Introduction

In this tutorial we are going to analyze the dataset of metabolic activities published in Terunuma et al (2017). In this paper the Authors characterized the transcriptomic and metabolomic profile of several types of human breast tumors to uncovered metabolite signatures using an untargeted discovery approach. They found that the oncometabolite 2-hydroxyglutarate (2HG) accumulates at high levels in a subset of tumors and discovered an association between increased 2HG levels and MYC pathway activation in breast cancer. As an example of mixed metabolites and genes topological analyses here we are going to use both the dataset of metabolite abundances and gene expression as reported in the supplementary files selecting only tumour samples and comparing ER+ and ER- breast cancers. We will use pathway information from the KEGG and SMPDB databases to identify a set of features whose activity changes significantly between the two sample classes. This result will hopefully hint at some specific biological activities that are pathologically altered in tumoral samples.

Data Preparation

We start by loading the experimental data into R. The measurements are stored in two simple text files, one for the genes and one for the metabolites. Thanks to the utility functions provided by R, we can combine the download and the actual reading of a table in a single operation. We also specify the details of the file format: the first row of input represents the header of the table; the tabulation character \t separates values; we don’t want R to change strings into factors.

```r
gene_url <-
gexpr <- read.table(gene_url, header = TRUE, sep = "\t", row.names = 1,
                    stringsAsFactors = FALSE)

metab_url <-
  url("https://romualdi.bio.unipd.it/wp-uploads/2018/04/Terunuma_metabolite_expr.txt")
mexpr <- read.table(metab_url, header = TRUE, sep = "\t", row.names = NULL,
                    stringsAsFactors = FALSE)
```

The loaded table for the metabolites contains 436 rows and 112 columns, but not all of them actually correspond to measurements. We ask R to list the names of the columns:

```r
colnames(mexpr)
#> [1] "METABOLON_ID"  "KEGG_ID"    "CAS"          "HMDB_ID"
#> [5] "POS.NORMAL"    "POS.TUMOR"  "POS.NORMAL.1" "POS.TUMOR.1"
#> [9] "NEG.TUMOR"     "NEG.NORMAL" "NEG.TUMOR.1"  "POS.NORMAL.2"
#> [13] "POS.TUMOR.2"   "POS.TUMOR.3" "POS.NORMAL.3" "NEG.NORMAL.1"
#> [17] "NEG.TUMOR.2"   "NEG.NORMAL.2" "NEG.TUMOR.3"  "NEG.NORMAL.3"
#> [21] "NEG.TUMOR.4"   "POS.NORMAL.4" "POS.TUMOR.4"  "POS.TUMOR.5"
#> [25] "POS.NORMAL.5"  "POS.TUMOR.6" "NEG.NORMAL.4"  "NEG.TUMOR.5"
#> [29] "NEG.NORMAL.5"  "NEG.NORMAL.6" "NEG.TUMOR.6"  "NEG.NORMAL.7"
#> [33] "NEG.TUMOR.7"   "NEG.TUMOR.8" "POS.NORMAL.6"  "POS.TUMOR.7"
#> [37] "POS.NORMAL.7"  "POS.NORMAL.8" "POS.TUMOR.8"  "POS.TUMOR.9"
#> [41] "POS.TUMOR.10"  "POS.NORMAL.9" "POS.NORMAL.10" "POS.TUMOR.11"
```
As you can see the first four columns actually contain metabolite identifiers. We store their indices in a variable for future reference.

```r
idcols <- 1:4
```

### Missing Values

We display the first few rows of the table to get a feeling for its content. For the moment, we concentrate on the metabolite measurements.

```r
head(mexpr[,5:10])
```

```
#> POS.NORMAL  POS.TUMOR  POS.NORMAL.1  POS.TUMOR.1  NEG.TUMOR  NEG.NORMAL
#> 1 24562.00 93036.29 108741.08 26753.39 340237.2 32141.90
#> 2 NA NA 33376.83 33413.49 242799.0 53352.70
#> 3 NA 151243.74 NA 45499.88 592226.4 NA
#> 4 NA NA 37739.89 NA 26460.05 359925.3 NA
#> 5 24939.09 109819.69 71440.01 185000.21 193026.5 94471.26
#> 6 NA 128398.33 NA 26460.05 359925.3 NA
```

It looks like there is a significant number of missing values. Let’s plot a distribution of the fraction of NAs for each row.

```r
nas_by_row <- rowSums(is.na(data.matrix(mexpr[-idcols]))) / (ncol(mexpr) - length(idcols))
hist(nas_by_row, xlab = "Fraction of NA values", ylab = "Number of rows", main = NULL)
```
We cannot make good use of rows with too many missing values. We decide to drop them from our dataset. Specifically, we are going to remove anything with more than 50% of NAs.

dexpr <- mexpr[nas_by_row < 0.5,]

This operation leaves us with 216 of the original 216 rows.

