OmnipathR: utility functions to work with Omnipath in R

Alberto Valdeolivas *1 **_and Attila Gabor_** †1

[1]Institute of Computational Biomedicine, Heidelberg University, Faculty of Medicine, 69120 Heidelberg, Germany

*alvaldeolivas@gmail.com †gaborattila87@gmail.com

October 29, 2019

Abstract

This vignette describes how to use the *OmnipathR* package to retrieve information from the Omnipath database:

http://omnipathdb.org/

In addition, it includes some utility functions to filter, analyse and visualize the data.

Package

OmnipathR 1.0.0

Feedbacks and bugreports are always very welcomed!

Please use the Github issue page to report bugs or for questions:

https://github.com/saezlab/OmnipathR/issues

Many thanks for using *OmnipathR*!

Contents

1 Introduction

OmnipathR is an *R* package built to provide easy access to the data stored in the Omnipath webservice [1]:

http://omnipathdb.org/

The webservice implements a very simple REST style API. This package make requests by the HTTP protocol to retreive the data. Hence, fast Internet access is required for a proper use of *OmnipathR*.

1.1 Query types

OmnipathR can retrieve five different types of data:

- **Interactions:** protein-protein interactions organized in different datasets:
  - **Omnipath:** the OmniPath data as defined in the original publication [1] and collected from different databases.
  - **Pathwayextra:** activity flow interactions without literature reference.
  - **Kinaseextra:** enzyme-substrate interactions without literature reference.
  - **Ligrecextra:** ligand-receptor interactions without literature reference.
  - **Tfregulons:** transcription factor (TF)-target interactions from DoRothEA [2, 3].
  - **Mirnatarget:** miRNA-mRNA and TF-miRNA interactions.
- **Post-translational modifications (PTMs):** It provides enzyme-substrate reactions in a very similar way to the aforementioned interactions. Some of the biological databases related to PTMs integrated in Omnipath are Phospho.ELM [4] and PhosphoSitePlus [5].
- **Complexes:** it provides access to a comprehensive database of more than 22000 protein complexes. This data comes from different resources such as: CORUM [6] or Hu.map [7].
- **Annotations:** it provides a large variety of data regarding different annotations about proteins and complexes. These data come from dozens of databases covering different topics such as: The Topology Data Bank of Transmembrane Proteins (TOPDB) [8] or ExoCarta [9], a database collecting the proteins that were identified in exosomes in multiple organisms.
- **Intercell:** it provides information on the roles in inter-cellular signaling. For instance. if a protein is a ligand, a receptor, an extracellular matrix (ECM) component, etc. The data does not come from original sources but combined from several databases by us. The source databases, such as CellPhoneDB [10] or Receptome [11], are also referred for each reacord.

Figure 1 shows an overview of the resources featured in OmniPath. For more detailed information about the original data sources integrated in Omnipath, please visit: http://omnipathdb.org/ and http://omnipathdb.org/info.

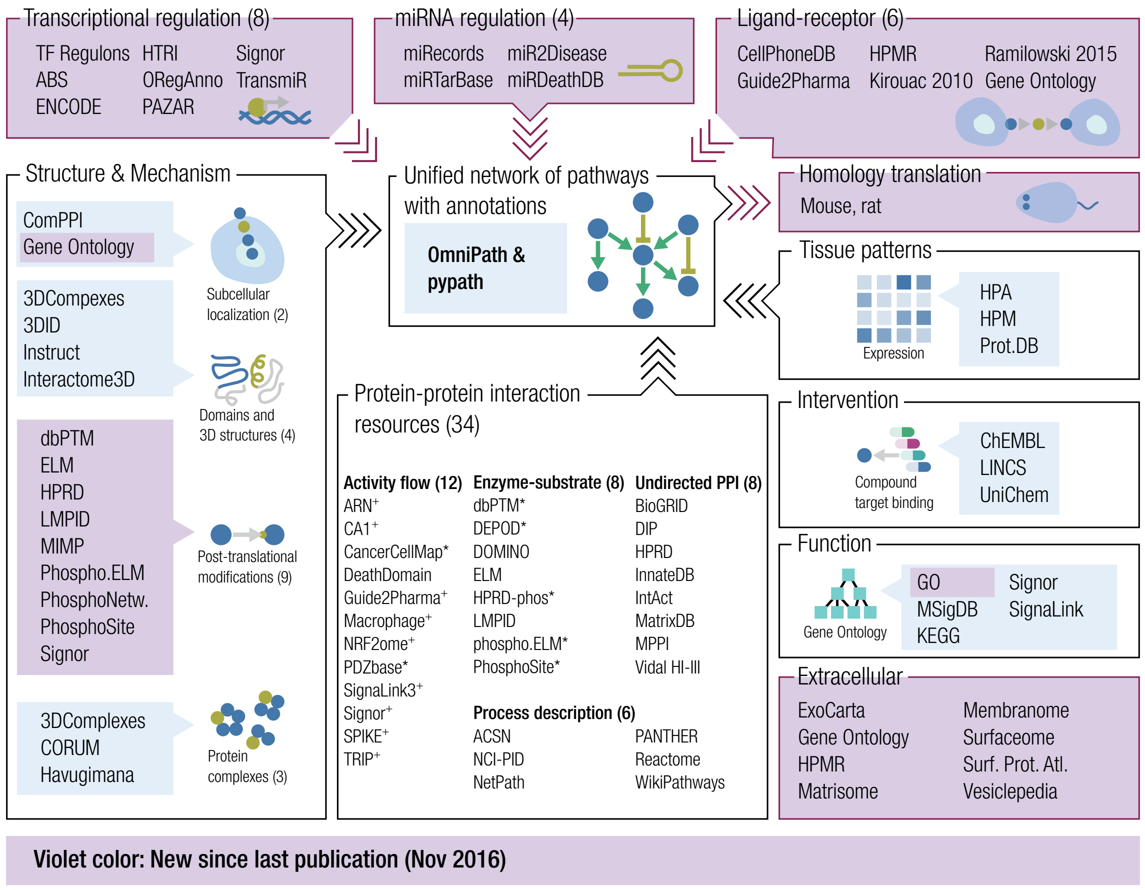

Figure 1: Overview of the resources featured in OmniPath
Causal resources (including activity-flow and enzyme-substrate resources) can provide direction (*) or sign and direction (+) of interactions.

