Analysis of the datasets

Learning from Networks - project

Matteo Meneghin

2025-12-19

Contents

1 Setup and datasets download

Due to the GitHub constraints about file size for some datasets, before you need to download all the datasets.

You can download the `vitagrapgh.tsv` file at https://www.kaggle.com/datasets/gianlucadecarlods/vitagraph/.

Download and save, inside a new directory called `CTD datasets`, these files:

File name	Where to find it
`CTD_chem_gene_ixns.tsv.gz`	https://ctdbase.org/downloads/#cg
`CTD_curated_chemicals_diseases.tsv.gz`	https://ctdbase.org/downloads/#c_cd
`CTD_chemicals_diseases.tsv.gz`	https://ctdbase.org/downloads/#cd
`CTD_curated_genes_diseases.tsv.gz`	https://ctdbase.org/downloads/#c_gd
`CTD_genes_diseases.tsv.gz`	https://ctdbase.org/downloads/#agg_gd

Download and save, inside a new directory called `SNAP datasets`, these files:

File name	Where to find it
ChG-InterDecagon_targets.csv.gz	https://snap.stanford.edu/biodata/datasets/10016/10016-ChG-InterDecagon.html
ChCh-Miner_durgbank-chem-chem.tsv.gz	https://snap.stanford.edu/biodata/datasets/10001/10001-ChCh-Miner.html
ChG-Miner_miner-chem-gene.tsv.gz	https://snap.stanford.edu/biodata/datasets/10002/10002-ChG-Miner.html
ChG-TargetDecagon_targets.csv.gz	https://snap.stanford.edu/biodata/datasets/10015/10015-ChG-TargetDecagon.html
DCh-Miner_miner-disease-chemical.tsv.gz	https://snap.stanford.edu/biodata/datasets/10004/10004-DCh-Miner.html
DG-AssocMiner_miner-disease-gene.tsv.gz	https://snap.stanford.edu/biodata/datasets/10012/10012-DG-AssocMiner.html
DD-Miner_miner-disease-disease.tsv.gz	https://snap.stanford.edu/biodata/datasets/10006/10006-DD-Miner.html
DG-Miner_miner-disease-gene.tsv.gz	https://snap.stanford.edu/biodata/datasets/10020/10020-DG-Miner.html

Setup the working directory in RStudio to the source file location, so we can use relative path.

2 Vitagraph

Is a unique file with all nodes and edges.

Load the dataset.

```
vitaGRAPH <- read.table("vitagraph/vitagraph.tsv", header = TRUE)
head(vitaGRAPH)
```

```
##              head interaction             tail source      type
## 1 Gene::NCBI:2157   GENE_BIND  Gene::NCBI:5264 bioarx Gene-Gene
## 2 Gene::NCBI:2157   GENE_BIND  Gene::NCBI:2158 bioarx Gene-Gene
## 3 Gene::NCBI:2157   GENE_BIND  Gene::NCBI:3309 bioarx Gene-Gene
## 4 Gene::NCBI:2157   GENE_BIND Gene::NCBI:28912 bioarx Gene-Gene
## 5 Gene::NCBI:2157   GENE_BIND   Gene::NCBI:811 bioarx Gene-Gene
## 6 Gene::NCBI:2157   GENE_BIND  Gene::NCBI:2159 bioarx Gene-Gene
```

Analyze the dataset.

```
# Extract all the identifiers from the first and third column
identifiers <- c(vitaGRAPH[, 1], vitaGRAPH[, 3])
# Use a specific regular expression to target TYPE::IDENTIFIER
prefixes_extracted <- sub("^([^:]+::[^:]+):.*", "\\1", identifiers)
# Find the unique values of the extracted prefixes
unique_prefixes <- unique(prefixes_extracted)
length(unique_prefixes)
```

```
## [1] 14
```

```
unique_prefixes
```

```
##  [1] "Gene::NCBI"                  "Compound::PubChem_Compounds"
##  [3] "Compound::molport"           "Compound::zinc"
##  [5] "Compound::CHEMBL"            "Disease::bioarx"
##  [7] "Disease::MESH"               "Compound::CHEBI"
```

```
##  [9] "Disease::DOID"               "Anatomy::UBERON"
## [11] "Gene::drugbank"              "Disease::OMIM"
## [13] "Symptom::MESH"               "SideEffect::umls"
```

The dataset has 14 different types of node, but we will only consider the one related to gene, drug (compound) and disease.

```
valid_nodes_vitagraph <- c("Gene::NCBI","Compound::PubChem_Compounds",
                           "Compound::molport","Compound::zinc",
                           "Compound::CHEMBL","Disease::bioarx",
                           "Disease::MESH", "Compound::CHEBI",
                           "Disease::DOID","Gene::drugbank","Disease::OMIM")
```

3 SNAP

In the case of SNAP and CTD the graphs are in different files, we need to analyze the identifiers of different nodes.

