

D:\NHRI\AMR\amr-main\explore\number_plugin_amr_results.py

```
1 import pandas as pd
2 import numpy as np
3 import seaborn as sns
4 from pathlib import Path
5 from db_tools import ezfuncs
6 from cod_prep.downloaders import *
7 from cod_prep.utils import *
8 from cod_prep.claude.configurator import Configurator
9 from amr_prep.utils.amr_io import get_amr_results
10 from amr_prep.utils.amr_io import AmrResult
11 from mcod_prep.compile_burden import summarize_draws
12 import warnings
13 warnings.filterwarnings("ignore")
14
15 CONF = Configurator()
16 LSV_ID = CONF.get_id('location_set_version')
17 CSV_ID = CONF.get_id('cause_set_version')
18
19 lh = get_current_location_hierarchy(location_set_version_id=LSV_ID)
20 ch = get_current_cause_hierarchy(cause_set_version_id=CSV_ID)
21
22
23 def get_results_wrapper(burden, measure, metric, syndrome=None,
24                         pathogen=None, abx_class=None, draws=False,
25                         location_id=1, counterfactual=None):
26     df = get_amr_results(
27         'summarize_burden',
28         burden,
29         year_id=2019,
30         cause_id=294,
31         location_id=location_id,
32         age_group_id=22,
33         infectious_syndrome=syndrome,
34         pathogen=pathogen,
35         abx_class=abx_class,
36         counterfactual=counterfactual,
37         measure_id=measure,
38         metric_id=metric,
39         sex_id=3,
40         hosp='all',
41         draws=draws,
42         filter_continuously=True
43     )
44     return df
45
46
47 def print_out_cf_lower_mean_upper(df, measure):
48     counterfactuals = df['counterfactual'].unique().tolist()
49
50     for counterfactual in counterfactuals:
51
```

```

52     lower = df.loc[df['counterfactual'] == counterfactual, 'amr_lower'].item()
53     mean = df.loc[df['counterfactual'] == counterfactual, 'amr_mean'].item()
54     upper = df.loc[df['counterfactual'] == counterfactual, 'amr_upper'].item()
55
56     if counterfactual == 'no_infection':
57         counterfactual = 'associated'
58     elif counterfactual == 'susceptible_infection':
59         counterfactual = 'attributable'
60
61     print(counterfactual + ' ' + measure + ': '
62           + str(mean) + ' (' + (str(lower) + ' - ' + str(upper)) + ')')
63
64
65 def aggregate_summarize_draws(df, to_aggregate, get_pct=False, denominator=None):
66     if isinstance(to_aggregate, str):
67         to_aggregate = [to_aggregate]
68     df.drop(columns=to_aggregate, inplace=True)
69     non_draw_cols = [col for col in df.columns if 'draw_' not in col]
70     draw_cols = [col for col in df.columns if 'draw_' in col]
71     df = df.groupby(non_draw_cols, as_index=False)[draw_cols].sum()
72
73     if get_pct:
74         assert (denominator is not None), 'If you would like to calculate pct, please pass
in denominator'
75         denominator.drop(columns=to_aggregate, inplace=True)
76         df = df.set_index(non_draw_cols)
77         denominator = denominator.set_index(non_draw_cols)
78         df = df.reorder_levels(non_draw_cols)
79         denominator = denominator.reorder_levels(non_draw_cols)
80         assert len(df) == len(denominator)
81         pct = df.div(denominator)
82         pct = pct.reset_index()
83
84     dfs = []
85     for adf in [df, pct]:
86         adf = adf.reset_index()
87         adf = summarize_draws(adf, draw_cols=draw_cols, prefix='amr_')
88         adf = adf.reset_index(drop=True)
89         adf.loc[
90             adf.counterfactual == 'susceptible_infection',
91             ['amr_mean', 'amr_upper', 'amr_lower']
92         ] = np.clip(adf[['amr_mean', 'amr_upper', 'amr_lower']], 0, None)
93         assert adf.notnull().values.all()
94         dfs.append(adf)
95     return dfs[0], dfs[1]
96 else:
97     df = summarize_draws(df, draw_cols=draw_cols, prefix='amr_')
98     df = df.reset_index(drop=True)
99     df.loc[
100         df.counterfactual == 'susceptible_infection',
101         ['amr_mean', 'amr_upper', 'amr_lower']
102     ] = np.clip(df[['amr_mean', 'amr_upper', 'amr_lower']], 0, None)
103     assert df.notnull().values.all()
104     return df

```

```

105
106
107 print('~~~~~')
108
109 for burden in ['fatal', 'nonfatal']:
110     print()
111     print('-----' + burden.upper() + '-----')
112     if burden == 'fatal':
113         measure='Deaths'
114         measure_id=1
115     elif burden == 'nonfatal':
116         measure='DALYs'
117         measure_id=2
118
119     print('Combo counts for fatal and nonfatal')
120     df = get_results_wrapper(burden, measure_id, 1, syndrome='all')
121     df = df.loc[~df['pathogen'].isin(['all', '(none_estimated)']), ]
122     df = df.loc[df['abx_class'] != '(none_tested)', ]
123     df.loc[df['pathogen'].str.contains('_escherichia'), 'pathogen'] = 'escherichia_coli'
124     print(burden + ' has ' + str(len(df[['pathogen']].drop_duplicates())) + ' pathogens')
125     print(df['pathogen'].unique().tolist())
126     df = df.loc[~df['abx_class'].str.contains('all_'), ]
127     print(burden + ' has ' + str(len(df[['pathogen', 'abx_class']].drop_duplicates())) + '
combos')
128     print(df[['pathogen', 'abx_class']].drop_duplicates())
129
130     print()
131     print('Total')
132     df = get_results_wrapper(burden, measure_id, 1, syndrome='all', pathogen='all',
abx_class='all_resistant')
133     print('Total AMR ' + measure + ':')
134     print_out_cf_lower_mean_upper(df, measure)
135
136     print()
137     print('Top 3 syndromes')
138     df = get_results_wrapper(burden, measure_id, 1, pathogen='all',
abx_class='all_resistant')
139     both_cfs = []
140     for counterfactual in ['no_infection', 'susceptible_infection']:
141         print(counterfactual)
142         print(df.loc[(df['infectious_syndrome'] != 'all')
& (df['counterfactual'] == counterfactual),
['infectious_syndrome', 'pathogen', 'amr_lower', 'amr_mean', 'amr_upper']]
.sort_values(by='amr_mean', ascending=False))
146         top3 = df.loc[(df['infectious_syndrome'] != 'all')
& (df['counterfactual'] == counterfactual),
].sort_values(by='amr_mean', ascending=False).head(3)
148         ['infectious_syndrome'].unique().tolist()
149         both_cfs.append(top3)
150     assert set(both_cfs[0]) == set(both_cfs[1])
151     df = get_results_wrapper(burden, measure_id, 1, syndrome=both_cfs[0], pathogen='all',
abx_class='all_resistant', draws=True)
152     total = get_results_wrapper(burden, measure_id, 1,
syndrome='all', pathogen='all', abx_class='all_resistant',

```

```

155         draws=True)
156     df, pct = aggregate_summarize_draws(df, to_aggregate='infectious_syndrome',
get_pct=True, denominator=total)
157     print('Top 3 syndromes AMR ' + measure + ':')
158     print_out_cf_lower_mean_upper(df, measure)
159     print('Top 3 syndromes AMR percentage out of total ' + measure + ':')
160     print_out_cf_lower_mean_upper(pct, measure)
161
162     print()
163     print('Top 6 pathogens')
164     df = get_results_wrapper(burden, measure_id, 1, syndrome='all',
abx_class='all_resistant')
165     both_cfs = []
166     for counterfactual in ['no_infection', 'susceptible_infection']:
167         print(counterfactual)
168         print(df.loc[(df['pathogen'] != 'all') & (df['counterfactual'] == counterfactual),
169             ['infectious_syndrome', 'pathogen', 'amr_lower', 'amr_mean', 'amr_upper']]
170             .sort_values(by='amr_mean', ascending=False))
171         top6 = df.loc[(df['pathogen'] != 'all') & (df['counterfactual'] == counterfactual),
172             ].sort_values(by='amr_mean', ascending=False).head(6)
