# eTransafe Omeprazole Heatmap

This is the use scenario that has been described in the eTox project for exploring the differences with respect to adverse events between omeprazole preclinical and clinical. In order to compute these results the following scenario is executed:

1. authenticate with the eTransafe keycloak
2. Translate the provided compound (e.g., omeprazole) to a SMILES
3. Retrieve similar compounds based on structural similarity
4. Retrieve data from the preclinical and clinical databases
5. Aggregate the data per system organ class
6. Visualize the data using a heatmap

(C) 2021 Erasmus University Medical Center, Rotterdam, The Netherlands Author: Erik M. van Mulligen, [e.vanmulligen@erasmusmc.nl](mailto:e.vanmulligen@erasmusmc.nl)

import sys  
sys.path.append('/Users/mulligen/git/etransafe-use-scenarios')  
from knowledgehub.api import KnowledgeHubAPI  
import ipywidgets as w  
from IPython.display import display, Javascript  
from ipypublish import nb\_setup  
import numpy as np  
import numpy.ma as ma  
import seaborn as sns  
import pandas  
from matplotlib.colors import LogNorm, Normalize  
import matplotlib.pyplot as plt  
import os  
import warnings  
warnings.filterwarnings("ignore")

api = KnowledgeHubAPI(server='TEST', client\_secret='39c644b3-1f23-4d94-a71f-e0fb43ebd760')

# 1. Authentication

Use eTransafe's authentication service to get access to available services

from IPython.core.display\_functions import clear\_output  
  
username = w.Text(value='erik.mulligen',placeholder='Knowledge Hub account', description='username:', disabled=False)  
password = w.Password(value='', placeholder='Knowledge Hub password', description='password:', disabled=False)  
loginBtn = w.Button(description='Login')  
def on\_button\_clicked(\_):  
 if api.login(username.value, password.value) == False:  
 print("Failed to login")  
 else:  
 print("successfully logged in")  
 display(Javascript('IPython.notebook.execute\_cell\_range(IPython.notebook.get\_selected\_index()+1, IPython.notebook.get\_selected\_index()+2)'))  
  
loginBtn.on\_click(on\_button\_clicked)  
w.VBox([username, password, loginBtn])

## 2. Translate compound to SMILES using semantic services

For the entered compound name, retrieve the associated SMILES using the semantic services

compoundName = w.Text(value='omeprazole',placeholder='compound name', description='compound:', disabled=False)  
compoundBtn = w.Button(description='Retrieve')  
compoundSmile = None  
  
def on\_compound\_entered(\_):  
 global compoundSmile  
 print(f'retrieving smiles for {compoundName.value}')  
 compoundSmile = api.ChemistryService().getSMILESByName(compoundName.value)  
 print(f'Found SMILES {compoundSmile} for {compoundName.value}')  
 display(Javascript('IPython.notebook.execute\_cell\_range(IPython.notebook.get\_selected\_index()+1, IPython.notebook.get\_selected\_index()+2)'))  
  
compoundBtn.on\_click(on\_compound\_entered)   
w.VBox([compoundName, compoundBtn])

## 3. Retrieve similar compounds

similar\_compounds = api.SimilarityService().get(compoundSmile[0])  
compoundNames = []  
names = []  
smiles = []  
similarities = []  
  
if similar\_compounds is not None:  
 for similar\_compound in similar\_compounds:  
 names.append(similar\_compound['name'])  
 smiles.append(similar\_compound['smiles'])  
 similarities.append(similar\_compound['distance'])  
  
pd = nb\_setup.setup\_pandas()  
df = pd.DataFrame(np.random.rand(len(names),3),columns=['NAME','SMILES','SIMILARITY'])  
df.NAME = names  
df.SMILES = smiles  
df.SIMILARITY = similarities  
df.round(3)

# 4. Retrieve data from the preclinical and clinical databases

# filter studies on being able to have a findingCode and findingVocabulary and not having findings for dose is 0.0 (control group)  
def filterStudies(studies):  
 return [study for study in studies if study['FINDING']['findingVocabulary'] is not None and study['FINDING']['findingCode'] is not None and study['FINDING']['findingCode'] != 'MC:2000001'   
 and ('dose' not in study['FINDING'] or study['FINDING']['dose'] != 0.0)]  
  
def count\_studies(studies):  
 return len(set([study['STUDY']['id'] for study in studies]))  
  