Since the input data has no header, the `col.names` variable, of each SNAP dataset, was manually defined following an analysis of the data types.

```
SNAPChCh <- read.table("SNAP datasets/ChCh-Miner_durgbank-chem-chem.tsv.gz",
                       col.names = c("Drug (Drugbank)", "Drug (Drugbank)"))
head(SNAPChCh)
```

```
##   Drug..Drugbank. Drug..Drugbank..1
## 1         DB00862           DB00966
## 2         DB00575           DB00806
## 3         DB01242           DB08893
## 4         DB01151           DB08883
## 5         DB01235           DB01275
## 6         DB00018           DB00333
```

```
SNAPChG_ID <- read.table("SNAP datasets/ChG-InterDecagon_targets.csv.gz", sep=",",
                         col.names = c("Drug (Pubchem)", "Gene (NCBI)"))
head(SNAPChG_ID)
```

```
##   Drug..Pubchem. Gene..NCBI.
## 1   CID000060752        3757
## 2   CID006918155        2908
## 3   CID103052762        3359
## 4   CID023668479        1230
## 5   CID000028864        1269
## 6   CID000028864      124274
```

```
SNAPChG_M <- read.table("SNAP datasets/ChG-Miner_miner-chem-gene.tsv.gz",
                        col.names = c("Drug (Drugbank)", "Gene (Uniprot)"))
head(SNAPChG_M)
```

```
##   Drug..Drugbank. Gene..Uniprot.
## 1         DB00357         P05108
## 2         DB02721         P00325
## 3         DB00773         P23219
## 4         DB07138         Q16539
## 5         DB08136         P24941
## 6         DB01242         P23975
```

```
SNAPChG_TD <- read.table("SNAP datasets/ChG-TargetDecagon_targets.csv.gz", sep=",",
                         col.names = c("Drug (Pubchem)", "Gene (NCBI)"))
head(SNAPChG_TD)
```

```
##   Drug..Pubchem. Gene..NCBI.
## 1   CID000003488        1559
## 2   CID000003488        8647
## 3   CID000077992        3351
## 4   CID000077992        3350
## 5   CID000077992        3352
## 6   CID000002083        1269
```

```
SNAPDCh <- read.table("SNAP datasets/DCh-Miner_miner-disease-chemical.tsv.gz",
                      col.names = c("Disease (MESH)", "Drug (Drugbank)"))
head(SNAPDCh)
```

```
##   Disease..MESH. Drug..Drugbank.
## 1   MESH:D005923         DB00564
## 2   MESH:D009503         DB01072
## 3   MESH:D016115         DB01759
## 4   MESH:D018476         DB00451
## 5   MESH:C567059         DB00641
## 6   MESH:D010198         DB00481
```

```
SNAPDD <- read.table("SNAP datasets/DD-Miner_miner-disease-disease.tsv.gz",
                     col.names = c("Disease (DOID)", "Disease (DOID)"))
head(SNAPDD)
```

```
##   Disease..DOID. Disease..DOID..1
## 1   DOID:0001816        DOID:1115
## 2   DOID:0002116       DOID:10124
## 3   DOID:0014667           DOID:4
## 4   DOID:0050004       DOID:10400
## 5   DOID:0050012         DOID:934
## 6   DOID:0050013     DOID:0060158
```

```
SNAPDG_AM <- read.table("SNAP datasets/DG-AssocMiner_miner-disease-gene.tsv.gz",
                        col.names = c("Disease (UMLS)", "Disease desc.", "Gene (NCBI)"))
head(SNAPDG_AM)
```

```
##   Disease..UMLS.            Disease.desc. Gene..NCBI.
## 1       C0036095 Salivary Gland Neoplasms        1462
## 2       C0036095 Salivary Gland Neoplasms        1612
## 3       C0036095 Salivary Gland Neoplasms         182
## 4       C0036095 Salivary Gland Neoplasms        2011
## 5       C0036095 Salivary Gland Neoplasms        2019
## 6       C0036095 Salivary Gland Neoplasms        2175
```

```
SNAPDG_M <- read.table("SNAP datasets/DG-Miner_miner-disease-gene.tsv.gz",
                       col.names = c("Disease (MESH)", "Gene (Uniprot)"))
head(SNAPDG_M)
```

```
##   Disease..MESH. Gene..Uniprot.
## 1   MESH:D005756     A0A087WZV0
## 2   MESH:D055370         P11464
## 3   MESH:D007410         Q92945
## 4   MESH:D014062         Q6ISS4
```

```
## 5   MESH:D054549         Q96RU8
## 6   MESH:D009771         O94986
```