1.2 Mouse and rat

Excluding the miRNA interactions, all interactions and PTMs are available for human, mouse and rat. The rodent data has been translated from human using the NCBI Homologene database. Many human proteins do not have known homolog in rodents hence rodent datasets are smaller than their human counterparts.

In case you work with mouse omics data you might do better to translate your dataset to human (for example using the pypath.homology module, https://github.com/saezlab/pypath/) and use human interaction data.

2 Installation of the *OmnipathR* package

First of all, you need a current version of *R* (www.r-project.org). *OmnipathR* is a freely available package deposited on http://bioconductor.org/ and https://github.com/saezlab/OmnipathR. You can install it by running the following commands on an *R* console:

```
> if (!requireNamespace("BiocManager", quietly = TRUE))
+     install.packages("BiocManager")
> BiocManager::install("OmnipathR")
```

3 Usage Examples

In the following paragraphs, we provide some examples to describe how to use the *OmnipathR* package to retrieve different types of information from Omnipath webserver. In addition, we play around with the data aiming at obtaining some biological relevant information.

Noteworthy, the sections **complexes**, **annotations** and **intercell** are linked. We explore the annotations and roles in inter-cellular communications of the proteins involved in a given complex. This basic example shows the usefulness of integrating the information avaiable in the different **Omnipath** resources.

3.1 Interactions

Proteins interact among them and with other biological molecules to perform cellular functions. Proteins also participates in pathways, linked series of reactions occurring inter/intra cells to transform products or to transmit signals inducing specific cellular responses. Protein interactions are therefore a very valuable source of information to understand cellular functioning.

We are going to download the original **Omnipath** human interactions [1]. To do so, we first check the different source databases and select some of them. Then, we print some of the downloaded interactions ("+" means activation, "-" means inhibition and "?" means undirected interactions or inconclusive data).

```
> library(OmnipathR)
> library(tidyr)
> library(dnet)
> library(gprofiler2)
> ## We check some of the different interaction databases
> head(get_interaction_databases(),10)

 [1] "Wang"                 "BioGRID"              "ACSN"
 [4] "IntAct"               "MIMP"                 "PhosphoELM_MIMP"
 [7] "PhosphoSitePlus_MIMP" "HPRD"                 "InnateDB"
[10] "PhosphoPoint"

> ## The interactions are stored into a data frame.
> interactions <-
+     import_Omnipath_Interactions(filter_databases=c("SignaLink3","PhosphoSite",
+     "Signor"))
> ## We visualize the first interactions in the data frame.
> print_interactions(head(interactions))

            source interaction        target nsources nrefs
31 CTNND1 (O60716)  ==( + )==> CDH1 (P12830)        7    27
5    EGFR (P00533)  ==( + )==> CDH1 (P12830)        8     7
```

```
24  CASP3 (P42574)  ==( ? )==> CDH1 (P12830)        6     2
20  CTBP1 (Q13363)  ==(+/-)==> CDH1 (P12830)        3     1
26   CDH1 (P12830)  ==( + )==> LRP6 (O75581)        2     1
27 ADAM10 (O14672)  ==( + )==> CDH1 (P12830)        2     1
```

3.1.1 Protein-protein interaction networks

Protein-protein interactions are usually converted into networks. Describing protein interactions as networks not only provides a convenient format for visualization, but also allows applying graph theory methods to mine the biological information they contain.

We convert here our set of interactions to a network/graph (*igraph* object). Then, we apply two very common approaches to extract information from a biological network:

- **Shortest Paths:** finding a path between two nodes (proteins) going through the minimum number of edges. This can be very useful to track consecutive reactions within a given pathway. We display below the shortest path between two given proteins and all the possible shortests paths between two other proteins. It is to note that the functions *printPath_es* and *printPath_vs* display very similar results, but the first one takes as an input an edge sequence and the second one a node sequence.

```
> ## We transform the interactions data frame into a graph
> OPI_g <- interaction_graph(interactions = interactions)
> ## Find and print shortest paths on the directed network between proteins
> ## of interest:
> printPath_es(shortest_paths(OPI_g,from = "TYRO3",to = "STAT3",
+     output = 'epath')$epath[[1]],OPI_g)

          source interaction         target nsources nrefs
1 TYRO3 (Q06418)  ==( + )==>  GRB2 (P62993)        1     1
2  GRB2 (P62993)  ==( + )==>  EGFR (P00533)       11    63
3  EGFR (P00533)  ==( + )==> STAT3 (P40763)       10    21

> ## Find and print all shortest paths between proteins of interest:
> printPath_vs(all_shortest_paths(OPI_g,from = "DYRK2",
+     to = "MAPKAPK2")$res,OPI_g)
```

