Supplementary File S1: A graph-based approach to mapping human exposure-outcome associations for chemical contaminants

Taylor A. M. Wolffe[1,2], Paul Whaley[1,3], Crispin Halsall[1]

[1]Lancaster Environment Centre, Lancaster University, Lancaster, UK
[2]Yordas Group, Lancaster Environment Centre, Lancaster University, Lancaster, UK
[3]Evidence-Based Toxicology Collaboration, Johns Hopkins Bloomberg School of Public Health, Baltimore, MD 21205, USA

Processing extracted exposure-outcome association data and pushing it to graph

First, all packages required for the processing of the raw data were imported. A connection with the Neo4j graph database was also established:

In [10]:
```
#Importing all required packages
import pandas as pd
from py2neo import  Graph, Node, Relationship

#Connecting to the neo4j graph database...
graph = Graph("http://localhost:7474/db/data/", auth=('UserHere', 'YourPasswordHere'))
```

Next, the long-form, flat data file containing all extracted exposure-outcome associations was loaded and read as a pandas dataframe:

In [11]:
```
#reading in the extracted data (Table S2) as a pandas dataframe
df = pd.read_excel('Table_S2.xlsx')

#removing all trailing white spaces from the values in the dataframe
df = df.apply(lambda x: x.str.strip() if x.dtype == "object" else x)
```

A data model was devised for mapping the exposure-outcome associations as a graph, and is depicted below in Figure S1.

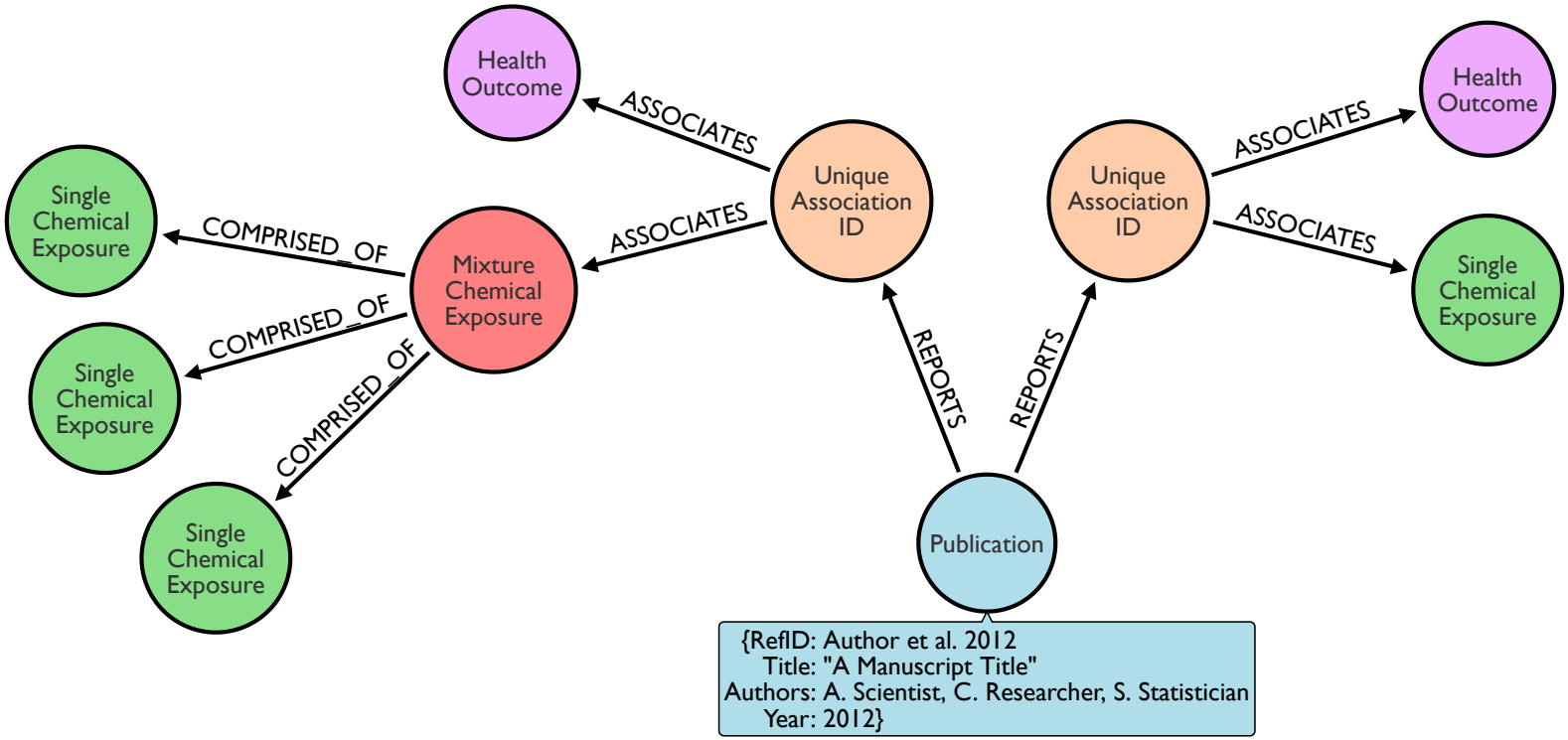

Figure S1: Graph Data Model describing the exposure-outcome associations studied using NHANES data in included publications