173     ['pathogen'].unique().tolist()
174     both_cfs.append(top6)
175     df = get_results_wrapper(burden, measure_id, 1, syndrome='all', pathogen=both_cfs[0],
176     abx_class='all_resistant', draws=True)
177     total = get_results_wrapper(burden, measure_id, 1,
178     syndrome='all', pathogen='all', abx_class='all_resistant',
179     draws=True)
180     df, pct = aggregate_summarize_draws(df, to_aggregate='pathogen', get_pct=True,
denominator=total)
181     print('Top 6 pathogens AMR ' + measure + ':')
182     print_out_cf_lower_mean_upper(df, measure)
183     print('Top 6 pathogens AMR percentage out of total ' + measure + ':')
184     print_out_cf_lower_mean_upper(pct, measure)
185
186     print()
187     print('Top Combos')
188     df = get_results_wrapper(burden, measure_id, 1, syndrome='all')
189     both_cfs = []
190     for counterfactual in ['no_infection', 'susceptible_infection']:
191         print(counterfactual)
192         print(df.loc[(df['pathogen'] != 'all')
193             & (df['counterfactual'] == counterfactual)
194             & ~(df['abx_class'].str.contains('all_'))
195             & (df['abx_class'] != '(none_tested)'),
196             ['infectious_syndrome', 'pathogen', 'abx_class', 'amr_lower', 'amr_mean',
'amr_upper']]
197             .sort_values(by='amr_mean', ascending=False)
198             .head(20))
199
200     print()
201     print('Rate By Super Regions and Regions')
202     regions = lh.loc[lh['level'] <= 2, 'location_id'].unique().tolist()
203     df = get_results_wrapper(burden, measure_id, 3,
syndrome='all', pathogen=['all'],

```

```

204     abx_class=['all_resistant'], location_id=regions)
205 df = add_location_metadata(df, 'location_name')
206 df.loc[df.metric_id == 3, ['amr_mean', 'amr_lower', 'amr_upper']] = \
207     df[['amr_mean', 'amr_lower', 'amr_upper']] * 100_000
208 both_cfs = []
209 for counterfactual in ['no_infection', 'susceptible_infection']:
210     print(counterfactual)
211     pd.set_option('display.max_rows', 600)
212     print(df.loc[(df['counterfactual'] == counterfactual),
213         ['location_name', 'infectious_syndrome', 'pathogen', 'amr_lower', 'amr_mean',
214         'amr_upper']])
215         .sort_values(by='amr_mean', ascending=False))
216     print()
217
218 print('~~~~~')
219
220
221 print('MISC One-offs')
222 print()
223 print('Find S. Aureus, E. Coli in High-Income')
224 df = get_results_wrapper('fatal', 1, 1,
225     pathogen=['staphylococcus_aureus'], syndrome='all', abx_class='all_resistant',
226     location_id=64,
227     draws=True)
228 total = get_results_wrapper('fatal', 1, 1,
229     syndrome='all', pathogen='all', abx_class='all_resistant', location_id=64, draws=True)
230 df, pct = aggregate_summarize_draws(df, to_aggregate='pathogen', get_pct=True,
231     denominator=total)
232 print('S. Aureus AMR percentage out of total in High-Income deaths:')
233 print_out_cf_lower_mean_upper(pct, 'deaths')
234
235
236 df = get_results_wrapper('fatal', 1, 1,
237     pathogen=['escherichia_coli'], syndrome='all', abx_class='all_resistant',
238     location_id=64,
239     draws=True)
240 total = get_results_wrapper('fatal', 1, 1,
241     syndrome='all', pathogen='all', abx_class='all_resistant', location_id=64, draws=True)
242 df, pct = aggregate_summarize_draws(df, to_aggregate='pathogen', get_pct=True,
243     denominator=total)
244 print('E. Coli AMR percentage out of total in High-Income deaths:')
245 print_out_cf_lower_mean_upper(pct, 'deaths')
246
247 print()
248 print('Find S. pneumoniae, K. pneumoniae in sub-Saharan Africa')
249 df = get_results_wrapper('fatal', 1, 1,
250     pathogen=['streptococcus_pneumoniae'], syndrome='all', abx_class='all_resistant',
251     location_id=166,
252     draws=True)
253 total = get_results_wrapper('fatal', 1, 1,