- **Clustering:** grouping nodes (proteins) in such a way that nodes belonging to the same group (called cluster) are more connected in the network to each other than to those in other groups (clusters). Since proteins interact to perform their functions, proteins within the same cluster are likely to be implicated in similar biological tasks. Figure 2 shows the subgraph containing the proteins and interactions of a specifc protein. The *igraph* *R* package contains functions to apply sevaral different cluster methods on graphs (visit https://igraph.org/r/doc/ for detailed information.)

```
> ## We apply a clustering algorithm (Louvain) to group proteins in
> ## our network. We apply here Louvain which is fast but can only run
> ## on undirected graphs. Other clustering algorithms can deal with
> ## directed networks but with longer computational times,
> ## such as cluster_edge_betweenness. These cluster methods are directly
> ## available in the igraph package.
> OPI_g_undirected <- as.undirected(OPI_g, mode=c("mutual"))
> cl_results <- cluster_louvain(OPI_g_undirected)
> ## We extract the cluster where a protein of interest is contained
> cluster_id <- cl_results$membership[which(cl_results$names == "CD22")]
> module_graph <- induced_subgraph(OPI_g_undirected,
+     V(OPI_g)$name[which(cl_results$membership == cluster_id)])
```

```
> ## We print that cluster with its interactions.
> par(mar=c(0.1,0.1,0.1,0.1))
> plot(module_graph, vertex.label.color="black",vertex.frame.color="#ffffff",
+     vertex.size= 15, edge.curved=.2,
+     vertex.color = ifelse(igraph::V(module_graph)$name == "CD22","yellow",
+     "#00CCFF"), edge.color="blue",edge.width=0.8)
```

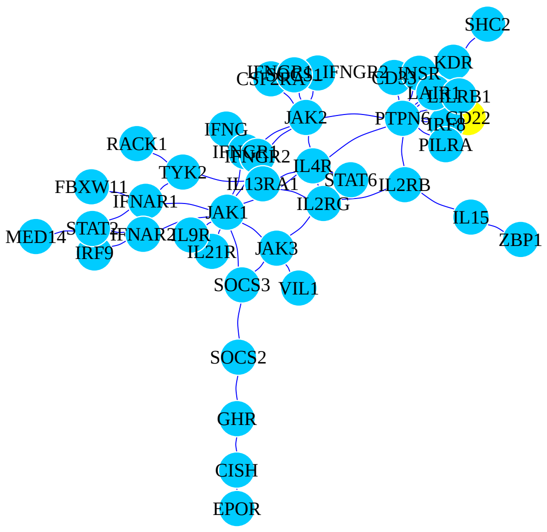

Figure 2:
Subnetwork extracted from the interactions graph representing the cluster where we can find the gene *CD22* (yellow node).

3.1.2 Other interaction datasets

We used above the interactions from the dataset described in the original **Omnipath** publication [1]. In this section, we provide examples on how to retry and deal with interactions from the remaining datasets. The same functions can been applied to every interaction dataset.

In the first example, we are going to get the interactions from the **pathwayextra** dataset, which contains activity flow interactions without literature reference. We are going to focus on the mouse interactions for a given gene in this particular case.

```
> ## We query and store the interactions into a dataframe
> interactions <-
+     import_PathwayExtra_Interactions(filter_databases=c("BioGRID","IntAct"),
+     select_organism = 10090)
> ## We select all the interactions in which Amfr gene is involved
> interactions_Amfr <- dplyr::filter(interactions, source_genesymbol == "Amfr" |
+     target_genesymbol == "Amfr")
> ## We print these interactions:
> print_interactions(interactions_Amfr)

         source interaction        target nsources
3  Gpi (P06745)  ==( + )==> Amfr (Q9R049)        6
1 Amfr (Q9R049)  ==( + )==>  Vcp (Q01853)        5
2 Amfr (Q9R049)  ==( - )==> Sod1 (P08228)        2
```

Next, we download the interactions from the **kinaseextra** dataset, which contains enzyme-substrate interactions without literature reference. We are going to focus on rat reactions targeting a particular gene.

```
> ## We query and store the interactions into a dataframe
> interactions <-
+     import_KinaseExtra_Interactions(filter_databases=c("PhosphoPoint",
+     "PhosphoSite"), select_organism = 10116)
> ## We select the interactions in which Dpysl2 gene is a target
> interactions_TargetDpysl2 <- dplyr::filter(interactions,
+     target_genesymbol == "Dpysl2")
> ## We print these interactions:
> print_interactions(interactions_TargetDpysl2)

          source interaction          target nsources
2 Gsk3b (P18266)  ==(+/-)==> Dpysl2 (P47942)       14
3 Rock2 (Q62868)  ==( + )==> Dpysl2 (P47942)       11
1  Cdk5 (Q03114)  ==( + )==> Dpysl2 (P47942)        8
4 Rock1 (Q63644)  ==( ? )==> Dpysl2 (P47942)        6
5   Fer (P09760)  ==( ? )==> Dpysl2 (P47942)        3
```

In the following example we are going to work with the **ligrecextra** dataset, which contains ligand-receptor interactions without literature reference. Our goal is to find the shortest path between two proteins of our interest. For a more global overview, we induce a network containing the genes involved in our shortest path and their first neighbors (Figure 3).

```
> ## We query and store the interactions into a dataframe
> interactions <- import_LigrecExtra_Interactions(filter_databases=c("HPRD",
+     "Guide2Pharma"),select_organism=9606)
> ## We transform the interactions data frame into a graph
> OPI_g <- interaction_graph(interactions = interactions)
> ## We aim at finding the shortest path between two genes of interest.
> path <- shortest_paths(OPI_g, "B2M", "TFR2")
> printPath_vs(path$vpath,OPI_g)
> ## We induce a network with the genes involved in the shortest path and their
> ## first neighbors to get a more general overview of the interactions
> Induced_Network <-  dNetInduce(g=OPI_g,
+     nodes_query=as.character(path$vpath[[1]]$name), knn=1,
+     remove.loops=FALSE, largest.comp=FALSE)
>
```

```
> ## We print the induced network
> par(mar=c(0.1,0.1,0.1,0.1))
> plot(Induced_Network, vertex.label.color="black",
+     vertex.frame.color="#ffffff",vertex.size= 20, edge.curved=.2,
+     vertex.color =
+         ifelse(igraph::V(Induced_Network)$name %in% c("B2M","TFR2"),
+         "yellow","#00CCFF"), edge.color="blue",edge.width=0.8)
```

