graphite

***Gabriele Sales**[*], **Enrica Calura**[†] **and Chiara Romualdi**[‡]

Department of Biology, University of Padua

[*]gabriele.sales@unipd.it [†]enrica.calura@unipd.it [‡]chiara.romualdi@unipd.it

October 19, 2021

Contents

1 Introduction

graphite (GRAPH Interaction from pathway Topological Environment) was designed to:

- provide networks derived from eight public pathway databases,
- automate the conversion of node identifiers (e.g. from Entrez IDs to gene symbols),

- facilitate the execution of topological pathway analyses on the provided networks.

The pathway databases available in this version are:

- KEGG [1]
- Panther [10]
- PathBank [13]
- PharmGKB [11]
- Reactome [2]
- SMPDB [12]
- WikiPathways [14]

All provided pathways are annotated in human and, since version 1.14, other 13 species are available.

graphite pathways are collected and pre-processed at every BioConductor release (roughly every 6 months). This guarantees the synchronization of the provided data with all other BioConductor annotation packages.

The topological pathway analyses directly supported are:

- *SPIA* [5, 6, 4]
- *topologyGSA* [7]
- *clipper* [8]

Since version 1.24, every *graphite* pathway can be accessed through three separate views:

gene-only network where metabolite nodes have been removed and edges have been propagated through them;

metabolite-only network where protein nodes have been removed and edges have been propagated through them;

mixed network containing both proteins and metabolites, useful for metabolomic and transcriptomic data integration.

2 Pathway topology conversion to gene network

In order to gather curated information about pathways, we have collected data from different public databases that have emerged as reference points for the systems biology community. The KEGG database has been in development by Kanehisa Laboratories since 1995, and is now a prominent knowledge base for integration and interpretation of large-scale molecular data sets generated by genome sequencing and other high-throughput experimental technologies. KEGG is the only pathway database not in `BioPax` format, as it stores the information using the KGML format. KEGG pathways (KGML format) provides maps for both signaling and metabolic pathways [1]. Reactome, backed by the EBI, is one of the most complete repository; it is frequently updated and provides a semantically rich description of each pathway [2]. Finally, Panther [10] is data are a comprehensive, curated database of pathways, protein families, trees, subfamilies and functions available at http://pantherdb.org backed by the University of Southern California. Since version 1.24, *graphite* contains also SMPDB [12] and PharmGKB [11], important resources for metabolomic data analyses.

2.1 Pathway definition

graphite pathways are derived by conversion of `KGML` and `BioPax` data formats. KEGG database provides a separate `xml` file, one for each pathway. Thus, a pathway is defined by all the reactions defined within each file. For the other databases, we define a pathway for each element of type `pathway` in the `BioPax` document.

2.2 Nodes with multiple elements

Within a pathway, a node can correspond to multiple gene products. These nodes with multiple elements can be divided into protein complexes (proteins linked by protein-protein interactions) and the groups containing alternative members (genes generally with similar biochemical functions). When considering signal propagation these groups should be considered differently; the first (hereafter group AND) should be expanded into a clique (all proteins connected to the others), while the second (hereafter group OR) should be expanded without connection among the contained elements.

In the KGML format there are two ways of defining nodes with multiple elements: protein complexes (group AND defined by entry `type="group"`) and group with alternative members (group OR defined by entry `type="gene"`). In the BioPax format only one type of group is allowed: protein complexes (group AND) with type `complex`. However, it often happens that `protein` tag contains multiple `xref` pointing to alternative elements of the process (group OR).

2.3 Metabolite and protein-mediated interactions

Metabolites mediated interactions are interactions for which a metabolite acts as a bridge between two proteins. Since metabolites are not measured during gene expression analysis (using microarray or RNA-seq), their removal from the original network is useful. However, the trivial elimination of the metabolites, without signal propagation, will strongly bias the topology, interrupting the signals that pass through them. If element A is linked to metabolite c and metabolite c is linked to element B, thus elements A should be linked to elements B.

Within the KGML format, there are two different ways of describing a metabolite mediated interaction: i) direct interaction `type="PPrel"` (A interacts whit B through metabolite c) and ii) indirect one `type="PCrel"` (A interacts whit metabolite c and c interacts whit B).

Since proper signal propagation is crucial for topological gene set analysis we decided to include additional rules for the propagation reconstructing a connection between two genes connected through a series of metabolites. Not all metabolites are considered for the propagation because some of them, such as Hydrogen, H2O, ATP, ADP etc., are highly frequent in map descriptions and the signal propagation through them would lead to degenerate too long chains of metabolites. The metabolites not considered for propagation are not characteristic of a specific reaction but secondary substrates/products widely shared among different processes.

