Create the dataset

Learning from Networks - project

Matteo Meneghin

2025-12-28

Contents

1 The work done in this document

In this files we will create the unique file that will be used in our work. The final file will be a single .csv/.tsv with 2 columns, and each row will be an edge between 2 nodes. The nodes will be only of these types, that we will call 'final types': Gene::NCBI, Drug::Pubchem_Compounds, Disease::MESH.

Iterating one file at time we will extract the nodes already in one of the three final types, and then map the others of a different type to one of the final types.

1.1 from DOID/OMIM to Disease::MESH

We will create a single `data.frame`, taking data from two different files, and then using OMIM and DOID as index to a specific MESH value.

- First file at https://raw.githubusercontent.com/natacourby/Disease_ontologies_for_knowledge_graphs/refs/heads/master/data/prepared_ontologies/cross_references.tsv

It'a a .tsv, easily parsable with `read_tsv`.

```
if (!require("readr")) install.packages("readr")
library(readr)
cross_ref <- read_tsv(
  "https://raw.githubusercontent.com/natacourby/Disease_ontologies_for_knowledge_graphs/refs/heads/maste
  show_col_types = FALSE
)
MESH_cross_ref <- cross_ref[,c("MESH", "DOID", "OMIM")]

# Removes rows in which MESH=NA, or both DOID and OMIM are NA
MESH_cross_ref <- subset(MESH_cross_ref,
                         !is.na(MESH) & (!is.na(DOID) | !is.na(OMIM)))
```

We analyze the duplicates, to understand how to manage data.

```r
duplicates_summary <- sapply(MESH_cross_ref, function(x) {
  clean_x <- x[!is.na(x)]

  total_ids <- length(clean_x)
  unique_ids <- length(unique(clean_x))

  return(c(
    Total_IDs = total_ids,
    Unique_IDs = unique_ids,
    Dups = total_ids - unique_ids
  ))
})

print(duplicates_summary)
```

```
##            MESH DOID OMIM
## Total_IDs  6654 3840 4515
## Unique_IDs 6648 3840 4514
## Dups          6    0    1
```

We can see that we have 6 duplicates of MESH.

```r
if (!require("dplyr")) install.packages("dplyr")
library(dplyr)

mesh_dups <- MESH_cross_ref %>%
  group_by(MESH) %>%
  filter(n() > 1 & !is.na(MESH)) %>%
  arrange(MESH)

print(mesh_dups)
```

```
## # A tibble: 12 x 3
## # Groups:   MESH [6]
##    MESH    DOID         OMIM
##    <chr>   <chr>        <chr>
##  1 C537206 DOID_0050979 OMIM_613371
##  2 C537206 DOID_0050978 OMIM_117360
##  3 D000699 <NA>         OMIM_243000
##  4 D000699 DOID_0070145 OMIM_608654
##  5 D009221 DOID_668     <NA>
##  6 D009221 DOID_13374   OMIM_135100
##  7 D024741 DOID_0080326 <NA>
##  8 D024741 <NA>         OMIM_609015
##  9 D054739 DOID_8538    OMIM_267730
## 10 D054739 DOID_7848    <NA>
## 11 D056731 <NA>         OMIM_262190
## 12 D056731 DOID_0050470 OMIM_246200
```

This isn't a problem, simply different OMIM or different DOID map to the same MESH.

There aren't DOID duplicates, so we focus on the OMIM duplicates:

```r
omim_dup <- MESH_cross_ref %>%
  group_by(OMIM) %>%
  filter(n() > 1 & !is.na(OMIM))
```

```r
print(omim_dup)
```

```
## # A tibble: 2 x 3
## # Groups:   OMIM [1]
##   MESH    DOID         OMIM
##   <chr>   <chr>        <chr>
## 1 C580062 DOID_0111162 OMIM_162900
## 2 D009506 <NA>         OMIM_162900
```

This is a problem, the same OMIM (OMIM_162900.) map to different MESH.