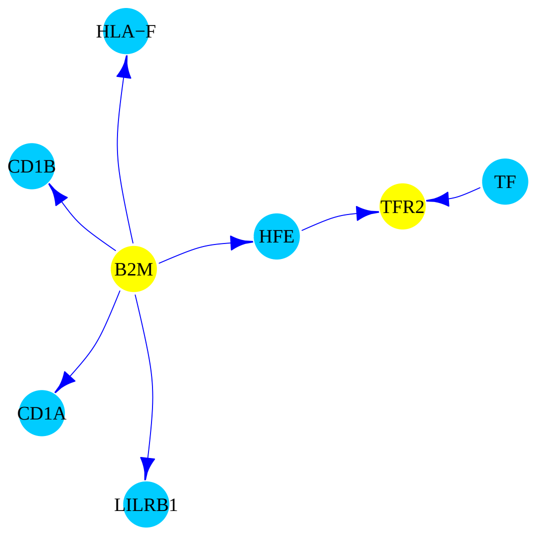

Figure 3:
Subnetwork extracted from the **kinaseextra** interactions graph containing the shortest path between *B2M* and *TFR2* (yellow nodes). The first neighbors of the genes involved in the shortest path are also shown.

Another very interesting interaction dataset also available in Omnipath are the **tfregulons** from DoRothEA [2, 3]. It contains transcription factor (TF)-target interactions with confidence score, ranging from A-E, being A the most confident interactions. In the code chunk shown below, we select and print the most confident interactions for a given TF.

```
> ## We query and store the interactions into a dataframe
> interactions <- import_TFregulons_Interactions(filter_databases=c("tfact",
+     "ARACNe-GTEx"),select_organism=9606)
> ## We select the most confident interactions for a given TF and we print
> ## the interactions to check the way it regulates its different targets
> interactions_A_GLI1  <- dplyr::filter(interactions, tfregulons_level=="A",
+     source_genesymbol == "GLI1")
> print_interactions(interactions_A_GLI1)

         source interaction          target nsources
3 GLI1 (P08151)  ==( + )==>  PTCH1 (Q13635)        3
1 GLI1 (P08151)  ==( + )==> IGFBP6 (P24592)        2
2 GLI1 (P08151)  ==( - )==>  SLIT2 (O94813)        2
```

The last dataset describing interactions is **mirnatarget**. It stores miRNA-mRNA and TF-miRNA interactions. These interactions are only available for human so far. We next select the miRNA interacting with the TF selected in the previous code chunk, *GLI1*. The main function of miRNAs seems to be related with gene regulation. It is therefore interesting to see how some miRNA can regulate the expression of a TF which in turn regulates the expression of other genes. Figure 4 shows a schematic network of the miRNA targeting *GLI1* and the genes regulated by this TF.

```
> ## We query and store the interactions into a dataframe
> interactions <-
+   import_miRNAtarget_Interactions(filter_databases=c("miRTarBase","miRecords"))
> ## We select the interactions where a miRNA is interacting with the TF
> ## used in the previous code chunk and we print these interactions.
> interactions_miRNA_GLI1 <-
+     dplyr::filter(interactions,  target_genesymbol == "GLI1")
> print_interactions(interactions_miRNA_GLI1)

                          source interaction        target nsources nrefs
1  hsa-miR-324-5p (MIMAT0000761)  ==( ? )==> GLI1 (P08151)        3     2
2     hsa-miR-326 (MIMAT0000756)  ==( ? )==> GLI1 (P08151)        2     1
3 hsa-miR-125b-5p (MIMAT0000423)  ==( ? )==> GLI1 (P08151)        2     1
4    hsa-miR-133b (MIMAT0000770)  ==( ? )==> GLI1 (P08151)        1     1
5     hsa-miR-202 (MIMAT0002811)  ==( ? )==> GLI1 (P08151)        1     1

> ## We transform the previous selections to graphs (igraph objects)
> OPI_g_1 <-interaction_graph(interactions = interactions_A_GLI1)
> OPI_g_2 <-interaction_graph(interactions = interactions_miRNA_GLI1)
```

```
> ## We print the union of both previous graphs
> par(mar=c(0.1,0.1,0.1,0.1))
> plot(OPI_g_1 %u% OPI_g_2, vertex.label.color="black",
+     vertex.frame.color="#ffffff",vertex.size= 20, edge.curved=.25,
+     vertex.color = ifelse(grepl("miR",igraph::V(OPI_g_1 %u% OPI_g_2)$name),
+     "red",ifelse(igraph::V(OPI_g_1 %u% OPI_g_2)$name == "GLI1",
+     "yellow","#00CCFF")), edge.color="blue",
+     vertex.shape = ifelse(grepl("miR",igraph::V(OPI_g_1 %u% OPI_g_2)$name),
+     "vrectangle","circle"),edge.width=0.8)
```

3.2 Post-translational modifications (PTMs)

Another query type available is PTMs which provides enzyme-substrate reactions in a very similar way to the aforementioned interactions. PTMs refer generally to enzymatic modification of proteins after their synthesis in the ribosomes. PTMs can be highly context-specific and they play a main role in the activation/inhibition of biological pathways.

In the next code chunk, we download the **PTMs** for human. We first check the different available source databases, even though we do not perform any filter. Then, we select and print the reactions involving a specific enzyme-substrate pair. Those reactions lack information about activation or inhibition. To obtain that information, we match the data with **Omnipath** interactions. Finally, we show that it is also possible to build a graph using this information, and to retrieve PTMs from mouse or rat.

```
> ## We check the different PTMs databases
> get_ptms_databases()
```

