D:\NHRI\AMR\amr-main\explore\pathogen_paper\gbd_fix_draws.py

```
"""
Purpose: Adjust several GBD causes according to mcod scalars (based on causes in chain and as underlying cause of death) and add uncertainty
"""
```

Getting adjustment values for several GBD causes

```
import pandas as pd
import numpy as np
import dask.dataframe as dd
import ipdb

from db_queries import get_cause_metadata
from cod_prep.downloaders import get_current_cause_hierarchy, add_location_metadata, add_cause_metadata, get_current_location_hierarchy, prep_child_to_available_parent_map, get_all_child_cause_ids, add_age_metadata
from mcod_prep.utils.mcause_io import *
from mcod_prep.utils.nids import add_nid_metadata
from amr_prep.utils.amr_io import *
from db_tools import ezfuncs, query_tools
from db_queries import get_outputs as go
from get_draws.api import get_draws
from db_queries import get_population

from functools import reduce
import argparse

CONF = Configurator()

lh = get_current_location_hierarchy()
ch = get_cause_metadata(cause_set_id=3, gbd_round_id=6, decomp_step="step4")

DIR = 'FILEPATH'
mcod_final = pd.read_csv(f'{DIR}FILEPATH')

all_u5 = [2,3,42,1,388,34,238,389,5,4]
gbd2019_u5 = [2,3,4,5]

# group ages for over 5
o5 = list(range(6,21)) + [30,31,32,235]

def change_names_of_pathogens(df):

    '''
    change names of cause based on cause id
    '''
    df.loc[df.cause_id==1026, 'cause_name'] = 'Total burden related to hepatitis B'
    df.loc[df.cause_id==959, 'cause_name'] = 'Invasive Non-typhoidal Salmonella (iNTS)'
    df.loc[df.cause_id==396, 'cause_name'] = 'Gonococcal infection'
    df.loc[df.cause_id==320, 'cause_name'] = "Paratyphoid fever"
    return df

def fix_ages(df, draws=True, keep_all=False):
```

```
    '''
    create under 5 and over 5 age categories
    '''

    keep_col1 = ["agg_age_group_id", "measure_id", "cause_id", "location_id"]
    keep_col2 = ["age_group_id", "measure_id", "cause_id", "location_id"]

    if draws == True:
        keep_col1 += ["draws"]
        keep_col2 += ["draws"]

    df.loc[df["age_group_id"].isin(all_u5), "agg_age_group_id"] = 1
    df.loc[df["age_group_id"].isin(o5), "agg_age_group_id"] = 192

    df_u5 = df.loc[df["agg_age_group_id"] == 1]
    df_o5 = df.loc[df["agg_age_group_id"] == 192]

    if keep_all == True:
        df_all = df.loc[df["age_group_id"] == 22]

    df_u5 = df_u5.groupby(keep_col1, as_index=False)["val"].sum()
    df_o5 = df_o5.groupby(keep_col1, as_index=False)["val"].sum()

    df_u5.rename(columns={"agg_age_group_id":"age_group_id"}, inplace=True)
    df_o5.rename(columns={"agg_age_group_id":"age_group_id"}, inplace=True)

    if keep_all == True:
        df_fixed = pd.concat([df_all, df_u5, df_o5])
    else:
        df_fixed = pd.concat([df_u5, df_o5])


    df_fixed = add_age_metadata(df_fixed, "age_group_name")

    return df_fixed

def output_scalars(tot_draws_df):

        '''
        save mean values and UI for the scalars used for gbd adjustment
        '''

        scalars = tot_draws_df[['draws','scalar','age_group_name','pathogen']]

        scalars['mean_val'] = scalars.groupby(by= ['age_group_name','pathogen'])
['scalar'].transform(np.mean)
        scalars['lower_val'] = scalars.groupby(by= ['age_group_name','pathogen'])
['scalar'].transform(np.quantile,q=0.025)
        scalars['upper_val'] = scalars.groupby(by= ['age_group_name','pathogen'])
['scalar'].transform(np.quantile,q=0.975)
        scalars =
scalars.drop_duplicates(['age_group_name','pathogen','mean_val','lower_val','upper_val'])
        scalars.drop(['draws','scalar'], axis=1, inplace=True)

```

```
        scalars['scalar'] = scalars.apply(lambda x: f"{str(round(x.mean_val,2))} ({str(round(x.lower_val,2))} - {str(round(x.upper_val,2))})", axis=1)

        scalars.rename(columns= {'age_group_name':'Age', 'pathogen':'Pathogen'}, inplace=True)
        scalars.drop(['mean_val','lower_val','upper_val'], axis=1, inplace=True)