Since version 1.24, *graphite* can be used also for metabolomics pathway analyses. We applied the same procedure descibed above to provide network with metabolites only propagating the signal through proteins.

Finally, *graphite* includes pathways containing both proteins and metabolites (without edges propagation) allowing also integrated pathway analyses of gene expression and metabolomic data.

2.4 Edge attributes

graphite allows the user to see the single/multiple relation types that characterize an edge. The type of edges have been preserved to remain as close as possible to the original annotations. Some new types have been introduced due to the needs of the topological conversion. For instance a *Process(indirect)* is introduced when the edge is generated propagating the signal from a gene to another gene passing through metabolites, or from a metabolite to another metabolite passing through proteins.

2.5 Loading pathways

Human pathways are natively distribuited with the package. Since the version 1.14.0, non-human pathway data are also available and can be dowloaded automatically using the `pathways` function.

`pathways` requires the name of the specie of interest and the name of the pathway database name as follows:

```
> humanReactome <- pathways("hsapiens", "reactome")
> names(humanReactome)[1:10]

 [1] "Interleukin-6 signaling"
 [2] "Apoptosis"
 [3] "Hemostasis"
 [4] "Intrinsic Pathway for Apoptosis"
 [5] "PKB-mediated events"
 [6] "PI3K Cascade"
 [7] "MAPK3 (ERK1) activation"
 [8] "Translesion synthesis by REV1"
 [9] "Translesion synthesis by Y family DNA polymerases bypasses lesions on DNA template"
[10] "Recognition of DNA damage by PCNA-containing replication complex"

> p <- humanReactome[["ABC-family proteins mediated transport"]]
> p

"ABC-family proteins mediated transport" pathway
Native ID       = R-HSA-382556
Database        = Reactome
Species         = hsapiens
Number of nodes = 121
Number of edges = 1866
Retrieved on    = 16-10-2021
URL             = http://reactome.org/PathwayBrowser/#/R-HSA-382556
```

A pathway database is a list of pathways. We can access a pathway through its name, as above, or through its position in the list, as follow:

```
> p <- humanReactome[[21]]
> pathwayTitle(p)

[1] "Activation of BAD and translocation to mitochondria"
```

In the pathway, nodes represent genes/proteins:

```
> head(nodes(p))

[1] "UNIPROT:P10415" "UNIPROT:P31749" "UNIPROT:P31751" "UNIPROT:P48454"
[5] "UNIPROT:P63098" "UNIPROT:Q92934"
```

Edges can be characterized by multiple functional relationships:

```
> head(edges(p))

  src_type    src dest_type   dest  direction
1  UNIPROT P10415   UNIPROT P55957 undirected
2  UNIPROT P31749   UNIPROT Q92934   directed
3  UNIPROT P31751   UNIPROT Q92934   directed
4  UNIPROT P48454   UNIPROT P27348   directed
5  UNIPROT P48454   UNIPROT P31946   directed
6  UNIPROT P48454   UNIPROT P31947   directed
                                          type
1                                      Binding
2 Control(Out: ACTIVATION of BiochemicalReaction)
3 Control(Out: ACTIVATION of BiochemicalReaction)
4 Control(Out: ACTIVATION of BiochemicalReaction)
5 Control(Out: ACTIVATION of BiochemicalReaction)
6 Control(Out: ACTIVATION of BiochemicalReaction)
```

By default, the function `edges` and `nodes` provide the edges or the nodes of the pathway containing only proteins with signals propagated through metabolites. Since version 1.24, using the option *which* we can access the pathway with proteins and metabolites (*which* = *"mixed"*) or the pathway with only metabolites with edges propagated through proteins (*which* = "metabolites").

```
> head(nodes(p), which = "mixed")

[1] "UNIPROT:P10415" "UNIPROT:P31749" "UNIPROT:P31751" "UNIPROT:P48454"
[5] "UNIPROT:P63098" "UNIPROT:Q92934"

> head(edges(p), which = "mixed")

  src_type    src dest_type   dest  direction
1  UNIPROT P10415   UNIPROT P55957 undirected
2  UNIPROT P31749   UNIPROT Q92934   directed
3  UNIPROT P31751   UNIPROT Q92934   directed
4  UNIPROT P48454   UNIPROT P27348   directed
5  UNIPROT P48454   UNIPROT P31946   directed
6  UNIPROT P48454   UNIPROT P31947   directed
                                          type
1                                      Binding
2 Control(Out: ACTIVATION of BiochemicalReaction)
3 Control(Out: ACTIVATION of BiochemicalReaction)
4 Control(Out: ACTIVATION of BiochemicalReaction)
5 Control(Out: ACTIVATION of BiochemicalReaction)
6 Control(Out: ACTIVATION of BiochemicalReaction)
```