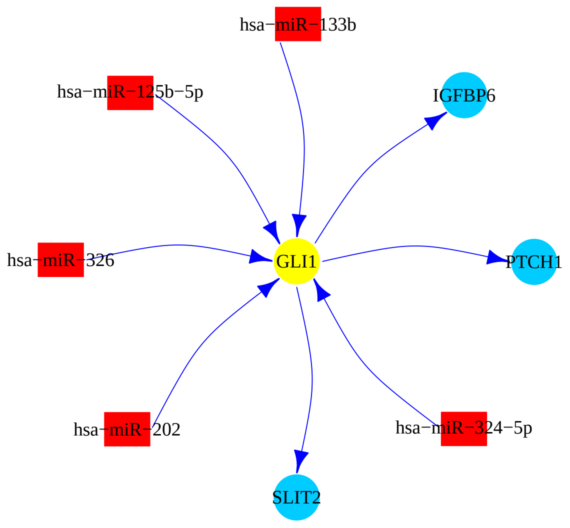

Figure 4:
Schematic network of the miRNA (red square nodes) targeting *GLI1* (yellow node) and the genes regulated by this TF (blue round nodes).

```
[1] "MIMP"              "PhosphoSite"     "Signor"          "phosphoELM"
[5] "HPRD"              "PhosphoNetworks" "dbPTM"           "Li2012"
[9] "DEPOD"

> ## We query and store the ptms into a dataframe. No filtering by
> ## databases in this case.
> ptms <- import_Omnipath_PTMS()
> ## We can select and print the reactions between a specific kinase and
> ## a specific substrate
> print_interactions(dplyr::filter(ptms,enzyme_genesymbol=="MAP2K1",
+     substrate_genesymbol=="MAPK3"))

           enzyme interaction           substrate    modification nsources
4 MAP2K1 (Q02750)       ====> MAPK3_Y204 (P27361) phosphorylation        6
3 MAP2K1 (Q02750)       ====> MAPK3_T202 (P27361) phosphorylation        6
1 MAP2K1 (Q02750)       ====> MAPK3_Y210 (P27361) phosphorylation        1
2 MAP2K1 (Q02750)       ====> MAPK3_T207 (P27361) phosphorylation        1
5 MAP2K1 (Q02750)       ====>  MAPK3_T80 (P27361) phosphorylation        1
6 MAP2K1 (Q02750)       ====> MAPK3_Y222 (P27361) phosphorylation        1

> ## In the previous results, we can see that ptms does not contain sign
> ## (activation/inhibition). We can generate this information based on the
> ## protein-protein Omnipath interaction dataset.
> interactions <- import_Omnipath_Interactions()
> ptms <- get_signed_ptms(ptms,interactions)
> ## We select again the same kinase and substrate. Now we have information
> ## about inhibition or activation when we print the ptms
> print_interactions(dplyr::filter(ptms,enzyme_genesymbol=="MAP2K1",
+     substrate_genesymbol=="MAPK3"))
```

```
           enzyme interaction           substrate    modification nsources
5 MAP2K1 (Q02750)  ==( + )==> MAPK3_Y204 (P27361) phosphorylation        6
6 MAP2K1 (Q02750)  ==( + )==> MAPK3_T202 (P27361) phosphorylation        6
1 MAP2K1 (Q02750)  ==( + )==> MAPK3_Y210 (P27361) phosphorylation        1
2 MAP2K1 (Q02750)  ==( + )==> MAPK3_Y222 (P27361) phosphorylation        1
3 MAP2K1 (Q02750)  ==( + )==> MAPK3_T207 (P27361) phosphorylation        1
4 MAP2K1 (Q02750)  ==( + )==>  MAPK3_T80 (P27361) phosphorylation        1

> ## We can also transform the ptms into a graph.
> ptms_g <- ptms_graph(ptms = ptms)
> ## We download PTMs for mouse
> ptms <- import_Omnipath_PTMS(filter_databases=c("PhosphoSite", "Signor"),
+     select_organism=10090)
```

3.3 Complexes

Some studies indicate that around 80% of the human proteins operate in complexes, and many proteins belong to several different complexes [12]. These complexes play critical roles in a large variety of biological processes. Some well-known examples are the proteasome and the ribosome. Thus, the description of the full set of protein complexes functioning in cells is essential to improve our understanding of biological processes.

The **complexes** query provides access to more than 20000 protein complexes. This comprehensive database has been created by integrating different resources. We now download these molecular complexes filtering by some of the source databases. We check the complexes where a couple of specific genes participate. First, we look for the complexes where any of these two genes participate. We then identify the complex where these two genes are jointly involved. Finally, we perform an enrichment analysis with the genes taking part in that complex. You should keep an eye on this complex since it will be used again in the forthcoming sections.

```
> ## We check the different complexes databases
> get_complexes_databases()

 [1] "hu.MAP"         "PDB"            "Signor"         "Compleat"
 [5] "CORUM"          "ComplexPortal"  "CellPhoneDB"    "Havugimana2012"
 [9] "HPMR"           "Guide2Pharma"   "CFinder"        "NetworkBlast"

> ## We query and store complexes from some sources into a dataframe.
> complexes <- import_Omnipath_complexes(filter_databases=c("CORUM", "hu.MAP"))
> ## We check all the molecular complexes where a set of genes participate
> query_genes <- c("LMNA","CTCF")
> ## Complexes where any of the input genes participate
> complexes_query_genes_any <- get_complex_genes(complexes,query_genes,
+     total_match=FALSE)
> ## We print the components of the different selected components
> complexes_query_genes_any$components_genesymbols

[1] "CTCF_H2AFZ_HIST2H2AA3_KPNA1_KPNA3_LMNA_NPM1_PARP1_TOP2A"
[2] "CTCF_SMAD3_SMAD4"
[3] "CTCF_SET"
[4] "CTCF_NPM1"
```

```
[5] "ACTB_EMD_LMNA_LMNB1_NMI_SPTAN1"
[6] "BANF2_C1QBP_EMD_HIST1H1A_HIST1H3E_HNRNPU_LMNA_LMNB1_MCM2_MCM4_MCM6_NMI_RB1_RBL2_SAP130"
[7] "LMNA_NFYA"
[8] "LAMC3_LMNA"
[9] "BCAS2_CDC5L_EIF2S2_LMNA_MCM2_PDS5A_PDS5B_PLRG1_PRPF19_SAFB_SMC1A_TOP2A"

> ## Complexes where all the input genes participate jointly
> complexes_query_genes_join <- get_complex_genes(complexes,query_genes,
+     total_match=TRUE)
> ## We print the components of the different selected components
> complexes_query_genes_join$components_genesymbols

[1] "CTCF_H2AFZ_HIST2H2AA3_KPNA1_KPNA3_LMNA_NPM1_PARP1_TOP2A"
```

```
> genes_complex <-
+   unlist(strsplit(complexes_query_genes_join$components_genesymbols, "_"))
> ## We can perform an enrichment analyses with the genes in the complex
> EnrichmentResults <- gost(genes_complex, significant = TRUE,
+     user_threshold = 0.001, correction_method = c("fdr"),
+     sources=c("GO:BP","GO:CC","GO:MF"))
> ## We show the most significant results
> EnrichmentResults$result %>%
+   dplyr::select(term_id, source, term_name,p_value) %>%
+   dplyr::arrange(source, p_value)

      term_id source                     term_name      p_value
1  GO:0018995  GO:CC       host cellular component 0.0001989654
2  GO:0043657  GO:CC                     host cell 0.0001989654
3  GO:0031981  GO:CC                 nuclear lumen 0.0004314515
4  GO:0005635  GO:CC              nuclear envelope 0.0005881088
5  GO:0044428  GO:CC                  nuclear part 0.0005881088
6  GO:0005654  GO:CC                   nucleoplasm 0.0008604777
7  GO:0031974  GO:CC       membrane-enclosed lumen 0.0008604777
8  GO:0032993  GO:CC           protein-DNA complex 0.0008604777
9  GO:0043233  GO:CC               organelle lumen 0.0008604777
10 GO:0070013  GO:CC intracellular organelle lumen 0.0008604777
```