        scalars.to_csv(f'{DIR}FILEPATH', index=False, float_format='%.2f')


def check_results(df, locs):

    '''
    check that all need columns are present in the results
    '''
    cols = ['location_id', 'pathogen', 'age_group_name', 'measure_id'] + draw_name
    for col in cols:
        assert col in df.columns, f'{col} is not present in the result dataframe'

    # check it has all age groups
    for i in ['All Ages', 'Under 5']:
        assert i in df.age_group_name.unique(), f'{i} is missing from the dataframe'

    #check the location number is correct
    assert len(locs) == len(df.location_id.unique()), 'the number of locations is not consistent'



if __name__ =='__main__':

    parser = argparse.ArgumentParser()
    parser.add_argument("--locs", help= "'global' to for global locations,'region' for regions, and 'country' for countries. if not specified, runs for regions and global",type=str, default='all')
    parser.add_argument('--scalars', help= '"save" to output scalars')
    arg = parser.parse_args()


    if arg.locs == 'global':
        locs = [1]
        loc_type = 'global'
    elif arg.locs =='region':
        locs = lh.loc[lh["level"].isin([1]), "location_id"].unique().tolist()
        loc_type = 'regions'
    elif arg.locs == 'country':
        locs = lh.loc[lh["level"]==3,"location_id"].unique().tolist()
        loc_type = 'countries'
    else:
        locs = lh.loc[lh["level"].isin([2,1]), "location_id"].unique().tolist()+[1]
        loc_type = 'regions_global'
```

```
    #list of GBD pathogens of interest (POI) to calculate uncertainty for
    poi = [
        "Tuberculosis",
        'African trypanosomiasis',
        "HIV/AIDS",
        "Malaria",
        "Pertussis", #Whooping Cough
        "Syphilis",
        "Measles",
        "Acute hepatitis A",
        "Dengue",
        "Tetanus",
        "Hepatitis B",
        "Varicella and herpes zoster",
        "Rabies",
        "Schistosomiasis",
        "Chagas disease",
        "Visceral leishmaniasis",
        "Diphtheria",
        "Yellow fever",
        "Ascariasis",
        "Acute hepatitis E",
        "Cystic echinococcosis",
        "Cysticercosis",
        "Cervical cancer",
        "Other neglected tropical diseases",
        "Zika virus",
        "Neisseria gonorrhoeae",
    ]

    mcod_poi = mcod_final.loc[mcod_final["pathogen"].isin(poi)]

    # Calculate uncertainty
    x = 1000
    draw_name = [f'draw_{i}' for i in list(range(0, x))]

    draw_dfs = []

    np.random.seed(42)
    for path in mcod_poi["pathogen"].unique().tolist():
        for age in mcod_poi.loc[mcod_poi["pathogen"] == path,
"age_group_name"].unique().tolist():
            path_df = mcod_poi.loc[(mcod_poi["pathogen"] == path) &
(mcod_poi["age_group_name"] == age)]

            n = float(path_df["Deaths where UCoD"] + path_df["Deaths in Chain"])
            if n != 0:
                p = float(path_df["Deaths where UCoD"]/n)

                draws = (1/(np.random.binomial(n, p, x)/n))

                path_draws = pd.DataFrame(list(zip(draw_name, draws)),
                       columns =['draws', 'scalar'])
                path_draws["age_group_name"] = age
```

```
                path_draws["pathogen"] = path

                draw_dfs.append(path_draws)

    tot_draws = pd.concat(draw_dfs)
    tot_draws.replace([np.inf, -np.inf], 1, inplace=True)

    tot_draws = tot_draws.fillna(1)


    mcod_poi["total"] = mcod_poi["Deaths where UCoD"] + mcod_poi["Deaths in Chain"]
    mcod_poi_for_vet = mcod_poi.groupby(["pathogen", "age_group_name"], as_index=False)["total"].sum()

    mcod_poi_for_vet = mcod_poi_for_vet.pivot(index=["pathogen"], columns="age_group_name", values="total")
    mpv = mcod_poi_for_vet.reset_index()

    # causes that shouldn't have any correction (scalar = 1)
    no_correction = []

    # causes that should have a all ages scalar used for all age and for under 5
    to_correct_u5 = []