These same steps can be used to access the KEGG, Panther, PathBank, PathBank, PharmGKB, Reactome, SMPDB and WikiPathways databases for *Homo sapiens*, but not all such databases are available for all other species. To know the pathway data available the user can use the `pathwayDatabases`.

```
> pathwayDatabases()
```

```
         species     database
1      athaliana         kegg
2      athaliana     pathbank
3      athaliana wikipathways
4        btaurus         kegg
5        btaurus     pathbank
6        btaurus     reactome
7        btaurus wikipathways
8       celegans         kegg
9       celegans     pathbank
10      celegans     reactome
11      celegans wikipathways
12   cfamiliaris         kegg
13   cfamiliaris     reactome
14   cfamiliaris wikipathways
15 dmelanogaster         kegg
16 dmelanogaster     pathbank
17 dmelanogaster     reactome
18 dmelanogaster wikipathways
19        drerio         kegg
20        drerio     reactome
21        drerio wikipathways
22         ecoli         kegg
23         ecoli     pathbank
24         ecoli wikipathways
25       ggallus         kegg
26       ggallus     reactome
27       ggallus wikipathways
28      hsapiens         kegg
29      hsapiens      panther
30      hsapiens     pathbank
31      hsapiens     pharmgkb
32      hsapiens     reactome
33      hsapiens        smpdb
34      hsapiens wikipathways
35     mmusculus         kegg
36     mmusculus     pathbank
37     mmusculus     reactome
38     mmusculus wikipathways
39   rnorvegicus         kegg
40   rnorvegicus     pathbank
41   rnorvegicus     reactome
42   rnorvegicus wikipathways
43   scerevisiae         kegg
44   scerevisiae     pathbank
```

```
45   scerevisiae     reactome
46   scerevisiae wikipathways
47       sscrofa         kegg
48       sscrofa     reactome
49       sscrofa wikipathways
50       xlaevis         kegg
```

3 Pathway graph

The function `pathwayGraph` builds a *graphNEL* object from a pathway `p`:

```
> g <- pathwayGraph(p)
> g

A graphNEL graph with directed edges
Number of Nodes = 15
Number of Edges = 39
```

```
> edgeData(g)[1]

$`UNIPROT:P10415|UNIPROT:P55957`
$`UNIPROT:P10415|UNIPROT:P55957`$weight
[1] 1

$`UNIPROT:P10415|UNIPROT:P55957`$edgeType
[1] "Binding"
```

Similarly to the `edges` and `nodes`, by default, the function `pathwayGraph` provide the pathway with only proteins and edges propagated through metabolites. Since version 1.24, using the option *which* we can access the pathway with proteins and metabolites (*which* = *"mixed"*) or the pathway with only metabolites with edges propagated through proteins (*which* = "metabolites").

```
> g <- pathwayGraph(p, which = "mixed")
> g

A graphNEL graph with directed edges
Number of Nodes = 19
Number of Edges = 69
```

```
> edgeData(g)[1]

$`CHEBI:15377|CHEBI:43474`
$`CHEBI:15377|CHEBI:43474`$weight
[1] 1

$`CHEBI:15377|CHEBI:43474`$edgeType
[1] "Process(BiochemicalReaction)"
```

4 Identifiers

Gene annotations databases are widely used as public repositories of biological information. Our current knowledge on biological elements is spread out over a number of databases (such as: Entrez Gene , RefSeq, backed by the NCBI http://www.ncbi.nlm.nih.gov/, UniProt, ENSEMBL backed by the EBI http://www.ebi.ac.uk/ to name just a few), specialised on a subset of specific biological entities (for instance, UniProt focuses on proteins while Entrez Gene focuses on genes). Key identifiers (IDs) in the internal structure of each database uniquely represent biological entities, thus biological entities can be identified by homogeneous IDs according to the selected database they refer to. Due to their different origins and specificity, switching from an ID to another is possible but not trivial: there could be either no correspondence among them or many-to-many relations. For detailed information about IDs, their structures and differences please consult those resources.