studies = {}  
studies['faerspa'] = filterStudies(api.Faers().getStudiesBySMILES(smiles))  
print(f'{count\_studies(studies["faerspa"])} FAERS studies')  
api.SemanticService().getSocs(studies['faerspa'], algorithm='MEDDRAPT2MEDDRASOC')  
studies['medlinepa'] = filterStudies(api.Medline().getStudiesBySMILES(smiles))  
print(f'{count\_studies(studies["medlinepa"])} MEDLINE studies')  
api.SemanticService().getSocs(studies['medlinepa'], algorithm='MEDDRAPT2MEDDRASOC')  
studies['clinicaltrialspa'] = filterStudies(api.ClinicalTrials().getStudiesBySMILES(smiles))  
print(f'{count\_studies(studies["clinicaltrialspa"])} CT studies')  
api.SemanticService().getSocs(studies['clinicaltrialspa'], algorithm='MEDDRAPT2MEDDRASOC')  
studies['dailymedpa'] = filterStudies(api.DailyMed().getStudiesBySMILES(smiles))  
print(f'{count\_studies(studies["dailymedpa"])} DAILYMED studies')  
api.SemanticService().getSocs(studies['dailymedpa'], algorithm='MEDDRAPT2MEDDRASOC')  
studies['eTOXsys'] = filterStudies(api.eToxSys().getStudiesByCompoundNames(names))  
print(f'{count\_studies(studies["eTOXsys"])} eTOX studies')  
api.SemanticService().getSocs(studies['eTOXsys'], algorithm='MA2MEDDRASOC')

# 5. Aggregate the data per system organ class

Since the eTox data reports events with an organ. We use our own method to map it to MedDRA's system organ class to make it comparable

service\_names = [  
 {'name': 'faerspa', 'title':'FAERS'},  
 {'name': 'medlinepa', 'title': 'MEDLINE'},  
 {'name': 'eTOXsys', 'title': 'eTOXsys'},  
 {'name': 'dailymedpa', 'title': 'DailyMed'},  
 {'name': 'clinicaltrialspa', 'title': 'ClinicalTrials'},  
 ]  
  
system = {}  
all\_compounds = [c.lower() for c in names]  
socs = {}  
  
for service in service\_names:  
 source = service['name']  
 if source in studies:  
 for study in studies[source]:  
 soc = study['FINDING']['\_\_soc']  
 if soc not in socs:  
 if soc not in socs:  
 socs[soc] = set()  
 socs[soc].add(study['STUDY']['id'])  
  
   
# sort the socs per count  
all\_socs = {k: v for k, v in sorted(socs.items(), key=lambda item: item[1], reverse=True)}  
  
  
# traverse all studies and create a matrix per source  
total\_count = {}  
  
for service in service\_names:  
 source = service['name']  
 if source in studies:  
 source\_study\_ids = set()   
 system[source] = {  
 'data':np.zeros((len(all\_socs),len(all\_compounds)), dtype=int).tolist(),  
 'studies': [[set() for i in range(len(all\_compounds))] for j in range(len(all\_socs))],  
 'rows':list(all\_socs.keys()),  
 'cols':all\_compounds}  
 # study\_ids = [ [set()] \* len(all\_compounds) for i in range(len(all\_socs)) ]  
 for study in studies[source]:   
 soc = study['FINDING']['\_\_soc']  
 row = system[source]['rows'].index(soc)  
 col = system[source]['cols'].index(study['COMPOUND']['name'].lower())  
 # study\_ids[row][col].add(study['STUDY']['id'])  
 system[source]['data'][row][col] += 1  
 system[source]['studies'][row][col].add(study['STUDY']['id'])  
 source\_study\_ids.update(system[source]['studies'][row][col])  
 total\_count[source] = len(source\_study\_ids)

# 6. Visualize the data using a heatmap

Using seaborn to visualize the content of the various databases. Note that we have to think about ways to easier compare the various results.

for source,value in system.items():  
 plt.figure(figsize=(12,9))  
 data = system[source]['data']  
   
 # create mask  
 data\_mask = ma.array(np.zeros((len(all\_socs.keys()), len(all\_compounds))))  
 for r in range(0, len(all\_socs.keys())):  
 for c in range(0, len(all\_compounds)):  
 data\_mask[r][c] = 1 if data[r][c] == 0 else 0  
   
 colormap = sns.cubehelix\_palette(as\_cmap=True, light=.9)  
   
 col\_count = [0] \* len(all\_compounds)  
 # for row in data:  
 # for col in range(0, len(row)):  
 # col\_count[col] += row[col]  
 col\_studies = [set() for \_ in range(0, len(all\_compounds))]  
 for row in range(len(all\_socs)):  
 for col in range(0, len(all\_compounds)):  
 col\_count[col] += data[row][col]  
 col\_studies[col].update(system[source]['studies'][row][col])  
  
   
 col\_labels = [f'{all\_compounds[c]} (N={len(col\_studies[c])})' for c in range(0, len(all\_compounds))]  
   
 ax = sns.heatmap(data, mask=data\_mask, xticklabels=col\_labels, yticklabels=list(all\_socs.keys()), annot=True, fmt=".0f", cmap=colormap)   
 ax.set\_xticklabels(ax.get\_xmajorticklabels(), rotation=45)  
   
 title = source  
 for service\_name in service\_names:  
 if service\_name['name'] == source:  
 title = f'{service\_name["title"]} (N={total\_count[source]})'  
 break  
   
 plt.title(title, fontsize = 14)  
 plt.ylabel("Findings per organ class", fontsize = 12)  
 plt.xlabel("Similar compounds", fontsize = 12)  
 plt.show()  
  
 print('')  
 print('')