We can delete row "D009506, NA, OMIM_162900" because the initial D of the code identifies "Chemicals and Drugs", so we will consider the one with the initial C that identifies "Diseases". See https://en.wikipedia.org/wiki/Medical_Subject_Headings#Categories for reference.

```r
# Delete the specific row: D009506, NA, OMIM_162900
MESH_cross_ref <- subset(MESH_cross_ref,
                                  !(MESH == "D009506" & is.na(DOID)
                                    & OMIM == "OMIM_162900"))
```

We clean `MESH_cross_ref` removing the prefixes and explode lines with mutiple values.

```r
if (!require("stringr")) install.packages("stringr")
library(stringr)
if (!require("tidyr")) install.packages("tidyr")
library(tidyr)

# Clean the data.frame removing prefixes
MESH_cross_ref <- MESH_cross_ref %>%
  mutate(
    DOID = str_remove_all(DOID, "DOID[:_]"),
    OMIM = str_remove_all(OMIM, "OMIM[:_]"),
    MESH = str_remove_all(MESH, "MESH:")
  )

# Explodes OMIM and DOID lines with more values
# Separate by "|" or ","
MESH_cross_ref <- MESH_cross_ref %>%
  separate_rows(DOID, sep = "\\|") %>%
  separate_rows(DOID, sep = ",") %>%
  separate_rows(OMIM, sep = "\\|") %>%
  separate_rows(OMIM, sep = ",") %>%
  mutate(across(everything(), str_trim)) %>%
  distinct()
```

- second file at https://raw.githubusercontent.com/DiseaseOntology/HumanDiseaseOntology/main/src/ontology/doid.obo

More complex to parse.

```r
if (!require("ontologyIndex")) install.packages("ontologyIndex")
library(ontologyIndex)

# Load the ontology (extract_tags = "everything" is required to get 'xref')
url <- "https://raw.githubusercontent.com/DiseaseOntology/HumanDiseaseOntology/main/src/ontology/doid.ob
doid_onto <- get_ontology(url, extract_tags = "everything")
```

```r
# Helper function to extract specific prefix from the xref list
extract_xref <- function(xref_list, prefix) {
  # Find all strings in the list that start with the prefix (e.g., "MESH:")
  matches <- grep(paste0("^", prefix), xref_list, value = TRUE)
  # Clean the prefix from the result (e.g., "MESH:D006394" -> "D006394")
  clean_matches <- gsub(paste0("^", prefix, ":"), "", matches)
  # Collapse multiple matches with a comma if they exist
  if (length(clean_matches) > 0) return(paste(clean_matches, collapse = ", ")) else return(NA)
}

MESH_obo <- data.frame(
  MESH = sapply(doid_onto$xref, extract_xref, prefix = "MESH"),
  DOID = doid_onto$id,
  OMIM  = sapply(doid_onto$xref, extract_xref, prefix = "MIM"),
  stringsAsFactors = FALSE
)

rownames(MESH_obo) <- NULL

# Removes rows in which MESH=NA, or both DOID and OMIM are NA
MESH_obo <- subset(MESH_obo,
                        !is.na(MESH) & (!is.na(DOID) | !is.na(OMIM)))

head(MESH_obo)
```

```
##           MESH         DOID OMIM
## 1      D006394 DOID:0001816 <NA>
## 3      D008659 DOID:0014667 <NA>
## 93  D000081012 DOID:0040091 <NA>
## 131    D012373 DOID:0050052 <NA>
## 134    D004887 DOID:0050061 <NA>
## 142 C000656784 DOID:0050072 <NA>
```

We analyze the duplicates, to understand how to manage data.

```r
duplicates_summary <- sapply(MESH_obo, function(x) {
  clean_x <- x[!is.na(x)]

  total_ids <- length(clean_x)
  unique_ids <- length(unique(clean_x))

  return(c(
    Total_IDs = total_ids,
    Unique_IDs = unique_ids,
    Dups = total_ids - unique_ids
  ))
})
print(duplicates_summary)
```

```
##            MESH DOID OMIM
## Total_IDs  3988 3988 1771
## Unique_IDs 3650 3988 1768
## Dups        338    0    3
```

Also here we need to handle only the OMIM duplicates.

```r
omim_dup <- MESH_obo %>%
  group_by(OMIM) %>%
  filter(n() > 1 & !is.na(OMIM))

print(omim_dup)
```

```
## # A tibble: 6 x 3
## # Groups:   OMIM [3]
##   MESH    DOID          OMIM
##   <chr>   <chr>         <chr>
## 1 C536350 DOID:0070331 612075
## 2 C536350 DOID:0080127 612075
## 3 D009084 DOID:0111402 252940
## 4 D014063 DOID:11514   137400
## 5 D009084 DOID:12801   252940
## 6 D005929 DOID:1455    137400
```

The identical OMIM that map to different MESH is only 137400: it maps to "fissured tongue" (MESH:D014063) and "geographic tongue" (MESH:D005929).