Creating Publication Nodes

A subset of all the attributes belonging to the "Publication" entities (i.e. Reference ID, Title, Authors and Publication Year) was created. The duplicates which result from the long-form data table were removed and nodes created using the dataframe of unique publications as below:

In [12]:
```
#Creating a subset of bibliographic information, i.e. all attributes belonging to the 'publication' entity.
df_Bib = df[['Reference ID', 'Title', 'Authors', 'PublicationYear']]

#Removing all duplicates from the subset of bibliographic information, such that only unique publications remain.
df_Bib = df_Bib.drop_duplicates()

#Iterating through the subset of bibliographic information and creating a node for every unique publication*:
for index, row in df_Bib.iterrows():
    a = Node("Publication", RefID=row['Reference ID'], title=row['Title'], authors=row['Authors'], year=row['Publicati
onYear'])
    graph.create(a)

#*Only run the above code once, to prevent creating duplicate nodes in the graph.
```

Creating Association Nodes

Associations are the unique entity described across a row in the flat, long-form data table (Table S2). Thus every new line of the table represents a new association. A node for every association in the dataset was created as above - first confirming that all of the assigned "Association IDs" were unique, and that none of these IDs had been erroneously duplicated.

In [13]:
```
#Checking for duplicated Association IDs
df[df[['Association ID']].duplicated() == True]
```

Out[13]:

	Notes	Reference ID	Title	Authors	PublicationYear	Association ID	Chemical	Chemical_and_Biological_Medium	Biological Medium, if reported	AssociatedG

0 rows × 38 columns

In [14]:
```
#Iterating through every row of data and creating a node for every unique Association ID*:
for index, row in df.iterrows():
    b = Node("Association", AssocID=row['Association ID'])
    graph.create(b)

#*Only run the above code once, to prevent creating duplicate nodes in the graph.
```

Creating 'REPORTS' relationship between Publication and Association

A "REPORTS" relationship was created between corresponding Publication and Association nodes as below:

In [15]:
```
#Creating a list of 'Publication-REPORTS->Association' triples by iterating through the data:
Pub_Assoc_Rels = []

for index, row in df.iterrows():
    rel = [row['Reference ID'], "REPORTS", row['Association ID']]
    Pub_Assoc_Rels.append(rel)
#As every row in the data contains a unique Association, there should not be any duplicates in the Pub_Assoc_Results l
ist


#Creating the relationships by matching the corresponding publication and association nodes in the graph*:
for rel in Pub_Assoc_Rels:
    b = graph.nodes.match(RefID=rel[0]).first()
    c = graph.nodes.match(AssocID=rel[2]).first()
    d = Relationship(b, rel[1], c)
    graph.create(d)

#*Only run the above code once, to prevent creating duplicate nodes in the graph.
```

Creating Single Chemical Exposure Nodes and Association-Exposure Relationships

The majority of values within the "Chemical_and_Biological_Medium" column of the dataframe are single chemical exposures. However, some of these values are actually mixtures (e.g. "ΣPFAS"). The individual components which make up these mixed exposures (e.g. "PFNA", "PFOA" and "PFOS") are housed within the "ChemicalConstituent_x" columns, and are themselves considered "single" chemical exposures in the graph data model (see Fig. S1).

In [16]:
```
#Single chemical exposures within the 'Chemical_and_Biological_Medium' column:

#Creating a list of 'Association-ASSOCIATES->Single Chemical Exposure' triples by iterating through the data:
Single_chems_assocs = []

for index, row in df.iterrows():
    #single chemical exposures will not have any data in the "ChemicalConstituent_1" column
    if pd.notna(row['ChemicalConstituent_1']) is False:
        Single_chems_assocs.append([row['Association ID'], "ASSOCIATES", "`" + row['Chemical_and_Biological_Medium'] +
"`"])

#Creating a list of unique 'Single Chemical Exposure' nodes by iterating through the triples:
single_chem_nodes = []

for triple in Single_chems_assocs:
    if triple[2] not in single_chem_nodes:
        single_chem_nodes.append(triple[2])


#Single chemical exposures within the 'ChemicalConstituent_x' columns:

#Slicing the last 28 rows of the dataframe, which contain the individual chemical components
chem_constituents = df.iloc[:, -28:]

#Creating a list of column headers (i.e. "ChemicalConstitent_1" to "ChemicalConstituent_28"):
cols = list (chem_constituents.columns)

#Creating a list of unique single chemical exposures by iterating through the data:
single_constituents = []

for index, row in chem_constituents.iterrows():
    for item in cols:
        if pd.notna(row[item]) is True and row[item] not in single_constituents:
            single_constituents.append("`" + row[item] + "`")



#Creating all single chemical exposure nodes:

#Ensuring all single chemical exposures are presented in one list, and that there are no duplicates:
for item in single_constituents:
```

```
    if item not in single_chem_nodes:
        single_chem_nodes.append(item)

#Pushing all the single chemical exposure nodes to graph*:
for node in single_chem_nodes:
    a = Node("SingleChemicalExposure", name=node)
    graph.create(a)

#Pushing the relationship between Association and Exposure to graph*:
for triple in Single_chems_assocs:
    b = graph.nodes.match(AssocID=triple[0]).first()
    c = graph.nodes.match(name=triple[2]).first()
    d = Relationship(b, triple[1], c)
    graph.create(d)

#*Only run the above code once, to prevent creating duplicate nodes in the graph.
```

