_color_per_de(results, pathways, sample_group, "Tumor",
                                   "Normal", group_by = "uniprot")
create_report(comp, pathways, "save_colors_uniprot",
              node_colors = colors_de_uni, group_by = "uniprot")
visualize_report("save_colors_uniprot", port = 4002)
```

As before, to stop the server and free the port, use the information about how to stop each server individually in the output of each `visualize_report` function or stop all servers at a time, using

```
servr::daemon_stop()
```

7.6 Interpreting HTML results

The interactive visualization of *hipathia* results includes three panels and a legend. The legend is on top of the page resuming the main information depicted in the images. The left panel is the pathways panel, where the currently selected pathway is shown. The layout of the pathway is similar to the layout shown in KEGG.

As before, edges belonging to significant down-activated pathways are depicted in blue, those belonging to significant up-activated subpathways are depicted in red, and those belonging to non-significant subpathways are depicted in grey. Similarly, when nodes are colored by their differential expression, down-regulated nodes are colored in blue, up-regulated nodes are colored in red and non-significant nodes are colored in white. Different shades of the colors indicate different levels of significance with respect to the p-value of the differential expression.

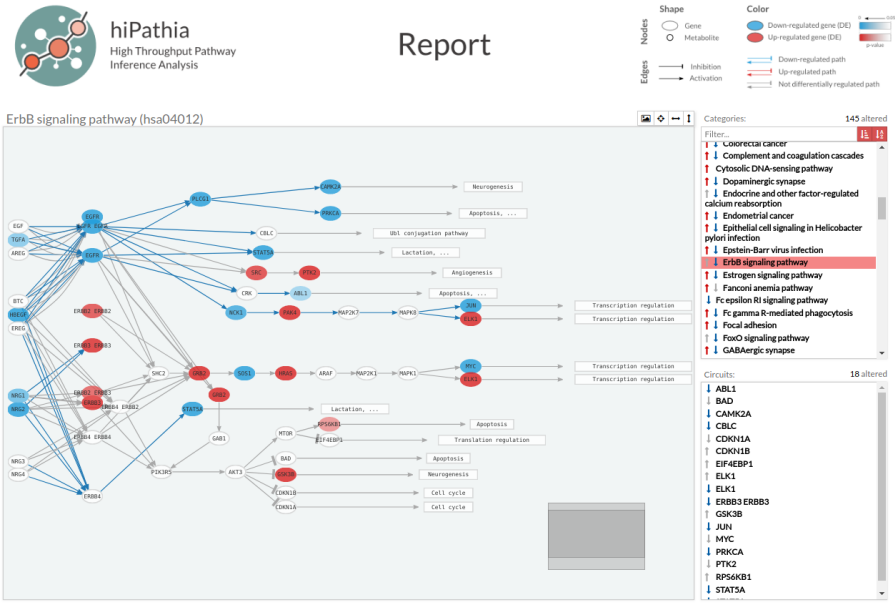

Figure 18: **Interactive Hipathia report visualization**

The selected pathway to be shown can be modified through the pathway list in the top right panel. Arrows pointing up and down to the left of the names of the pathways indicates that the pathways contain up- or down-activated subpathways, respectively. When the arrows are colored in red or blue, it means that there are significant up- or down-regulated subpathways, respectively. The pathways list can be filtered through the *Filter...* box, or ordered by means of the buttons in the top right part of the panel.

All computed subpathways of the currently selected pathway are listed in the subpathways list in the bottom right panel. Arrows pointing up and down by the names of the subpathways indicates that they are up- or down-activated, respectively. When the arrows are colored in red or blue, it means that they are significantly up- or down-regulated, respectively. When a subpathway is selected from the list, only the arrows and nodes belonging to this subpathway will be highlighted. Clicking again on this subpathway will deselect it.

8 Utilities

8.1 Functions

We have developed some simple functions to ease the use of data in *hipathia*.

8.1.1 hhead

Function `hhead` has been conceived as a generalization of function `head` to matrices, dataframes and SummarizedExperiment objects. It returns the values of the n first rows and columns of the matrix. In case the object is a SummarizedExperiment, it returns the values of the n first rows and columns of the (first) assay included in it. In case the object is not a matrix, dataframe or SummarizedExperiment object, it returns the result of applying function `head` to the object.

```
class(brca)
## [1] "SummarizedExperiment"
## attr(,"package")
## [1] "SummarizedExperiment"
hhead(brca, 4)
##      TCGA.BH.A1FM.11B.23R.A13Q.07 TCGA.E2.A1LB.11A.22R.A144.07
## 2                     10.5317320                     9.732938
## 8647                  -3.3266788                    -3.457515
## 5244                  -0.3600828                    -1.139309
## 1244                   2.2876961                     1.724625
##      TCGA.BH.A208.11A.51R.A157.07 TCGA.BH.A18K.11A.13R.A12D.07
## 2                      9.7958036                    10.868669
## 8647                  -2.5261155                    -3.584934
## 5244                  -0.7368491                    -1.257797
## 1244                   1.0217356                     1.467979
```

```
class(assay(brca))
## [1] "matrix"
hhead(assay(brca), 4)
##      TCGA.BH.A1FM.11B.23R.A13Q.07 TCGA.E2.A1LB.11A.22R.A144.07
## 2                     10.5317320                     9.732938
## 8647                  -3.3266788                    -3.457515
## 5244                  -0.3600828                    -1.139309
## 1244                   2.2876961                     1.724625
##      TCGA.BH.A208.11A.51R.A157.07 TCGA.BH.A18K.11A.13R.A12D.07
## 2                      9.7958036                    10.868669
## 8647                  -2.5261155                    -3.584934
## 5244                  -0.7368491                    -1.257797
## 1244                   1.0217356                     1.467979
```

8.1.2 get_path_name

The results object returned by function `hipathia` includes the names of the subpathways, and they are shown in the table returned by function `do_wilcoxon`. However, in case we need to transform subpath IDs to comprehensive subpath names, we can use `get_path_names` function:

```
get_path_names(pathways, c("P-hsa03320-37", "P-hsa04010-15"))
## [1] "PPAR signaling pathway: HMGCS2" "MAPK signaling pathway: NFKB1"
```