I will rely on the choice of Monarch Initiative (https://monarchinitiative.org/MONDO:0007655) that choose to associate OMIM:137400 to MESH:D014063.

```r
MESH_obo[MESH_obo$DOID == "DOID:1455" & MESH_obo$MESH == "D005929", "OMIM"] <- NA
```

We clean `MESH_obo` removing the prefixes and explode lines with mutiple values.

```r
# Clean the data.frame removing prefixes
MESH_obo <- MESH_obo %>%
  mutate(
    DOID = str_remove_all(DOID, "DOID[:_]"),
    OMIM = str_remove_all(OMIM, "(OMIM|MIM)[:_]"),
    MESH = str_remove_all(MESH, "MESH:")
  )

# Explodes OMIM and DOID lines with more values
# Separate by "|" or ","
MESH_obo <- MESH_obo %>%
  separate_rows(DOID, sep = "\\|") %>%
  separate_rows(DOID, sep = ",") %>%
  separate_rows(OMIM, sep = "\\|") %>%
  separate_rows(OMIM, sep = ",") %>%
  mutate(across(everything(), str_trim)) %>%
  distinct()
```

Merge the 2 `data.frame` in a single one.

```r
# Merge MESH_cross_ref and MESH_obo in an unique data.frame
from_DOID_OMIM_to_MESH <- rbind(MESH_cross_ref, MESH_obo)

# Keep only unique rows
from_DOID_OMIM_to_MESH <- unique(from_DOID_OMIM_to_MESH)
```

Sometimes, in the same cell we find 2 different MESH, in the format Cxx|Dyy, as mentioned before, we will keep only the Cxx types. And others times, more than one Cxx type. Since there are a lot of cases (>100), and our team doesn't have enough biological knowledge, we will simply keep the first one of the list.

```r
from_DOID_OMIM_to_MESH <- from_DOID_OMIM_to_MESH %>%
  mutate(
    # Attempt to extract the first code starting with C
    C_code = str_extract(MESH, "(?<=^|[|,])C[^|,]*"),
    # If no C code was found, take everything before the first delimiter
    MESH = coalesce(C_code, str_extract(MESH, "^[^|,]*"))
  ) %>%
  select(-C_code) %>%
  distinct()
from_DOID_OMIM_to_MESH <- unique(from_DOID_OMIM_to_MESH)
```

We analyze the duplicates, to understand how to manage data.

```r
duplicates_summary <- sapply(from_DOID_OMIM_to_MESH, function(x) {
  clean_x <- x[!is.na(x)]

  total_ids <- length(clean_x)
  unique_ids <- length(unique(clean_x))

  return(c(
    Total_IDs = total_ids,
    Unique_IDs = unique_ids,
    Dups = total_ids - unique_ids
  ))
})
print(duplicates_summary)
```

```
##            MESH DOID OMIM
## Total_IDs  8681 5865 5903
## Unique_IDs 7191 5009 5254
## Dups       1490  856  649
```

Now we handle some specific cases.

Let's consider this DOID case, but it's the same with OMIM.

If I have these rows:

- 1, 1, 1
- 1, 1, NA
- 1, 1, 2

We will collapse the row with the NA value in one of the other two, still keeping saved the other full row.

```r
# OMIM
from_DOID_OMIM_to_MESH <- from_DOID_OMIM_to_MESH %>%
  group_by(MESH, DOID) %>%
  fill(OMIM, .direction = "downup") %>%
  ungroup() %>%
  distinct()

# DOID
from_DOID_OMIM_to_MESH <- from_DOID_OMIM_to_MESH %>%
  group_by(MESH, OMIM) %>%
  fill(DOID, .direction = "downup") %>%
  ungroup() %>%
  distinct()
```

```r
duplicates_summary <- sapply(from_DOID_OMIM_to_MESH, function(x) {
  clean_x <- x[!is.na(x)]
  total_ids <- length(clean_x)
  unique_ids <- length(unique(clean_x))
  return(c(
    Total_IDs = total_ids,
    Unique_IDs = unique_ids,
    Dups = total_ids - unique_ids
  ))
})
print(duplicates_summary)
```

```
##            MESH DOID OMIM
## Total_IDs  8065 5574 5578
## Unique_IDs 7191 5009 5254
## Dups        874  565  324
```

Now the duplicates to handle are the half.