3.4 Annotations

Biological annotations are statements, usually traceable and curated, about the different features of a biological entity. At the genetic level, annotations describe the biological function, the subcellular situation, the DNA location and many other related properties of a particular gene or its gene products.

The annotations query provides a large variety of data about proteins and complexes. These data come from dozens of databases and each kind of annotation record contains different fields. Because of this, here we have a record_id field which is unique within the records of each database. Each row contains one key value pair and you need to use the record_id to connect the related key-value pairs (see examples below).

Now, we focus in the annotations of the complex studied in the previous section. We first inspect the different available databases in the omnipath webserver. Then, we download the annotations for our complex itself as a biological entity. We find annotations related to the nucleus and transcriptional control, which is in agreement with the enrichment analysis results of its individual components.

```
> ## We check the different annotation databases
> get_annotation_databases()

 [1] "CPAD"               "CSPA"                "CancerSEA"
 [4] "CellPhoneDB"        "Exocarta"            "GO_Intercell"
 [7] "Guide2Pharma"       "Matrisome"           "Signor"
[10] "kinase.com"         "DisGeNet"            "HPMR"
[13] "Kirouac2010"        "HPA"                 "Locate"
[16] "Phosphatome"        "SignaLink3"          "TopDB"
[19] "Vesiclepedia"       "Integrins"           "MatrixDB"
[22] "NetPath"            "Zhong2015"           "Adhesome"
[25] "CancerGeneCensus"   "HGNC"                "IntOGen"
[28] "KEGG"               "Membranome"          "OPM"
[31] "Ramilowski2015"     "Ramilowski_location" "Surfaceome"
[34] "TFcensus"           "ComPPI"              "CORUM_Funcat"
[37] "CORUM_GO"           "CellPhoneDB_complex" "DGIdb"
[40] "HPMR_complex"

> ## We can further investigate the features of the complex selected
> ## in the previous section.
>
> ## We first get the annotations of the complex itself:
> annotations <-import_Omnipath_annotations(select_genes=paste0("COMPLEX:",
+   complexes_query_genes_join$components_genesymbols))
> dplyr::select(annotations,source,label,value)

               source    label                                      value
1 Ramilowski_location location                                    nucleus
2              ComPPI location                                    nucleus
3        CORUM_Funcat   funcat                                    nucleus
4        CORUM_Funcat   funcat                     transcriptional control
5            CORUM_GO       go regulation of transcription, DNA-templated
6            CORUM_GO       go                                    nucleus
7            CORUM_GO       go     regulation of RNA biosynthetic process
```

Afterwards, we explore the annotations of the individual components of the complex in some databases. We check the pathways where these proteins are involved. Once again, we also find many nucleus related annotations when checking their cellular location.

```
> ## Then, we explore some annotations of its individual components
>
> ## Pathways where the proteins belong:
> annotations <- import_Omnipath_annotations(select_genes=genes_complex,
+     filter_databases=c("NetPath"))
> dplyr::select(annotations,genesymbol,value)

  genesymbol                                             value
1      PARP1                              Androgen receptor (AR)
```

```
2      PARP1       TNF-related weak inducer of apoptosis (TWEAK)
3      PARP1                                 Oncostatin-M (OSM)
4      PARP1               Corticotropin-releasing hormone (CRH)
5      PARP1                   Tumor necrosis factor (TNF) alpha
6      KPNA1                   Fibroblast growth factor-1 (FGF1)
7       CTCF Transforming growth factor beta (TGF-beta) receptor
8      KPNA3                   Tumor necrosis factor (TNF) alpha
9       LMNA                 Thymic stromal lymphopoietin (TSLP)

> ## Cellular localization of our proteins
> annotations <-import_Omnipath_annotations(select_genes=genes_complex,
+    filter_databases=c("ComPPI"))
> ## Since we have same record_id for some results of our query, we spread
> ## these records across columns
> spread(annotations, label,value) %>%
+     dplyr::arrange(desc(score)) %>%
+     dplyr::top_n(10, score)

   uniprot genesymbol entity_type source record_id location          score
1   P06748       NPM1     protein ComPPI      1283  nucleus  0.99999993088
2   P09874      PARP1     protein ComPPI     11110  nucleus 0.999999887104
3   P11388      TOP2A     protein ComPPI      2557  nucleus 0.999999887104
4   P49711       CTCF     protein ComPPI       821  nucleus    0.999999232
5   P0C0S5      H2AFZ     protein ComPPI     34549  nucleus      0.9999328
6   P02545       LMNA     protein ComPPI      1328  nucleus    0.999884752
7   P52294      KPNA1     protein ComPPI      1136  cytosol       0.999496
8   O00505      KPNA3     protein ComPPI      1156  cytosol        0.99928
9   O00505      KPNA3     protein ComPPI      1154  nucleus        0.99832
10  P02545       LMNA     protein ComPPI      1329  cytosol        0.99832
11  P52294      KPNA1     protein ComPPI      1133  nucleus        0.99832
```