Creating Mixed Chemical Exposure Nodes and Association-Exposure Relationships

As the constituents which make up a mixed exposure can vary from one publication to the next (e.g. Publication A might consider "ΣPFAS" to be comprised of "PFNA", "PFOA" and "PFOS" whereas Publication B might consider "ΣPFAS" to be comprised of "PFHxS", "PFOS" and "PFBS"), each of the mixed chemical exposures were treated as a unique entity. To ensure the nodes created based on these entities remain unique, the names of the mixed chemicals extracted from included publications were appended with the Reference ID of the publication reporting the mixed chemical exposure.

Mixed chemical exposure nodes, relationships between these mixed chemicals and their single chemical constituents, and between mixed chemical exposures and Association nodes were created as below:

In [17]:
```
#Creating a list of Assocation-ASSOCIATES->MixedChemicalExposure triples:
multi_chems_assoc = []

for index, row in df.iterrows():
    #Mixed chemical exposures will at least have an entry in the "ChemicalConstituent_1" column of the dataframe
    if pd.notna(row['ChemicalConstituent_1']) is True:
        row['Chemical_and_Biological_Medium'] = row['Chemical_and_Biological_Medium'] + " (" + row['Reference ID'] +
")"
        multi_chems_assoc.append([row['Association ID'], "ASSOCIATES", "`" + row['Chemical_and_Biological_Medium'] +
"`"])

#Creating a list of MixedChemicalExposure-COMPRISED_OF->SingleChemicalExposure triples:
multi_single_assocs = []

for index, row in df.iterrows():
    for item in cols:
        if pd.notna(row[item]) is True:
            multi_single_assocs.append(["`" + row['Chemical_and_Biological_Medium'] + " (" + row['Reference ID'] + ")"
+ "`", "COMPRISED_OF", "`" + row[item] + "`"])


#Creating a list of unique mixed chemical exposures
multi_chem_nodes = []
for item in multi_single_assocs:
    if item[0] not in multi_chem_nodes:
        multi_chem_nodes.append(item[0])

#Pushing all the mixed chemical exposure nodes to graph*:
for node in multi_chem_nodes:
    a = Node("MixedChemicalExposure", name=node)
    graph.create(a)


#Pushing the relationship between Association and mixed chemical exposure, and between eixed chemical exposure and sin
gle chemical exposure to graph*:
for triple in multi_chems_assoc:
    b = graph.nodes.match(AssocID=triple[0]).first()
    c = graph.nodes.match(name=triple[2]).first()
    d = Relationship(b, triple[1], c)
    graph.create(d)
```

```
for triple in multi_single_assocs:
    b = graph.nodes.match(name=triple[0]).first()
    c = graph.nodes.match(name=triple[2]).first()
    d = Relationship(b, triple[1], c)
    graph.create(d)

#*Only run the above code once, to prevent creating duplicate nodes in the graph.
```

Creating Health Outcome Nodes and Association-Outcome Relationships

Finally, the 'Health Outcome' nodes were created and related to the appropriate Association nodes as below:

In [18]:
```
#Creating a list of all Association-ASSOCIATES->Health Outcome triples
outcome_assocs = []

for index, row in df.iterrows():
    outcome_assocs.append([row['Association ID'], "ASSOCIATES", "`" + row['AssociatedOutcome'] + "`"])

#Creating a list of unique 'Health Outcome' nodes by iterating through the triples
outcome_nodes = []

for triple in outcome_assocs:
    if triple[2] not in outcome_nodes:
        outcome_nodes.append(triple[2])

#Pushing all unique HealthOutcome nodes to graph*:
for node in outcome_nodes:
  a = Node("HealthOutcome", name=node)
  graph.create(a)

#Pushing the relationship between Association and Outcome to graph*:
for triple in outcome_assocs:
    b = graph.nodes.match(AssocID=triple[0]).first()
    c = graph.nodes.match(name=triple[2]).first()
    d = Relationship(b, triple[1], c)
    graph.create(d)

#*Only run the above code once, to prevent creating duplicate nodes in the graph.
```