Now we handle an other case, in which a pair (DOID,OMIM) map to two different MESH: we will keep only the first of type Cxx (if exists).

```r
from_DOID_OMIM_to_MESH <- from_DOID_OMIM_to_MESH %>%
  group_by(DOID, OMIM) %>%
  mutate(has_C = any(startsWith(MESH, "C"), na.rm = TRUE)) %>%
  filter(!(has_C & startsWith(MESH, "D"))) %>%
  slice(1) %>%
  ungroup() %>%
  select(-has_C)
```

```r
duplicates_summary <- sapply(from_DOID_OMIM_to_MESH, function(x) {
  clean_x <- x[!is.na(x)]
  total_ids <- length(clean_x)
  unique_ids <- length(unique(clean_x))
  return(c(
    Total_IDs = total_ids,
    Unique_IDs = unique_ids,
    Dups = total_ids - unique_ids
  ))
})
print(duplicates_summary)
```

```
##            MESH DOID OMIM
## Total_IDs  8065 5574 5578
## Unique_IDs 7191 5009 5254
## Dups        874  565  324
```

Now the duplicates to handle are less.

Now we need to handle an other type of duplicates: (MESH,DOID) map to to different OMIM and (MESH,OMIM) map to different DOID. This duplicates are not a problem, simply are many-to-one cases.

Other case to handle:

- 1, x, A
- 2, z, A

A needs to be mapped to a single MESH: the first Cxx type found. The other A will be set to NA.

```r
# OMIM
from_DOID_OMIM_to_MESH <- from_DOID_OMIM_to_MESH %>%
  group_by(OMIM) %>%
  arrange(desc(startsWith(MESH, "C")), .by_group = TRUE) %>%
  # Edit the OMIM
  mutate(
    OMIM = ifelse(row_number() == 1, OMIM, NA)
  ) %>%
  ungroup()

# DOID
from_DOID_OMIM_to_MESH <- from_DOID_OMIM_to_MESH %>%
  group_by(DOID) %>%
  arrange(desc(startsWith(MESH, "C")), .by_group = TRUE) %>%
  # Edit the DOID
  mutate(
    DOID = ifelse(row_number() == 1, DOID, NA)
  ) %>%
  ungroup()
```

```r
duplicates_summary <- sapply(from_DOID_OMIM_to_MESH, function(x) {
  clean_x <- x[!is.na(x)]
  total_ids <- length(clean_x)
  unique_ids <- length(unique(clean_x))
  return(c(
    Total_IDs = total_ids,
    Unique_IDs = unique_ids,
    Dups = total_ids - unique_ids
  ))
})
print(duplicates_summary)
```

```
##            MESH DOID OMIM
## Total_IDs  7936 5445 5515
## Unique_IDs 7101 5009 5254
## Dups        835  436  261
```

As seen before, MESH duplicates are not a problem, so our `from_DOID_OMIM_to_MESH` is clean and ready to be used.

1.2 from Uniprot to Gene::NCBI

2 From Vitagraph

2.1 Nodes already in final type

Vitagraph is already an unique file. We need only to extract the nodes (and so the edges) of our interest.

```r
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
```

The nodes (that don't need some transformation) of our interest are:

```
valid_nodes_vitagraph <- c("Gene::NCBI","Compound::PubChem_Compounds",
                           "Disease::MESH")
```

```
# Create a search pattern from the valid_nodes_vitagraph vector
pattern <- paste(valid_nodes_vitagraph, collapse = "|")

# Filter the data frame:
# check if the pattern exists in both the 'head' AND the 'tail' columns
clean_vitagraph_small <- vitaGRAPH[
  grepl(pattern, vitaGRAPH$head) & grepl(pattern, vitaGRAPH$tail),
  c("head", "tail")
]

head(clean_vitagraph_small)
```

```
##              head             tail
## 1 Gene::NCBI:2157  Gene::NCBI:5264
## 2 Gene::NCBI:2157  Gene::NCBI:2158
## 3 Gene::NCBI:2157  Gene::NCBI:3309
## 4 Gene::NCBI:2157 Gene::NCBI:28912
## 5 Gene::NCBI:2157   Gene::NCBI:811
## 6 Gene::NCBI:2157  Gene::NCBI:2159
```

```
# Save the results to .tsv file
write.table(clean_vitagraph_small,
            file = "clean_vitagraph_small.tsv",
            sep = "\t",
            row.names = FALSE,
            quote = FALSE)

# Save the results to .csv file
write.csv(clean_vitagraph_small,
          file = "clean_vitagraph_small.csv",
          row.names = FALSE)
```

2.2 Map to MESH

Now we need to consider two other types of nodes that needs to be map: Disease::DOID and Disease::OMIM.