    # causes that should have a all age scalar for over 5
    to_correct_o5 = []

    cut_off = 200

    for i in mpv.pathogen.unique():
        if i not in ['Acute hepatitis E','Cervical cancer']:
            if mpv.loc[mpv.pathogen==i, 'All Ages'].values[0]<200:
                no_correction.append(i)
                continue
            elif mpv.loc[mpv.pathogen==i, 'Under 5'].values[0]<200:
                to_correct_u5.append(i)
                continue
    # check if all_age corrections needed for over 5:
    for i in mpv.pathogen.unique():
        if i not in ['Acute hepatitis E','Cervical cancer']:
            if (mpv.loc[mpv.pathogen==i, 'All Ages'].values[0]>=200)&(mpv.loc[mpv.pathogen==i, '5 plus'].values[0]<200):
                to_correct_o5.append(i)

    print(f"we are not correcting the following causes: {no_correction}")
    print(f"we are using the all age scalar the following causes for under 5: {to_correct_u5}")
    print(f"we are using the all age scalar the following causes for over 5: {to_correct_o5}")


    # no correction for under 5 and all ages:
    for i in no_correction:
        tot_draws.loc[tot_draws["pathogen"] == i, "scalar"]  = 1
```

```
250
251     # to correct using only all age scalars
252     for i in to_correct_u5:
253         tot_draws.loc[(tot_draws["pathogen"] == i)
254             & (tot_draws["age_group_name"] == "Under 5"), "scalar"] =
    tot_draws.loc[(tot_draws["pathogen"] == i)
255                         & (tot_draws["age_group_name"] == "All Ages"), "scalar"]
256
257     for i in to_correct_o5:
258         tot_draws.loc[(tot_draws["pathogen"] == i)
259             & (tot_draws["age_group_name"] == "5 plus"), "scalar"] =
    tot_draws.loc[(tot_draws["pathogen"] == i)
260                         & (tot_draws["age_group_name"] == "All Ages"), "scalar"]
261
262
263     # remove Cervical cancer for under 5
264     tot_draws = tot_draws.loc[~((tot_draws["pathogen"] == "Cervical cancer")
265                     & (tot_draws["age_group_name"] == "Under 5"))]
266
267     # Adjust for Hepatitis E
268     tot_draws.loc[(tot_draws["pathogen"] == "Acute hepatitis E")
269                   & (tot_draws["age_group_name"] == "Under 5"), "scalar"]  = 1
270     # add cause_id to tot_draws
271     tot_draws = tot_draws.merge(ch[['cause_name','cause_id']], how='left',
    left_on='pathogen', right_on='cause_name')
272
273     # Name and cause id adjustments
274     tot_draws.loc[tot_draws.pathogen=='Neisseria gonorrhoeae', 'cause_id'] =
    ch.loc[ch.cause_name=='Gonococcal infection','cause_id'].unique()[0]
275     tot_draws.loc[tot_draws.pathogen=='Hepatitis B', 'cause_id'] =
    ch.loc[ch.acause=='total_hep_b_reporting','cause_id'].unique()[0]
276     tot_draws.loc[tot_draws.pathogen=='Pertussis', 'cause_id'] =
    ch.loc[ch.acause=='whooping','cause_id'].unique()[0]
277
278     if arg.scalars =='save':
279         output_scalars(tot_draws)
280
281     gbd_causes = tot_draws.cause_id.unique()
282
283     assert len(gbd_causes)==26
284
285     # get draws (counts) for Deaths and YLL (years of life lost)
286     cc = get_draws("cause_id",gbd_causes,
287                    source="codcorrect",
288             metric_id=1,
289             measure_id=[1,4],
290             release_id=6,
291             location_id=locs,
292             sex_id=3,
293             year_id=2019,
294            age_group_id=gbd2019_u5 + o5)
295     cc = cc.fillna(0)
296
297     # Load in YLDs (rates)
```

```
    como = get_draws("cause_id",gbd_causes,
                     source="como",
             metric_id=3,
             measure_id=[3],
             release_id=6,
             location_id=locs,
             sex_id=3,
             year_id=2019,
            age_group_id=gbd2019_u5 + o5)


    # Melt draws for gbd
    IDs = ["measure_id",
           "location_id",
           "metric_id",
        "age_group_id",
        "cause_id"
     ]

    print("melting!")