The function `converterIdentifiers` allows the user to convert the pathway IDs into different types, to fit the user needs. This mapping process, however, may lead to the loss of some nodes (not all identifiers may be recognized) and has an impact on the topology of the network (one ID may correspond to multiple IDs in another annotation or vice versa). We based the function of ID conversion through the species-specific Bioconductor *AnnotationDbi*, such as *org.Hs.eg.db*. Since the version 1.14, all the conversion supported in the annotation packages can be used. `converterIdentifiers` needs a list of pathways or a single pathway and a string describing the type of the identifier as provided by an Annotation package (for example,'"ENTREZID"'), while the values '"entrez"', '"symbol"' remains for backward compatibility.

```
> pSymbol <- convertIdentifiers(p, "SYMBOL")
> pSymbol

"Activation of BAD and translocation to mitochondria" pathway
Native ID       = R-HSA-111447
Database        = Reactome
Species         = hsapiens
Number of nodes = 19
Number of edges = 59
Retrieved on    = 16-10-2021
URL             = http://reactome.org/PathwayBrowser/#/R-HSA-111447

> head(nodes(pSymbol))

[1] "SYMBOL:BCL2"   "SYMBOL:AKT1"   "SYMBOL:AKT2"   "SYMBOL:PPP3CC"
[5] "SYMBOL:PPP3R1" "SYMBOL:BAD"
```

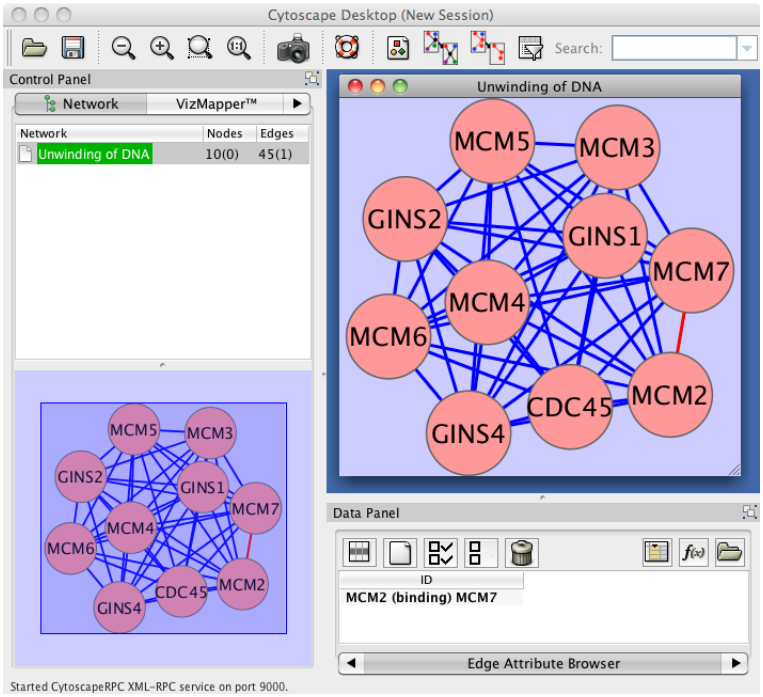
```
> reactomeSymbol <- convertIdentifiers(humanReactome[1:5], "SYMBOL")
```

Since the introduction of metabolites in *graphite* pathways (version 1.24), also the metabolite ID conversion is possible.

5 Cytoscape Plot

Several pathways have a huge number of nodes and edges, thus there is the need of an efficient system of visualization. To this end *graphite* uses the *Rcy3* package to export the network to Cytoscape 3, through the function `cytoscapePlot`. Cytoscape is a Java based software specifically built to manage biological network complexity and for this reason it is widely used by the biological community. Since version 1.24, using the option *which* we can access the pathway with proteins and metabolites (*which* = *"mixed"*) or the pathway with only metabolites and edges propagated through proteins (*which* = "metabolites").

```
> cytoscapePlot(convertIdentifiers(reactome$`Unwinding of DNA`, "symbol"), which =  "mixed")
```



6 Topological pathway analysis

graphite gives access to three types of topological pathway analyses. For more details on the results obtained by these analyses see the corresponding R packages. Following developper instructions, all the methods with the exception of *clipper* are available only for pathways with proteins and with edges propagated through metabolites. *clipper* is the only analysis that can be used to study metabolomics data.