We are not yet satisfied, though. The downstream analysis won’t be able to cope with any NA. What would happen if we were to apply a more stringent procedure, removing any NA?

\[
\text{sum(rowSums(is.na(dexpr[-idcols])) == 0)}
\]

## [1] 51

The above command is a little bit dense, so let’s go through it in small steps. First we select all columns, except the identifiers. Then we check each value, testing if it’s an NA. We tally how many NAs we have in each row and we consider only those in which the count is zero (meaning, no NA at all). The outer sum tells us how many rows would survive our filter.

The final verdict is quite grim. Only 51 out of 216 measurements would be used. We could do much better using a strategy that has become quite common in cases like this: imputation.

Instead of implementing it ourselves, we are going to use the excellent BioConductor package impute.

\[
\text{library(impute)}
\]

\[
iexpr <- \text{cbind(dexpr[,idcols],}
\text{impute.knn(data.matrix(dexpr,-idcols))$data,}
\text{stringsAsFactors = FALSE)}
\]

\[
\text{head(iexpr[, 1:6])}
\]

## METABOLON_ID KEGG_ID CAS HMDB_ID POS.NORMAL POS.TUMOR
## 1 35186 24562.00 93036.29
## 2 34214 C03819 25996.12 114752.62
## 5 27665 C02918 1005-24-9; HMDB00699 24393.09 109819.69
## 8 33960 C03916 19420-56-5; HMDB02815 62594.21 393447.09
## 11 33955 C04102 17364-16-8; 49247.53 3392199.22
## 12 21188 C01885 123-94-4; 24076.66 132101.01

We can explicitly check there are not NAs left:

```r
sum(is.na(iexpr[,!idcols]))
## [1] 0
```

### Missing Identifiers

We now concentrate on the metabolite identifiers. We start again from the first few rows.

```r
head(iexpr[,idcols])
## METABOLON_ID KEGG_ID CAS HMDB_ID
## 1 35186 <NA> <NA> <NA>
## 2 34214 C03819 <NA> <NA>
## 5 27665 C02918 1005-24-9 HMDB00699
## 8 33960 C03916 19420-56-5 HMDB02815
## 11 33955 C04102 17364-16-8 <NA>
## 12 21188 C01885 123-94-4 <NA>
```

The CAS identifiers have a trailing ; we don’t need and there are a number of empty strings, which really represent missing values. We fix both these issues.

```r
iexpr$CAS <- sub("\s*;.*$", "", iexpr$CAS)
iexpr[,idcols][iexpr[,idcols] == ""] <- NA
```

```r
head(iexpr[,idcols])
## METABOLON_ID KEGG_ID CAS HMDB_ID
## 1 35186 <NA> <NA> <NA>
## 2 34214 C03819 <NA> <NA>
## 5 27665 C02918 1005-24-9 <NA>
## 8 33960 C03916 19420-56-5 <NA>
## 11 33955 C04102 17364-16-8 <NA>
## 12 21188 C01885 123-94-4 <NA>
```

We get a measure of how many identifiers are present or missing:

```r
summary(is.na(iexpr[,idcols]))
## METABOLON_ID KEGG_ID CAS HMDB_ID
## Mode :logical Mode :logical Mode :logical Mode :logical
## FALSE:216 FALSE:118 FALSE:133 FALSE:120
## TRUE :98 TRUE :83 TRUE :96
```

We have a Metabolon ID for each metabolite in the matrix, while in the case of CAS identifiers we find only 133 usable rows.