3.1 Counting of the number of nodes for type and identifier

To count the number of nodes of a specific type of identifier, the columns with the same header have been unified in a single vector, using `unlist()` if all the dataset is considered in the counting.

```
SNAP_doids <- unique(unlist(SNAPDD))
length(SNAP_doids)
```

```
## [1] 6878
```

```
SNAP_mesh <- unique(c(SNAPDG_M[[1]], SNAPDCh[[1]]))
length(SNAP_mesh)
```

```
## [1] 5677
```

```
SNAP_uniprot <- unique(c(SNAPChG_M[[2]], SNAPDG_M[[2]]))
length(SNAP_uniprot)
```

```
## [1] 18147
```

```
SNAP_ncbi <- unique(c(SNAPChG_ID[[2]],SNAPChG_TD[[2]],SNAPDG_AM[[3]]))
length(SNAP_ncbi)
```

```
## [1] 11759
```

```
SNAP_pubchem <- unique(c(SNAPChG_ID[[1]],SNAPChG_TD[[1]]))
length(SNAP_pubchem)
```

```
## [1] 1774
```

```
SNAP_db <- unique(c(SNAPChG_M[[1]],SNAPDCh[[2]], unlist(SNAPChCh)))
length(SNAP_db)
```

```
## [1] 5520
```