    cc = cc.melt(
        id_vars=IDs,
        value_vars=draw_name, var_name='draws', value_name='val'
    )
    cc["val"] = cc["val"].fillna(0)

    como = como.melt(
        id_vars=IDs,
        value_vars=draw_name, var_name='draws', value_name='val'
    )


    pops = get_population(age_group_id=gbd2019_u5 + o5 + [22], location_id=locs, year_id=2019, sex_id=3, release_id=6)
    como = como.merge(pops[["age_group_id", "location_id", "population"]], how="left", on=["age_group_id", "location_id"], validate='m:1')

    como_count = como.copy()
    como_count["val"] = como_count["val"]*como_count["population"]
    como_count["metric_id"] = 1
    como_count["val"] = como_count["val"].fillna(0)

    cc = fix_ages(cc, draws=True)
    como_count = fix_ages(como_count, draws=True)

    cc = add_cause_metadata(cc, "cause_name")
    cc = change_names_of_pathogens(cc)

    como_count = add_cause_metadata(como_count, "cause_name")
    como_count = change_names_of_pathogens(como_count)

    cc.rename(columns={"cause_name":"pathogen"}, inplace=True)
    como_count.rename(columns={"cause_name":"pathogen", "val":"ylds"}, inplace=True)

```

```
350
351     # Fix for exceptions on GBD
352     cc = cc.loc[~((cc["pathogen"] == "Cervical cancer")
353                   & (cc["age_group_name"] == "Under 5"))]
354
355     pathogen_dict = {'Non-typhoidal Salmonella':'Invasive Non-typhoidal Salmonella (iNTS)',
356                      'Hepatitis B':'Total burden related to hepatitis B',
357                       'Neisseria gonorrhoeae' : 'Gonococcal infection'}
358
359     tot_draws.pathogen=tot_draws.pathogen.apply(lambda x: pathogen_dict[x] if x in
    pathogen_dict.keys() else x)
360
361     to_fix = cc.merge(tot_draws, how="left", on=["draws", "cause_id",
    "age_group_name"],suffixes= ['_cc', '_tot_draws'], validate='m:1')
362     to_fix = to_fix.drop(['pathogen_cc','cause_name'], axis=1)
363     to_fix = to_fix.rename(columns={'pathogen_tot_draws':'pathogen'})
364
365     to_fix["fixed_fatal"] = to_fix["val"]*to_fix["scalar"]
366
367     ylls = to_fix.loc[to_fix["measure_id"] == 4]
368     ylls.rename(columns={"fixed_fatal":"fixed_ylls"}, inplace=True)
369
370     deaths = to_fix.loc[to_fix["measure_id"] == 1]
371     deaths.rename(columns={"fixed_fatal":"fixed_deaths"}, inplace=True)
372
373     id_cols = ["location_id", "pathogen", "draws", "age_group_name"]
374     measures = [como_count[id_cols + ["ylds"]],
375                 ylls[id_cols + ["fixed_ylls"]],
376                 deaths[id_cols + ["fixed_deaths", "scalar"]]]
377
378     final_fix = reduce(lambda  left,right: pd.merge(left,right,on=id_cols,
379                                                     how='outer'), measures)
380
381     final_fix["fixed_dalys"] = final_fix["fixed_ylls"] + final_fix["ylds"]
382
383
384     # create All Age category
385     final_fix_aa = final_fix.groupby(["location_id", "pathogen", "draws"], as_index=False)
    ["fixed_deaths",
386                                                                          "fixed_ylls",
387
    "fixed_dalys"].apply(lambda x: x.sum())
388     final_fix_aa["age_group_name"] = "All Ages"
389
390     final_fix = pd.concat([final_fix, final_fix_aa])
391     final_fix = final_fix.fillna(0)
392
393     final_fix.drop("ylds", axis=1, inplace=True)
394     final_fix = pd.melt(final_fix, id_vars=["location_id", "pathogen", "draws",
    "age_group_name", "scalar"],
395                         value_vars=["fixed_ylls", "fixed_deaths", "fixed_dalys"],
396                         var_name="measure_id",
397                         value_name='fixed_val',
398                         ignore_index=True)
```

```
399
400
401     measure_map = {
402         "fixed_ylls":4,
403         "fixed_deaths":1,
404         "fixed_dalys":2
405     }
406
407     final_fix["measure_id"] = final_fix["measure_id"].map(measure_map)
408
409     results= final_fix.drop('scalar', axis=1)
410
411     results = results.pivot(index= ['location_id',
    'pathogen','age_group_name','measure_id'], columns='draws').reset_index()
412
413     draws_name=[f'draw_{i}' for i in range(1000)]
414
415     results.columns = results.columns.droplevel(1)
416
417     cols = []
418     count = 0
419     for column in results.columns:
420         if column == 'fixed_val':
421             cols.append(f'draw_{count}')
422             count+=1
423             continue
424         cols.append(column)
425
426
427     results.columns = cols
428     results = results[results.age_group_name.isin(['All Ages','Under 5'])]
429
430     assert len(results.pathogen.unique()==27)
431     check_results(results, locs)
432     results.to_csv(f'FILEPATH', index=False)
```