3.5 Intercell

Cells perceive cues from their microenvironment and neighboring cells, and respond accordingly to ensure proper activities and coordination between them. The ensemble of these communication process is called inter-cellular signaling (**intercell**).

Intercell query provides information about the roles of proteins in inter-cellular signaling (e.g. if a protein is a ligand, a receptor, an extracellular matrix (ECM) component, etc.) This query type is very similar to annotations. However, **intercell** data does not come from original sources, but combined from several databases by us into categories (we also refer to the original sources).

We first inspect the different categories available in the Omnipath webserver. Then, we focus again in our previously selected complex and we check its potential roles in inter-cellular signaling. We repeat the analysis with its individual components.

```
> ## We check some of the different intercell categories
> head(get_intercell_categories(),10)

 [1] "receptor_cellphonedb"       "receptor_surfaceome"
 [3] "receptor_go"                "receptor_hpmr"
 [5] "receptor_ramilowski"        "receptor_kirouac"
 [7] "receptor_guide2pharma"      "interleukin_receptors_hgnc"
 [9] "receptor_hgnc"              "receptor_dgidb"

> ## We import the intercell data into a dataframe
> intercell <- import_Omnipath_intercell()
> ## We check the intercell annotations for our previous complex itself
> dplyr::filter(intercell,
+     genesymbol == complexes_query_genes_join$components_genesymbols,
+     mainclass != "") %>%
+     dplyr::select(category,genesymbol, mainclass)

[1] category   genesymbol mainclass
<0 rows> (or 0-length row.names)

> ## We check the intercell annotations for the individual components of
> ## our previous complex. We filter our data to print it in a good format
> dplyr::filter(intercell,genesymbol %in% genes_complex, mainclass!="") %>%
+     dplyr::distinct(genesymbol,mainclass, .keep_all = TRUE) %>%
+     dplyr::select(category, genesymbol, mainclass) %>%
+     dplyr::arrange(genesymbol)

               category genesymbol     mainclass
1  intracellular_locate       CTCF intracellular
2  intracellular_comppi      H2AFZ intracellular
3  intracellular_comppi HIST2H2AA3 intracellular
4  intracellular_locate      KPNA1 intracellular
5     transporter_dgidb      KPNA1   transporter
6  intracellular_locate      KPNA3 intracellular
7  transmembrane_locate      KPNA3 transmembrane
8     transporter_dgidb      KPNA3   transporter
9     cell_surface_cspa       LMNA  cell_surface
10 intracellular_locate       LMNA intracellular
11   extracellular_cspa       LMNA extracellular
12 intracellular_locate       NPM1 intracellular
13 extracellular_comppi       NPM1 extracellular
14 intracellular_locate      PARP1 intracellular
15 extracellular_comppi      PARP1 extracellular
16 intracellular_locate      TOP2A intracellular

> ## We close graphical connections
> while (!is.null(dev.list()))  dev.off()
```

3.6 Conclusion

OmnipathR provides access to the wealth of data stored in the Omnipath webservice http://omnipathdb.org/ from the *R* enviroment. In addition, it contains some utility functions for visualization, filtering and analysis. The main strength of *OmnipathR* is the straightforward transformation of the different Omnipath data into commonly used *R* objects, such as dataframes and graphs. Consequently, it allows an easy integration of the different types of data and a gateway to the vast number of *R* packages dedicated to the analysis and representaiton of biological data. We highlighted these abilities in some of the examples detailed in previous sections of this document.

References

[1] Dénes Türei, Tamás Korcsmáros, and Julio Saez-Rodriguez. OmniPath: guidelines and gateway for literature-curated signaling pathway resources. *Nature Methods*, 13(12):966–967, November 2016. URL: https://doi.org/10.1038/nmeth.4077, doi:10.1038/nmeth.4077.

[2] Luz Garcia-Alonso, Francesco Iorio, Angela Matchan, Nuno Fonseca, Patricia Jaaks, Gareth Peat, Miguel Pignatelli, Fiammetta Falcone, Cyril H. Benes, Ian Dunham, Graham Bignell, Simon S. McDade, Mathew J. Garnett, and Julio Saez-Rodriguez. Transcription factor activities enhance markers of drug sensitivity in cancer. *Cancer Research*, 78(3):769–780, December 2017. URL: https://doi.org/10.1158/0008-5472.can-17-1679, doi:10.1158/0008-5472.can-17-1679.

[3] Luz Garcia-Alonso, Christian H. Holland, Mahmoud M. Ibrahim, Denes Turei, and Julio Saez-Rodriguez. Benchmark and integration of resources for the estimation of human transcription factor activities. *Genome Research*, 29(8):1363–1375, July 2019. URL: https://doi.org/10.1101/gr.240663.118, doi:10.1101/gr.240663.118.

[4] H. Dinkel, C. Chica, A. Via, C. M. Gould, L. J. Jensen, T. J. Gibson, and F. Diella. Phospho.ELM: a database of phosphorylation sites–update 2011. *Nucleic Acids Research*, 39(Database):D261–D267, November 2010. URL: https://doi.org/10.1093/nar/gkq1104, doi:10.1093/nar/gkq1104.