4 CTD

```
library(readr) # library to read .tsv of big size
file_path <- "CTD datasets/CTD_chem_gene_ixns.tsv.gz"

# The value of 28 is harcoded after watching carefully the .tsv file
partial_dataset <- readLines(file_path, n = 28)

header_line <- partial_dataset[28]
clean_header_line <- gsub("^#", "", header_line)
header_names <- unlist(strsplit(clean_header_line, "\t"))

CTD_CG <- read_tsv(
    file_path,
    skip = 29,
    comment = "",
    col_names = header_names # Assign the extracted names
)
```

```
## Rows: 3019050 Columns: 11
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (9):  ChemicalName, ChemicalID, CasRN, GeneSymbol, GeneForms, Organism, ...
## dbl (2): GeneID, OrganismID
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(CTD_CG)
```

```
## # A tibble: 6 x 11
##   ` ChemicalName` ChemicalID CasRN GeneSymbol GeneID GeneForms Organism
##   <chr>           <chr>      <chr> <chr>       <dbl> <chr>     <chr>
## 1 10074-G5        C534883    <NA>  AR            367 protein   Homo sapiens
## 2 10074-G5        C534883    <NA>  AR            367 protein   Homo sapiens
## 3 10074-G5        C534883    <NA>  AR            367 protein   Homo sapiens
## 4 10074-G5        C534883    <NA>  AR            367 protein   Homo sapiens
## 5 10074-G5        C534883    <NA>  EPHB2        2048 protein   Homo sapiens
## 6 10074-G5        C534883    <NA>  EPHB2        2048 protein   Homo sapiens
## # i 4 more variables: OrganismID <dbl>, Interaction <chr>,
## #   InteractionActions <chr>, PubMedIDs <chr>
```

```
file_path <- "CTD datasets/CTD_chemicals_diseases.tsv.gz"

# The value of 28 is harcoded after watching carefully the .tsv file
partial_dataset <- readLines(file_path, n = 28)

header_line <- partial_dataset[28]
clean_header_line <- gsub("^#", "", header_line)
header_names <- unlist(strsplit(clean_header_line, "\t"))

CTD_CD <- read_tsv(
    file_path,
    skip = 29,
    comment = "",
    col_names = header_names
)
```

```
## Rows: 9731180 Columns: 10
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (9):  ChemicalName, ChemicalID, CasRN, DiseaseName, DiseaseID, DirectEvi...
## dbl (1): InferenceScore
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(CTD_CD)
```

```
## # A tibble: 6 x 10
##   ` ChemicalName`          ChemicalID CasRN DiseaseName  DiseaseID DirectEvidence
##   <chr>                    <chr>      <chr> <chr>        <chr>     <chr>
## 1 06-Paris-LA-66 protocol C046983    <NA>  Precursor C~ MESH:D05~ therapeutic
## 2 10074-G5                C534883    <NA>  Adenocarcin~ MESH:D00~ <NA>
## 3 10074-G5                C534883    <NA>  Adenocarcin~ MESH:D00~ <NA>
## 4 10074-G5                C534883    <NA>  Alopecia     MESH:D00~ <NA>
```

```
## 5 10074-G5                C534883    <NA>  Androgen-In~ MESH:D01~ <NA>
## 6 10074-G5                C534883    <NA>  Astrocytoma  MESH:D00~ <NA>
## # i 4 more variables: InferenceGeneSymbol <chr>, InferenceScore <dbl>,
## #   OmimIDs <chr>, PubMedIDs <chr>
```

```
file_path <- "CTD datasets/CTD_curated_chemicals_diseases.tsv.gz"

# The value of 28 is harcoded after watching carefully the .tsv file
partial_dataset <- readLines(file_path, n = 28)

header_line <- partial_dataset[28]
clean_header_line <- gsub("^#", "", header_line)
header_names <- unlist(strsplit(clean_header_line, "\t"))

CTD_CD_cur <- read_tsv(
    file_path,
    skip = 29,
    comment = "",
    col_names = header_names
)
```

```
## Rows: 108440 Columns: 7
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (7):  ChemicalName, ChemicalID, CasRN, DiseaseName, DiseaseID, DirectEvi...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(CTD_CD_cur)
```

```
## # A tibble: 6 x 7
##   ` ChemicalName`          ChemicalID CasRN DiseaseName DiseaseID DirectEvidence
##   <chr>                    <chr>      <chr> <chr>       <chr>     <chr>
## 1 06-Paris-LA-66 protocol  C046983    <NA>  Precursor ~ MESH:D05~ therapeutic
## 2 10,10-bis(4-pyridinylme~ C112297    <NA>  Hyperkines~ MESH:D00~ marker/mechan~
## 3 10,10-bis(4-pyridinylme~ C112297    <NA>  Seizures    MESH:D01~ marker/mechan~
## 4 10,11-dihydro-10-hydrox~ C039775    <NA>  Epilepsy    MESH:D00~ therapeutic
## 5 10,11-dihydroxy-N-n-pro~ C425777    <NA>  Hyperkines~ MESH:D00~ marker/mechan~
## 6 10-hydroxycamptothecin   C028098    6765~ Huntington~ MESH:D00~ therapeutic
## # i 1 more variable: PubMedIDs <chr>
```

```
file_path <- "CTD datasets/CTD_genes_diseases.tsv.gz"

# The value of 28 is harcoded after watching carefully the .tsv file
partial_dataset <- readLines(file_path, n = 28)

header_line <- partial_dataset[28]
clean_header_line <- gsub("^#", "", header_line)
header_names <- unlist(strsplit(clean_header_line, "\t"))

CTD_GD <- read_tsv(
    file_path,
    skip = 29,
    comment = "",
    col_names = header_names
```

```
)
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
## Rows: 121371524 Columns: 9
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (5):  GeneSymbol, DiseaseName, DiseaseID, InferenceChemicalName, PubMedIDs
## dbl (2): GeneID, InferenceScore
## lgl (2): DirectEvidence, OmimIDs
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(CTD_GD)
```

```
## # A tibble: 6 x 9
##   ` GeneSymbol`    GeneID DiseaseName                   DiseaseID DirectEvidence
##   <chr>             <dbl> <chr>                         <chr>     <lgl>
## 1 11-BETA-HSD3  100174880 Abnormalities, Drug-Induced   MESH:D00~ NA
## 2 11-BETA-HSD3  100174880 Amyotrophic Lateral Sclerosis MESH:D00~ NA
## 3 11-BETA-HSD3  100174880 Anemia                        MESH:D00~ NA
## 4 11-BETA-HSD3  100174880 Anemia, Hemolytic             MESH:D00~ NA
## 5 11-BETA-HSD3  100174880 Asthenozoospermia             MESH:D05~ NA
## 6 11-BETA-HSD3  100174880 Birth Weight                  MESH:D00~ NA
## # i 4 more variables: InferenceChemicalName <chr>, InferenceScore <dbl>,
## #   OmimIDs <lgl>, PubMedIDs <chr>
```

```
file_path <- "CTD datasets/CTD_curated_genes_diseases.tsv.gz"

# The value of 28 is harcoded after watching carefully the .tsv file
partial_dataset <- readLines(file_path, n = 28)

header_line <- partial_dataset[28]
clean_header_line <- gsub("^#", "", header_line)
header_names <- unlist(strsplit(clean_header_line, "\t"))

CTD_GD_cur <- read_tsv(
    file_path,
    skip = 29,
    comment = "",
    col_names = header_names
)
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
## Rows: 35429 Columns: 7
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (5):  GeneSymbol, DiseaseName, DiseaseID, DirectEvidence, PubMedIDs
```

```
## dbl (2): GeneID, OmimIDs
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(CTD_GD_cur)
```

```
## # A tibble: 6 x 7
##   ` GeneSymbol` GeneID DiseaseName    DiseaseID DirectEvidence OmimIDs PubMedIDs
##   <chr>          <dbl> <chr>          <chr>     <chr>            <dbl> <chr>
## 1 A              50518 Dermatitis     MESH:D00~ marker/mechan~      NA 32937126
## 2 A              50518 Diabetes Mell~ MESH:D00~ marker/mechan~      NA 1473152|~
## 3 A              50518 Diabetes Mell~ MESH:D00~ marker/mechan~      NA 8146154
## 4 A              50518 Diabetic Neph~ MESH:D00~ marker/mechan~      NA 37769864
## 5 A              50518 Edema          MESH:D00~ marker/mechan~      NA 32937126
## 6 A              50518 Failure to Th~ MESH:D00~ marker/mechan~      NA 32937126
```