Unfortunately, at this point graphite does not support the Metabolon IDs. We do rely on CAS even if that means loosing a significant fraction of the rows.

```r
valid_cas <- !is.na(iexpr$CAS)
cas_col <- which(names(iexpr) == "CAS")
cexpr <- iexpr[valid_cas, c(cas_col, seq.int(5, ncol(iexpr)))]
head(cexpr[,1:6])
```
There is no reason at this point to keep the identifiers as a column inside the dataset. We move such information to the row names and transform the data.frame into a numeric matrix.

```
mexpr <- data.matrix(cexpr[, -1])
rownames(mexpr) <- paste("CAS", cexpr$CAS, sep = ":")
colnames(mexpr) <- colnames(gexpr)
head(mexpr[, 1:6])
```

Finally, we convert the metabolite levels by taking the logarithm and we merge the two matrices (one for the genes and one for the metabolites) together.

```
fexpr <- rbind(data.matrix(gexpr), log(mexpr))
```

### Pathway Analysis

To make sense of the changes in metabolic activity recorded in the data we have just loaded, we are going to use pathway information from KEGG (via the graphite package) and a statistical analysis capable of exploiting the topology of such pathways (here we'll rely on clipper). **clipper** performs a test on means and covariances between two conditions and it is able to deal with variables of heterogeneous nature.

We import the required packages:

```
library(graphite)
library(clipper)
```

**clipper** will need three pieces of information:

1. a set of pathways
2. the activity matrix (**mexpr** in our case)
3. a vector representing the two sample groups we are going to compare
Getting pathways is quite easy thanks to graphite.

```r
kpaths <- pathways("hsapiens", "kegg")
```

The above command retrieves all KEGG pathways for *Homo sapiens* (301 in total). We take a peek at the first entry:

```r
kpaths[[1]]
```

```
## "Glycolysis / Gluconeogenesis" pathway
## Native ID = hsa:00010
## Database = KEGG
## Species = hsapiens
## Number of nodes = 94
## Number of edges = 1191
## Retrieved on = 27-04-2018
## URL = http://www.kegg.jp/kegg-bin/show_pathway?org_name=hsa&mapno=00010
```

This summary doesn’t tell us which identifiers are used for the nodes in the pathway. We can get that from the list of edges.

```r
head(edges(kpaths[[1]], "mixed"))
```

```
## src_type src dest_type dest direction type
## 1 ENTREZID 130589 KEGGCOMP C00221 directed Process
## 2 ENTREZID 130589 KEGGCOMP C00267 directed Process
## 3 ENTREZID 160287 KEGGCOMP C00022 directed Process
## 4 ENTREZID 1737 KEGGCOMP C15973 directed Process
## 5 ENTREZID 2023 KEGGCOMP C00631 directed Process
## 6 ENTREZID 2026 KEGGCOMP C00631 directed Process
```

As we might expect, KEGG pathways are using KEGG compounds IDs. Since we’re relying on CAS in our data, we should convert them. *convertIdentifiers* to the rescue!

```r
rpaths <- convertIdentifiers(convertIdentifiers(kpaths, "ENSEMBL"), "CAS")
```

The edges of the first pathway have been converted into:

```r
head(edges(rpaths[[1]], "mixed"))
```

```
## src_type src dest_type dest direction type
## 1 CAS 57-60-3 ENSEMBL ENSG00000166800 directed Process
## 2 CAS 57-60-3 ENSEMBL ENSG00000166800 undirected Process
## 3 CAS 57-60-3 ENSEMBL ENSG00000134333 directed Process
## 4 CAS 57-60-3 ENSEMBL ENSG00000134333 undirected Process
## 5 CAS 57-60-3 ENSEMBL ENSG00000111716 directed Process
## 6 CAS 57-60-3 ENSEMBL ENSG00000111716 undirected Process
```