[5] Peter V. Hornbeck, Bin Zhang, Beth Murray, Jon M. Kornhauser, Vaughan Latham, and Elzbieta Skrzypek. PhosphoSitePlus, 2014: mutations, PTMs and recalibrations. *Nucleic Acids Research*, 43(D1):D512–D520, December 2014. URL: https://doi.org/10.1093/nar/gku1267, doi:10.1093/nar/gku1267.

[6] Madalina Giurgiu, Julian Reinhard, Barbara Brauner, Irmtraud Dunger-Kaltenbach, Gisela Fobo, Goar Frishman, Corinna Montrone, and Andreas Ruepp. CORUM: the comprehensive resource of mammalian protein complexes—2019. *Nucleic Acids Research*, 47(D1):D559–D563, October 2018. URL: https://doi.org/10.1093/nar/gky973, doi:10.1093/nar/gky973.

[7] Kevin Drew, Chanjae Lee, Ryan L Huizar, Fan Tu, Blake Borgeson, Claire D McWhite, Yun Ma, John B Wallingford, and Edward M Marcotte. Integration of over 9, 000 mass spectrometry experiments builds a global map of human protein complexes. *Molecular Systems Biology*, 13(6):932, June 2017. URL: https://doi.org/10.15252/msb.20167490, doi:10.15252/msb.20167490.

[8] László Dobson, Tamás Langó, István Reményi, and Gábor E. Tusnády. Expediting topology data gathering for the TOPDB database. *Nucleic Acids Research*, 43(D1):D283–D289, November 2014. URL: https://doi.org/10.1093/nar/gku1119, doi:10.1093/nar/gku1119.

[9] Shivakumar Keerthikumar, David Chisanga, Dinuka Ariyaratne, Haidar Al Saffar, Sushma Anand, Kening Zhao, Monisha Samuel, Mohashin Pathan, Markandeya Jois, Naveen Chilamkurti, Lahiru Gangoda, and Suresh Mathivanan. ExoCarta: A web-based compendium of exosomal cargo. *Journal of Molecular Biology*, 428(4):688–692, February 2016. URL: https://doi.org/10.1016/j.jmb.2015.09.019, doi:10.1016/j.jmb.2015.09.019.

[10] Roser Vento-Tormo, Mirjana Efremova, Rachel A. Botting, Margherita Y. Turco, Miquel Vento-Tormo, Kerstin B. Meyer, Jong-Eun Park, Emily Stephenson, Krzysztof Polański, Angela Goncalves, Lucy Gardner, Staffan Holmqvist, Johan Henriksson, Angela Zou, Andrew M. Sharkey, Ben Millar, Barbara Innes, Laura Wood, Anna Wilbrey-Clark, Rebecca P. Payne, Martin A. Ivarsson, Steve Lisgo, Andrew Filby, David H. Rowitch, Judith N. Bulmer, Gavin J. Wright, Michael J. T. Stubbington, Muzlifah Haniffa, Ashley Moffett, and Sarah A. Teichmann. Single-cell reconstruction of the early maternal–fetal interface in humans. *Nature*, 563(7731):347–353, November 2018. URL: https://doi.org/10.1038/s41586-018-0698-6, doi:10.1038/s41586-018-0698-6.

[11] I. Ben-Shlomo, S. Yu Hsu, R. Rauch, H. W. Kowalski, and A. J. W. Hsueh. Signaling receptome: A genomic and evolutionary perspective of plasma membrane receptors involved in signal transduction. *Science Signaling*, 2003(187):re9–re9, June 2003. URL: https://doi.org/10.1126/stke.2003.187.re9, doi:10.1126/stke.2003.187.re9.

[12] Tord Berggård, Sara Linse, and Peter James. Methods for the detection and analysis of protein–protein interactions. *PROTEOMICS*, 7(16):2833–2842, August 2007. URL: https://doi.org/10.1002/pmic.200700131, doi:10.1002/pmic.200700131.

A Session info

- R version 3.6.1 (2019-07-05), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_US.UTF-8`, `LC_COLLATE=C`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=en_US.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Running under: `Ubuntu 18.04.3 LTS`
- Matrix products: default
- BLAS: `/home/biocbuild/bbs-3.10-bioc/R/lib/libRblas.so`
- LAPACK: `/home/biocbuild/bbs-3.10-bioc/R/lib/libRlapack.so`
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: OmnipathR 1.0.0, dnet 1.1.5, gprofiler2 0.1.6, hexbin 1.27.3, igraph 1.2.4.1, supraHex 1.24.0, tidyr 1.0.0
- Loaded via a namespace (and not attached): BiocGenerics 0.32.0, BiocManager 1.30.9, BiocStyle 2.14.0, MASS 7.3-51.4, Matrix 1.2-17, R6 2.4.0, RCurl 1.95-4.12, Rcpp 1.0.2, Rgraphviz 2.30.0, ape 5.3, assertthat 0.2.1, backports 1.1.5, bitops 1.0-6, colorspace 1.4-1, compiler 3.6.1, crayon 1.3.4, data.table 1.12.6, digest 0.6.22, dplyr 0.8.3, evaluate 0.14, ggplot2 3.2.1, glue 1.3.1, graph 1.64.0, grid 3.6.1, gtable 0.3.0, htmltools 0.4.0, htmlwidgets 1.5.1, httr 1.4.1, jsonlite 1.6, knitr 1.25, lattice 0.20-38, lazyeval 0.2.2, lifecycle 0.1.0, magrittr 1.5, munsell 0.5.0, nlme 3.1-141, parallel 3.6.1, pillar 1.4.2, pkgconfig 2.0.3, plotly 4.9.0, purrr 0.3.3, rlang 0.4.1, rmarkdown 1.16, scales 1.0.0, stats4 3.6.1, tibble 2.1.3, tidyselect 0.2.5, tools 3.6.1, vctrs 0.2.0, viridisLite 0.3.0, xfun 0.10, yaml 2.2.0, zeallot 0.1.0