Note that, since version 1.24, given the introduction of nodes with mixed types, all the ID in the pathway have been prefixed with the type of identifier (e.g. ENTREZID:7157 or SYMBOL:TP53). Thus, also the gene expression or metabolite data, in order to match with the pathway nodes, should be in the same format (e.g. ENTREZID:7157 or SYMBOL:TP53).

6.1 SPIA

The analysis with *SPIA* requires the conversion of gene-gene networks in a suitable format. This conversion is performed by the function `prepareSPIA` that must be executed before the analysis command `runSPIA`. The *SPIA* data will be saved in the current working directory; every time you change it, you should also re-run `prepareSPIA`. Edges not included in SPIA have been coerced into the admitted SPIA types. Metabolite mediated interactions (edges of propagation through metabolites) annotated in graphite with the "indirect" type are mapped into the SPIA edge type "indirect effect". For more details see the *SPIA* package [5, 6, 4].

```
> library(SPIA)
> data(colorectalcancer)
> library(hgu133plus2.db)
> top$ENTREZ <- mapIds(hgu133plus2.db,
+                      top$ID, "ENTREZID", "PROBEID", multiVals = "first")
> top <- top[!is.na(top$ENTREZ) & !duplicated(top$ENTREZ), ]
> top$ENTREZ <- paste("ENTREZID", top$ENTREZ, sep = ":")
> tg1 <- top[top$adj.P.Val < 0.05, ]
> DE_Colorectal = tg1$logFC
> names(DE_Colorectal) <- tg1$ENTREZ
> ALL_Colorectal <- top$ENTREZ
> kegg <- pathways("hsapiens", "kegg")
> kegg <- convertIdentifiers(kegg, "ENTREZID")
> kegg <- kegg[c("Bladder cancer", "Cytosolic DNA-sensing pathway")]
> prepareSPIA(kegg, "keggEx")
> res <- runSPIA(de = DE_Colorectal, all = ALL_Colorectal, "keggEx")

Done pathway 1 : Bladder cancer..
Done pathway 2 : Cytosolic DNA-sensing pathway..
Done pathway 3 : <graphite_placeholder>..

> res

                          Name pSize NDE      pNDE         tA pPERT        pG
2 Cytosolic DNA-sensing pathway    41   8 0.6273027  1.0271604 0.786 0.8417147
3               Bladder cancer    28   4 0.8565870 -0.8630436 0.709 0.9101902
      pGFdr pGFWER    Status
2 0.9101902      1 Activated
3 0.9101902      1 Inhibited
```

6.2 topologyGSA

topologyGSA uses graphical models to test the pathway components and to highlight those involved in its deregulation.

In *graphite*, *topologyGSA* has a dedicated function, `runTopologyGSA`, which performs the analysis on a single pathway or on a pathway list.

```
> library(topologyGSA)
> data(examples)
> colnames(y1) <- paste("SYMBOL", colnames(y1), sep = ":")
> colnames(y2) <- paste("SYMBOL", colnames(y2), sep = ":")
> kegg <- pathways("hsapiens", "kegg")
```

```
> p <- convertIdentifiers(kegg[["Fc epsilon RI signaling pathway"]], "SYMBOL")
> runTopologyGSA(p, "var", y1, y2, 0.05)

        Pathway Variance Test

data: exp1, exp2 and g

lambda = 26.02199, df = 10, p-value = 0.003710726, equal variances: TRUE
```

The function `runTopologyGSA`, which easily performs the analysis on the entire pathway database, provides as result a list of two elements: a list with the results of the pathway analyses and the list of generated errors.

For more details see the *topologyGSA* package [7].

6.3 clipper

clipper is a package for topological analysis. It implements a two-step empirical approach based on the exploitation of graph decomposition into a junction tree to reconstruct the most relevant signal path. In the first step clipper selects significant pathways according to statistical tests on the means and the concentration matrices of the graphs derived from pathway topologies. Then, it "clips" the whole pathway identifying the signal paths having the greatest association with a specific phenotype.