To make the rest of the analysis more robust, we are going to filter pathways requiring that 1) metabolites represent at least 30% of the nodes and 2) at least 10 edges connect genes or metabolites for which we have experimental measurements.

```r
filter_pathways <- function(pathways, expr, min_edge_num) {
  node_names <- rownames(expr)

  pred <- function(p) {
    ns <- nodes(p, "mixed")
    # Add your conditions here
  }

  filtered <- pathways[unlist(lapply(pathways, pred, expr = expr, min_edge_num = min_edge_num))]
  return(filtered)
}
```

```r
filter_pathways(kpaths, expr, min_edge_num = 10)
```
```r
if (length(ns) == 0) {
    return(FALSE)
}

frac <- length(grep("^CAS:", ns)) / length(ns)
if (frac < 0.3) {
    return(FALSE)
}

es <- edges(p, "mixed")
mask <- (paste(es$src_type, es$src, sep = ":") %in% node_names) &
       (paste(es$dest_type, es$dest, sep = ":") %in% node_names)
sum(mask) >= min_edge_num
}

Filter(pred, pathways)

fpaths <- filter_pathways(rpaths, fexpr, 10)

Now about sample classes. We are going to split our dataset into two: POS TUMOR samples in the first group
and NEG TUMOUR samples in the other. clipper wants from us a vector with as many entries as there are
samples. An entry should be 1 if the corresponding samples belongs to the first class, or 2 in the other case.

pos_indices <- grep("^POS", colnames(fexpr))
neg_indices <- grep("^NEG", colnames(fexpr))

classes <- numeric(length = ncol(fexpr))
classes[neg_indices] <- 1
classes[pos_indices] <- 2

classes
## [1] 2 2 2 2 1 1 1 2 2 2 2 1 1 1 1 1 1 2 2 2 2 1 1 1 1 1 1 2 2 2 2
## [36] 2 2 2 2 2 1 1 1 1 1 1 2 1 2 2 2 2 1 1 2 2 2 2 1 1 1 1 1 1 1 1
## [71] 2 2 2 2 2 1 1 2 2 2 2 1 1 1 1 1 2 1 1 2 1 1 1 1 1 1 2 2 2 2 2 2
## [106] 2 2 2

We use the helper function runClipper to start our analysis. Note that we explicitly require metabolites
from the pathways.

out <- runClipper(fpaths, fexpr, classes, "mean", "mixed",
                  maxNodes = 150, seed = 42)

We check that there are no errors and the we extract the list of pathways which appears to be significantly
altered between the two conditions according to clipper.

names(out$results)
## [1] "Citrate cycle (TCA cycle)"
## [2] "Fatty acid biosynthesis"
## [3] "Arginine biosynthesis"
## [4] "Alanine, aspartate and glutamate metabolism"
## [5] "Cysteine and methionine metabolism"
## [6] "Taurine and hypotaurine metabolism"
## [7] "D-Glutamine and D-glutamate metabolism"
## [8] "Amino sugar and nucleotide sugar metabolism"
## [9] "Glyoxylate and dicarboxylate metabolism"```
We use an helper function to visualize the results.

```r
plot_altered_path <- function(result, pathways, node_scale = 2) {
  title <- names(result)
  pathway <- pathways[[title]]

  graph <- pathwayGraph(pathway, which = "mixed")

  labels <- sapply(nodes(graph), function(n) {
    n2 <- sub("^CAS:", "", n)
    if (n != n2) {
      n2
    } else {
      mapIds(org.Hs.eg.db, sub("^ENSEMBL:", "", n), "SYMBOL", "ENSEMBL",
             multiVas = "first")
    }
  })

  names(labels) <- nodes(graph)

  altered <- unlist(strsplit(result[[1]][1, "pathGenes"], ",;\)
  selected <- nodes(graph) %in% altered
  node_colors <- ifelse(selected, "red", "black")
  names(node_colors) <- nodes(graph)

  base <- 0.75
  heights <- ifelse(selected, base * node_scale, base)
  names(heights) <- nodes(graph)
  widths <- ifelse(selected, base * node_scale, base)
  names(widths) <- nodes(graph)

  base <- 14
  fontsizes <- ifelse(selected, base * node_scale, base)
  names(fontsizes) <- nodes(graph)

  between_altered <- function(edge_names, altered) {
    sapply(edge_names, function(edge_name) {
      nodes <- unlist(strsplit(edge_name, "~", fixed = TRUE))
      all(nodes %in% altered)
    })
  }

  edge_colors <- ifelse(between_altered(edgeNames(graph), altered),
                        "red", "black")

  plot(graph,
       attrs = list(edge = list(arrowsize = 0.5)),
       nodeAttrs = list(label = labels, color = node_colors, width = widths,
                        height = heights, fontsize = fontsizes),
       edgeAttrs = list(color = edge_colors),
       recipEdges = "combined", main = title)
}
```
selection <- c("Alanine, aspartate and glutamate metabolism",
"Arginine biosynthesis",
"D-Glutamine and D-glutamate metabolism"
)
for (idx in seq_along(out$results)) {
    res <- out$results[idx]
    if (names(res) %in% selection) {
        plot_altered_path(res, fpaths)
    }
}
Alanine, aspartate and glutamate metabolism
graphite analyses highlight “Alanine, aspartate and glutamate metabolism” and “Arginine biosynthesis”. These pathways enclose most of the genes and metabolites discussed in Terunuma et al. that follow the abundance pattern of 2HG such as N-acetyl amino acids, especially the mitochondrial N-acetyl-aspartate (NAA), the glutaminase (GLS1) protein, and the aspartoacylase (ASPA), which is generally reduced in breast tumors but most significantly within the ER-negative tumor group which suggests a possible cause for increased tumor-associated NAA. Another graphite pathway worthy of note is “D-Glutamine and D-glutamate metabolism” reported also in Terunuma et al. because it contains L-Glutamine (CAS:56-85-9) and L-Glutamic Acid (CAS:56-86-0), two of the most altered metabolites between ER+ and ER- breast tumours.

SMPDB Pathways