4.1 Counting of the number of nodes for type and identifier

```
CTD_ncbi <- unique(c(CTD_GD_cur[[2]],CTD_GD[[2]],CTD_CG[[5]]))
length(CTD_ncbi)
```

```
## [1] 57625
```

```
CTD_mesh_drug <- unique(c(CTD_CG[[1]], CTD_CD[[1]],CTD_CD_cur[[1]]))
length(CTD_mesh_drug)
```

```
## [1] 17964
```

```
CTD_all_disease <- c(
  as.character(CTD_CD_cur[[5]]),
  as.character(CTD_CD[[5]]),
  as.character(CTD_GD_cur[[4]]),
  as.character(CTD_GD[[4]])
)
CTD_all_disease <- CTD_all_disease[!is.na(CTD_all_disease)]

CTD_mesh_disease <- CTD_all_disease[grep("^MESH:", CTD_all_disease)]
CTD_mesh_disease <- length(unique(CTD_mesh_disease))
print(paste("Total unique MESH:", CTD_mesh_disease))
```

```
## [1] "Total unique MESH: 5854"
```

```
CTD_omim <- CTD_all_disease[grep("^OMIM:", CTD_all_disease)]
CTD_omim <- length(unique(CTD_omim))
print(paste("Total unique OMIM:", CTD_omim))
```

```
## [1] "Total unique OMIM: 1440"
```

5 Number of nodes and genes

A problem is that nodes in the different files have different identifiers, first we count them to understand which one were more relevant. The number of nodes, divided by identifiers, in the different datasets van be see in Figure 1.

The number of edges can be seen in Figure 2.