In *graphite*, *clipper* has a dedicated function, `runClipper`, which performs the analysis on a single pathway or on a pathway list.

```
> library(ALL)
> library(a4Preproc)
> library(clipper)
> data(ALL)
> pheno <- as(phenoData(ALL), "data.frame")
> samples <- unlist(lapply(c("NEG", "BCR/ABL"), function(t) {
+   which(grepl("^B\\d*", pheno$BT) & (pheno$mol.biol == t))[1:10]
+ }))
> classes <- c(rep(1,10), rep(2,10))
> expr <- exprs(ALL)[,samples]
> rownames(expr) <- paste("ENTREZID",
+                         featureData(addGeneInfo(ALL))$ENTREZID,
+                         sep = ":")
> k <- as.list(pathways("hsapiens", "kegg"))
> selection <- c("Bladder cancer", "Hippo signaling pathway - multiple species")
> selected <- k[selection]
> names(selected) <- c("Bladder cancer", "Hippo signaling")
> clipped <- runClipper(selected, expr, classes, "mean", pathThr = 0.1)
> resClip <- do.call(rbind, clipped$results)
> resClip[, c("startIdx", "endIdx", "maxIdx", "lenght",
+             "maxScore", "aScore")]

                startIdx endIdx maxIdx lenght         maxScore          aScore
Hippo signaling        1      3      2      3 5.83818294928244 2.91909147464122
```

The function `runClipper`, which easily performs the analysis on the entire pathway database, provides as result a list of two elements: the list with the results of the pathway analyses and the list of eventually generated errors.

Since version 1.24, using the option *which* we can access the pathway with proteins and metabolites (*which* = *"mixed"*) or the pathway with only metabolites and edges propagated through proteins (*which* = "metabolites").

For more details see the *clipper* package [8].

7 Build pathway

In *graphite*, it is also possible build a pathway object using `buildPathway`. This function creates a new object of type *Pathway* given a data frame describing its edges. The edges should be divided in three dataframes containing edges between proteins, edges between proteins and metabolites, and edges between metabolites, and the should be included in the new pathway using *proteinEdges*, *mixedEdges* and *metaboliteEdges* respectively. This practice will guarantee the compatibility with `convertIdentifiers`, this only works with the types of identifiers that are provided in the Annotation package (for example using the string,'"ENTREZID"').

```
> edges <- data.frame(src_type = "ENTREZID", src="672",
+                     dest_type = "ENTREZID", dest="7157",
+                     direction="undirected", type="binding")
> pathway <- buildPathway("#1", "example", "hsapiens", "database",
+                         proteinEdges = edges)
```

```
> edges <- data.frame(src_type = "ENTREZID", src="672",
+                     dest_type = "ENTREZID", dest="7157",
+                     direction="undirected", type="binding")
> edgemix <- data.frame(src_type = "CHEBI", src="77750",
+                     dest_type = "ENTREZID", dest="7157",
+                     direction="undirected", type="biochemicalReaction")
> edgemet <- data.frame(src_type = "CHEBI", src="15351",
+                     dest_type = "CHEBI", dest="77750",
+                     direction="undirected", type="biochemicalReaction")
> pathway <- buildPathway("#1", "example", "hsapiens", "database",
+                         proteinEdges = edges,
+                         mixedEdges = edgemix,
+                         metaboliteEdges = edgemet)
```

8 Parallelism

Some of *graphite* operations can be made significantly faster by exploiting the parallelism offered by recent hardware. Here is a list of the functions providing this option:

- `convertIdentifiers`
- `runClipper`
- `runTopologyGSA`

To exploit parallel processing you are going to need two ingredients. First, you have to specify the maximum number of cores that the package can use. For instance, if you have 6 cores on your computer you can set:

```
> options(Ncpus = 6)
```

Your code, then, should try to call *graphite* functions passing entire lists of pathways; you should not loop manually.

```
> original <- pathways("hsapiens", "reactome")
> # Do (will exploit parallelism)
> converted <- convertIdentifiers(original, "SYMBOL")
> # Don't (no parallelism here)
> converted <- lapply(original, convertIdentifiers, "SYMBOL")
```

Warning: Parallelism is not guaranteed to reduce run times. Splitting a given operation over multiple cores requires some coordination, and that has its own costs. In certain situations (small number of pathways / small number of cores) the overhead may actually make run times worse. Our suggestion is thus to:

- always set `Ncpus` to a value smaller or equal to the actual number of hardware cores (see `parallel::detectCores` for an indication);
- in any case, start from a small number like 2 or 4, and then work your way up measuring the actual speedups on your own hardware.