```r
spaths <- pathways("hsapiens", "smpdb")
fpaths <- filter_pathways(
    convertIdentifiers(convertIdentifiers(spaths, "ENSEMBL"), "CAS"),
    fexpr,
    10)
out <- runClipper(fpaths, fexpr, classes, "mean", "mixed",
                  maxNodes = 150, seed = 42)
names(out$results)
## [1] "Citrullinemia Type I"
## [2] "Carbamoyl Phosphate Synthetase Deficiency"
## [3] "Argininosuccinic Aciduria"
```
# Allen's Enzyme List

1. Glycine and Serine Metabolism
2. Ammonia Recycling
3. Arginine and Proline Metabolism
4. Methionine Metabolism
5. Sphingolipid Metabolism
6. Lysine Degradation
7. Amino Sugar Metabolism
8. Pyrimidine Metabolism
9. Purine Metabolism
10. Citric Acid Cycle
11. Urea Cycle
12. Tryptophan Metabolism
13. Glutamate Metabolism
14. 2-Hydroxyglutaric Aciduria (D And L Form)
15. Adenosine Deaminase Deficiency
16. Adenylosuccinate Lyase Deficiency
17. AICA-Ribosiduria
18. Beta Ureidopropionase Deficiency
19. Cystathionine Beta-Synthase Deficiency
20. Dihydropyrimidinase Deficiency
21. Dihydropyrimidine Dehydrogenase Deficiency (DHPD)
22. Dihydropyrimidine Dehydrogenase Deficiency (DHPD)
23. Glutaric Aciduria Type I
24. Guanidinoacetate Methyltransferase Deficiency (GAMT Deficiency)
25. Leigh Syndrome
26. MNGIE (Mitochondrial Neurogastrointestinal Encephalopathy)
27. Molaydenum Cofactor Deficiency
28. Ornithine Transcarbamylase Deficiency (OTC Deficiency)
29. Prolidase Deficiency (PD)
30. Prolidase Deficiency (PD)
31. Purine Nucleoside Phosphorylase Deficiency
32. Purine Nucleoside Phosphorylase Deficiency
33. Purine Nucleoside Phosphorylase Deficiency
34. Pyruvate Dehydrogenase Complex Deficiency
35. S-Adenosylhomocysteine (SAH) Hydrolase Deficiency
36. Sialuria or French Type Sialuria
37. Sialuria or French Type Sialuria
38. UMP Synthase Deficiency (Orotic Aciduria)
39. Xanthine Dehydrogenase Deficiency (Xanthinuria)
40. Methionine Adenosyltransferase Deficiency
41. Glycine N-methyltransferase Deficiency
42. Non Ketotic Hyperglycinemia
43. Saccharopinuria/Hyperlysine II
44. Salla Disease/Infantile Sialic Acid Storage Disease
45. Dimethylglycine Dehydrogenase Deficiency
46. 4-Hydroxybutyric Aciduria/Succinic Semialdehyde Dehydrogenase Deficiency
47. Sarcosinemia
48. Pyruvate Decarboxylase E1 Component Deficiency (PDHE1 Deficiency)
49. Hyperinsulinism-Hyperammonemia Syndrome
50. Methylenetetrahydrofolate Reductase Deficiency (MTHFRD)
51. Hypermethioninemia
52. Metachromatic Leukodystrophy (MLD)
53. Globoid Cell Leukodystrophy
54. Gaucher Disease
55. Argininemia
56. Hyperprolinemia Type II
Using the SMPDB database we obtained 190 significant pathways among which we found “The oncogenic action of 2-hydroxyglutarate” that is the pathway describing the role of 2HG in tumour environment.