	Type	VitaGraph	ctd	SNAP
DISEASE	*TOTAL (considered)*	4777	7294	12555
	DOID	2390	-	6878
	MESH	2356	5854	5677
	OMIM	31	1440	-
	UMLS	-	-	519
	bioarx	27	-	-
GENE	*TOTAL (considered)*	20844	57625	29906
	NCBI	20844	57625	11759
	Drugbank	62	-	-
	Uniprot	-	-	18147
DRUG	*TOTAL (considered)*	15302	17964	7294
	Pubchem_compounds	15302	-	1774
	MESH	-	17964	-
	drugbank	-	-	5520
	molport	221	-	-
	zinc	53	-	-
	CHEMBL	77	-	-
	CHEBI	176	-	-

Figure 1: Number of nodes by type and identifier

Edge type	**Vitagraph**	**ctd**	**SNAP**
Dis-Dis	543	-	6877
Drug-Drug	1 167 617	-	-
Gene-Gene	1 379 965	-	-
Dis-Gene	94 073	121 371 524 + 35 429 = 121 406 953	21 357+15 509 618 = 15 530 975
Drug-Gene	133 452	3 019 050	131 034+15 138+18 690 = 164 862
Dis-Drug	94 073	9 731 180+35 429 = 976 609	466 656

Figure 2: Number of edges

The numbers in these tables came directly from Vitagraph paper, and for ctd and SNAP we analyze each file, counting the number of lines for edges, and the number of unique values for columns for the number of nodes (seen in previous section).

6 Find unique type of identifiers

We have established a primary identifier for each node type (highlighted in yellow in the Figure 1) and developed a method to convert the data in the red cells to match this format.

6.1 Gene - identifier: NCBI

We will ignore the few entries from drugbank.

6.1.1 From Uniprot to NCBI

We will convert the ones from Uniprot using a R library (https://bioconductor.org/packages/release/bioc/html/UniProt.ws.html) that uses Web Services of the official site of Uniprot (https://www.uniprot.org/). Here an example:

```
library(UniProt.ws)
mapUniProt(
   from='UniProtKB_AC-ID',
   to='GeneID',
   query=c('P05108','P00325')
)
```

```
##     From   To
## 1 P05108 1583
## 2 P00325  125
```

6.2 Disease - identifier: MESH

We will ignore the few entries from bioarx and UMLS.

6.2.1 From DOID to MESH

We will use two different files as 'map' to try to cover more DOID possible.

One file, `doid.obo`, can be found at: https://raw.githubusercontent.com/DiseaseOntology/HumanDiseaseOntology/main/src/ontology/doid.obo

Here is a snippet:

```
[Term]
id: DOID:0001816
name: angiosarcoma
alt_id: DOID:267
alt_id: DOID:4508
def: "A vascular cancer that derives_from the cells that line the walls of
  blood vessels or lymphatic vessels."
  [url:http\://en.wikipedia.org/wiki/Hemangiosarcoma,
  url:https\://en.wikipedia.org/wiki/Angiosarcoma,
  url:https\://ncit.nci.nih.gov/ncitbrowser/ConceptReport.jsp?
  dictionary=NCI_Thesaurus&ns=ncit&code=C3088,
  url:https\://www.ncbi.nlm.nih.gov/pubmed/23327728]
subset: DO_cancer_slim
subset: NCIthesaurus
```

```
synonym: "hemangiosarcoma" EXACT []
xref: ICDO:9120/3
xref: MESH:D006394
xref: NCI:C3088
xref: NCI:C9275
xref: SNOMEDCT_US_2023_03_01:39000009
xref: UMLS_CUI:C0018923
xref: UMLS_CUI:C0854893
is_a: DOID:175 ! vascular cancer
```

By parsing the file we can search for a specific DOID and see if we can find a correspondent MESH, in the line with format `xref: MESH:`.

Since not all DOID have a correspondent MESH, we then used a different file: `cross_references.tsv`, that can be found at: https://raw.githubusercontent.com/natacourby/Disease_ontologies_for_knowledge_graphs/refs/heads/master/data/prepared_ontologies/cross_references.tsv

```
cross_references <- read_tsv("map to MESH datasets/cross_references.tsv")
```

```
## Rows: 21696 Columns: 12
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr (12): preferred_ontology_term, MESH, UMLS, EFO, NCIT, OMIM, DOID, Orphan...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(cross_references)
```

```
## # A tibble: 6 x 12
##   preferred_ontology_term MESH    UMLS    EFO   NCIT  OMIM  DOID  Orphanet HP
##   <chr>                   <chr>   <chr>   <chr> <chr> <chr> <chr> <chr>    <chr>
## 1 MONDO_0020678           D006319 <NA>    EFO_~ NCIT~ <NA>  <NA>  <NA>     HP_0~
## 2 MONDO_0002028           D010554 C00312~ HP_0~ NCIT~ <NA>  DOID~ <NA>     HP_0~
## 3 MONDO_0011122           D009765 C00287~ EFO_~ NCIT~ <NA>  DOID~ <NA>     HP_0~
## 4 MONDO_0021234           D013120 C00379~ EFO_~ NCIT~ <NA>  <NA>  <NA>     <NA>
## 5 MONDO_0021245           D009062 C00266~ EFO_~ NCIT~ <NA>  <NA>  <NA>     <NA>
## 6 MONDO_0015265           D001989 CN1991~ EFO_~ NCIT~ <NA>  DOID~ Orphane~ HP_0~
## # i 3 more variables: MONDO <chr>, `ICD-10` <chr>, label <chr>
```