9 Session Information

```
> sessionInfo()

R version 4.1.1 (2021-08-10)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 20.04.3 LTS

Matrix products: default
BLAS:   /home/biocbuild/bbs-3.14-bioc/R/lib/libRblas.so
LAPACK: /home/biocbuild/bbs-3.14-bioc/R/lib/libRlapack.so

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_GB              LC_COLLATE=C
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] stats4    stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
 [1] hgu95av2.db_3.13.0    clipper_1.33.0        Matrix_1.3-4
```

```
 [4] a4Preproc_1.41.0      ALL_1.35.0            topologyGSA_1.4.7
 [7] hgu133plus2.db_3.13.0 SPIA_2.45.0           KEGGgraph_1.53.0
[10] org.Hs.eg.db_3.14.0   AnnotationDbi_1.55.2  IRanges_2.27.2
[13] S4Vectors_0.31.5      Biobase_2.53.0        graphite_1.39.3
[16] graph_1.71.2          BiocGenerics_0.39.2

loaded via a namespace (and not attached):
  [1] colorspace_2.0-2            rjson_0.2.20
  [3] ellipsis_0.3.2              corpcor_1.6.10
  [5] XVector_0.33.0              GenomicRanges_1.45.0
  [7] rstudioapi_0.13             bit64_4.0.5
  [9] fansi_0.5.0                 mvtnorm_1.1-3
 [11] xml2_1.3.2                  R.methodsS3_1.8.1
 [13] cachem_1.0.6                knitr_1.36
 [15] spam_2.7-0                  Rsamtools_2.9.1
 [17] annotate_1.71.0             dbplyr_2.1.1
 [19] png_0.1-7                   R.oo_1.24.0
 [21] BiocManager_1.30.16         compiler_4.1.1
 [23] httr_1.4.2                  backports_1.2.1
 [25] assertthat_0.2.1            fastmap_1.1.0
 [27] htmltools_0.5.2             prettyunits_1.1.1
 [29] tools_4.1.1                 dotCall64_1.0-1
 [31] igraph_1.2.7                qtl_1.50
 [33] gtable_0.3.0                glue_1.4.2
 [35] GenomeInfoDbData_1.2.7      dplyr_1.0.7
 [37] maps_3.4.0                  rappdirs_0.3.3
 [39] Rcpp_1.0.7                  vctrs_0.3.8
 [41] Biostrings_2.61.2           rtracklayer_1.53.1
 [43] xfun_0.27                   stringr_1.4.0
 [45] lifecycle_1.0.1             restfulr_0.0.13
 [47] XML_3.99-0.8                zlibbioc_1.39.0
 [49] scales_1.1.1                BiocStyle_2.21.4
 [51] hms_1.1.1                   MatrixGenerics_1.5.4
 [53] RBGL_1.69.0                 parallel_4.1.1
 [55] SummarizedExperiment_1.23.5 gRbase_1.8-6.7
 [57] fields_12.5                 yaml_2.2.1
 [59] curl_4.3.2                  memoise_2.0.0
 [61] gridExtra_2.3               ggplot2_3.3.5
 [63] biomaRt_2.49.7              stringi_1.7.5
 [65] RSQLite_2.2.8               BiocIO_1.3.0
 [67] checkmate_2.0.0             GenomicFeatures_1.45.2
 [69] filelock_1.0.2              BiocParallel_1.27.17
 [71] GenomeInfoDb_1.29.10        rlang_0.4.12
 [73] pkgconfig_2.0.3             bitops_1.0-7
 [75] matrixStats_0.61.0          evaluate_0.14
 [77] lattice_0.20-45             purrr_0.3.4
 [79] GenomicAlignments_1.29.0    bit_4.0.4
 [81] tidyselect_1.1.1            magrittr_2.0.1
 [83] R6_2.5.1                    generics_0.1.0
 [85] DelayedArray_0.19.4         DBI_1.1.1
 [87] pillar_1.6.4                KEGGREST_1.33.0
```

```
 [89] RCurl_1.98-1.5             tibble_3.1.5
 [91] crayon_1.4.1               R.rsp_0.44.0
 [93] utf8_1.2.2                 BiocFileCache_2.1.1
 [95] rmarkdown_2.11             viridis_0.6.2
 [97] progress_1.2.2             grid_4.1.1
 [99] blob_1.2.2                 Rgraphviz_2.37.2
[101] digest_0.6.28              qpgraph_2.27.0
[103] xtable_1.8-4               R.cache_0.15.0
[105] R.utils_2.11.0             munsell_0.5.0
[107] viridisLite_0.4.0
```

References

[1] Ogata H, Goto S, Sato K, Fujibuchi W, Bono H, Kanehisa M. KEGG: Kyoto Encyclopedia of Genes and Genomes. Nucleic Acids Res. 1999 Jan 1;27(1):29-34.

[2] Matthews L, Gopinath G, Gillespie M, Caudy M, Croft D, de Bono B, Garapati P, Hemish J, Hermjakob H, Jassal B, Kanapin A, Lewis S, Mahajan S, May B, Schmidt E, Vastrik I, Wu G, Birney E, Stein L, D'Eustachio P. Reactome knowledgebase of human biological pathways and processes. Nucleic Acids Res. 2009 Jan;37(Database issue):D619-22. Epub 2008 Nov 3.

[3] Schaefer CF, Anthony K, Krupa S, Buchoff J, Day M, Hannay T, Buetow KH. PID: the Pathway Interaction Database. Nucleic Acids Res. 2009 Jan;37(Database issue):D674-9. Epub 2008 Oct 2.

[4] Draghici, S., Khatri, P., Tarca, A.L., Amin, K., Done, A., Voichita, C., Georgescu, C., Romero, R. A systems biology approach for pathway level analysis. Genome Research, 17, 2007.

[5] Tarca AL, Draghici S, Khatri P, Hassan SS, Mittal P, Kim JS, Kim CJ, Kusanovic JP, Romero R. A novel signaling pathway impact analysis. Bioinformatics. 2009 Jan 1;25(1):75-82.

[6] Adi L. Tarca, Sorin Draghici, Purvesh Khatri, et. al. A Signaling Pathway Impact Analysis for Microarray Experiments. Bioinformatics, 2009, 25(1):75-82.

[7] Massa MS, Chiogna M, Romualdi C. Gene set analysis exploiting the topology of a pathway. BMC System Biol. 2010 Sep 1;4:121.

[8] Martini P, Sales G, Massa MS, Chiogna M, Romualdi C. Along signal paths: an empirical gene set approach exploiting pathway topology. Nucleic Acids Res. 2013 Jan 7;41(1):e19. doi: 10.1093/nar/gks866. Epub 2012 Sep 21.

[9] Caspi R, Altman T, Dale JM, Dreher K, Fulcher CA, Gilham F, Kaipa P, Karthikeyan AS, Kothari A, Krummenacker M, Latendresse M, Mueller LA, Paley S, Popescu L, Pujar A, Shearer AG, Zhang P, Karp PD. The MetaCyc database of metabolic pathways and enzymes and the BioCyc collection of pathway/genome databases. Nucleic Acids Research 38:D473-9 2010.

[10] PANTHER in 2013: modeling the evolution of gene function, and other gene attributes, in the context of phylogenetic trees. Huaiyu Mi, Anushya Muruganujan and Paul D. Thomas Nucl. Acids Res. (2012) doi: 10.1093/nar/gks1118

[11] Whirl-Carrillo M, McDonagh EM, Hebert JM, Gong L, Sangkuhl K, Thorn CF, Altman RB, Klein TE. Pharmacogenomics knowledge for personalized medicine. Clin Pharmacol Ther. 2012 Oct;92(4):414-7. doi: 10.1038/clpt.2012.96. Review. PubMed

[12] Jewison T, Su Y, Disfany FM, Liang Y, Knox C, Maciejewski A, Poelzer J, Huynh J, Zhou Y, Arndt D, Djoumbou Y, Liu Y, Deng L, Guo AC, Han B, Pon A, Wilson M, Rafatnia S, Liu P, Wishart DS. SMPDB 2.0: big improvements to the Small Molecule Pathway Database. Nucleic Acids Res. 2014 Jan;42(Database issue):D478-84. doi:10.1093/nar/gkt1067. Epub 2013 Nov 6.

[13] Wishart DS, Li C, Marcu A, Badran H, Pon A, Budinski Z, Patron J, Lipton D, Cao X, Oler E, Li K, Paccoud M, Hong C, Guo AC, Chan C, Wei W, Ramirez-Gaona M. PathBank: a comprehensive pathway database for model organisms. Nucleic Acids Res. 2020 Jan 8;48(D1):D470-D478. doi: 10.1093/nar/gkz861.

[14] Martens M, Ammar A, Riutta A, Waagmeester A, Slenter D, Hanspers K, Miller R, Digles D, Lopes E, Ehrhart F, Dupuis L, Winckers L, Coort S, Willighagen E, Evelo C, Pico A, Kutmon M WikiPathways: connecting communities Nucleic Acids Research, Volume 49, Issue D1, 8 January 2021, Pages D613–D621, doi:10.1093/nar/gkaa1024