As we can see, searching for DOID in the 7th column, permit to find the correspondent MESH in column 2.

6.2.2 From OMIM to MESH

Like for DOID, we use the same files. In the first one searching for `xref: MIM:`, and in the second one using as index the 6th column.

6.3 Drug - Identifier: Pubchem_compounds

We will ignore the few entries from molport, zinc, CHEMBL, CHEBI.

6.3.1 From Drugbank to Pubchem_compounds

For MESH and drugbank we will use the Pubchem APIs (https://pubchem.ncbi.nlm.nih.gov/docs/pug-rest#section=URL-based-API), parsing the xml response.

In the base request URL: https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/name/DBcode, as `DBcode` values we will use drugbank IDs for SNAP datasets, in the format `DB<number>`.

Here an example. Request: https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/name/DB00829

This is the (partial) response:

```
<PC-Compounds xmlns="http://www.ncbi.nlm.nih.gov"
  xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
  xs:schemaLocation="http://www.ncbi.nlm.nih.gov
  ftp://ftp.ncbi.nlm.nih.gov/pubchem/specifications/pubchem.xsd"
>
  <PC-Compound>
    <PC-Compound_id>
        <PC-CompoundType>
          <PC-CompoundType_id>
            <PC-CompoundType_id_cid>3016</PC-CompoundType_id_cid>
          </PC-CompoundType_id>
      </PC-CompoundType>
    </PC-Compound_id>
    <PC-Compound_atoms>
    ...
    </PC-Compound_atoms>
    <PC-Compound_bonds>
    ...
    </PC-Compound_bonds>
    <PC-Compound_coords>
    ...
    </PC-Compound_coords>
    <PC-Compound_charge>0</PC-Compound_charge>
    <PC-Compound_props>
    ...
    </PC-Compound_props>
    <PC-Compound_count>
    ...
    </PC-Compound_count>
  </PC-Compound>
</PC-Compounds>
```

From which we can retrieve the `PC-CompoundType_id_cid`.

6.3.2 From MESH to Pubchem_compounds

We can use also here the Pubchem APIs, but not directly with the MESH identifier, but with the names of the chemical present in the `ChemicalName` columns seen before, in the CTD section.

Here an example. Request: https://pubchem.ncbi.nlm.nih.gov/rest/pug/compound/name/1,1,1,2-tetrafluoro-2-chloroethane

This is the (partial) response:

```
<PC-Compounds xmlns="http://www.ncbi.nlm.nih.gov"
 xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
 xs:schemaLocation="http://www.ncbi.nlm.nih.gov
 ftp://ftp.ncbi.nlm.nih.gov/pubchem/specifications/pubchem.xsd"
>
  <PC-Compound>
    <PC-Compound_id>
      <PC-CompoundType>
        <PC-CompoundType_id>
```

```
          <PC-CompoundType_id_cid>17822</PC-CompoundType_id_cid>
        </PC-CompoundType_id>
      </PC-CompoundType>
    </PC-Compound_id>
    <PC-Compound_atoms>
    ...
    </PC-Compound_atoms>
    <PC-Compound_bonds>
    ...
    </PC-Compound_bonds>
    <PC-Compound_stereo>
    ...
    </PC-Compound_stereo>
    <PC-Compound_coords>
    ...
    </PC-Compound_coords>
    <PC-Compound_charge>0</PC-Compound_charge>
    <PC-Compound_props>
    ...
    </PC-Compound_props>
    <PC-Compound_count>
    ...
    </PC-Compound_count>
  </PC-Compound>
</PC-Compounds>
```

From which we can retrieve the `PC-CompoundType_id_cid`.