254     syndrome='all', pathogen='all', abx_class='all_resistant', location_id=166, draws=True)
255 df, pct = aggregate_summarize_draws(df, to_aggregate='pathogen', get_pct=True,
256     denominator=total)

```

```

251 print('S. pneumoniae AMR percentage out of total in sub-Saharan Africa deaths:')
252 print_out_cf_lower_mean_upper(pct, 'deaths')
253
254
255 df = get_results_wrapper('fatal', 1, 1,
256     pathogen=['klebsiella_pneumoniae'], syndrome='all', abx_class='all_resistant',
257     location_id=166,
258     draws=True)
259 total = get_results_wrapper('fatal', 1, 1,
260     syndrome='all', pathogen='all', abx_class='all_resistant', location_id=166, draws=True)
261 df, pct = aggregate_summarize_draws(df, to_aggregate='pathogen', get_pct=True,
262     denominator=total)
263
264 print('K. pneumoniae AMR percentage deaths out of total in sub-Saharan Africa:')
265 print_out_cf_lower_mean_upper(pct, 'deaths')
266
267 print()
268 print('Check that fqns and beta lactams account over 70% burden attributable')
269 abxs = [
270     'anti_pseudomonal_penicillin',
271     'carbapenem',
272     'fourth_gen_ceph',
273     'third_gen_ceph',
274     'fluoroquinolone',
275     'aminopenicillin',
276     'penicillin',
277     'methicillin',
278     'beta_lactamase_inhibitor'
279 ]
280
281 df = get_results_wrapper('fatal', 1, 1,
282     syndrome='all', pathogen='all', abx_class=abxs, counterfactual='susceptible_infection',
283     draws=True)
284 print('check... ' + str(df['abx_class'].unique().tolist()))
285 total = get_results_wrapper('fatal', 1, 1,
286     syndrome='all', pathogen='all', abx_class='all_resistant',
287     counterfactual='susceptible_infection', draws=True)
288 df, pct = aggregate_summarize_draws(df, to_aggregate='abx_class', get_pct=True,
289     denominator=total)
290
291 print('flqs and beta lactams total attributable deaths out of total in global:')
292 print_out_cf_lower_mean_upper(pct, 'deaths')
293
294
295 overlap_4 = [
296     'mycobacterium_tuberculosis',
297     'staphylococcus_aureus',
298     'escherichia_coli',
299     'klebsiella_pneumoniae'
300 ]
301
302 df = get_results_wrapper('fatal', 1, 1, syndrome='all', pathogen=overlap_4,
303     counterfactual='susceptible_infection', draws=True, location_id=[42, 73])
304 df = aggregate_summarize_draws(df, to_aggregate=['pathogen', 'abx_class', 'location_id'])
305 print_out_cf_lower_mean_upper(df, 'deaths')
306
307

```

```

300
301 print()
302
303 print('Find the total burden attributable for the 11 combos that overlap with cassini')
304
305 cassini_overlap = [
306     ['carbapenem', 'acinetobacter_baumannii'],
307     ['vancomycin', 'enterococcus_faecalis'],
308     ['vancomycin', 'enterococcus_faecium'],
309     ['carbapenem', 'escherichia_coli'],
310     ['third_gen_ceph', 'escherichia_coli'],
311     ['carbapenem', 'klebsiella_pneumoniae'],
312     ['third_gen_ceph', 'klebsiella_pneumoniae'],
313     ['carbapenem', 'pseudomonas_aeruginosa'],
314     ['methicillin', 'staphylococcus_aureus'],
315     ['penicillin', 'streptococcus_pneumoniae'],
316     ['macrolide', 'streptococcus_pneumoniae'],
317 ]
318
319 overlap_pathogens = [
320     'acinetobacter_baumannii',
321     'enterococcus_faecalis',
322     'enterococcus_faecium',
323     'escherichia_coli',
324     'klebsiella_pneumoniae',
325     'pseudomonas_aeruginosa',
326     'staphylococcus_aureus',
327     'streptococcus_pneumoniae',
328 ]
329
330 df = get_results_wrapper('fatal', 1, 1, syndrome='all', pathogen=overlap_pathogens,
331     counterfactual='susceptible_infection', draws=True, location_id=[42, 73])
332
333 dfs = []
334 for combo in cassini_overlap:
335     adf = df.loc[(df['abx_class'] == combo[0]) & (df['pathogen'] == combo[1]), ]
336     dfs.append(adf)
337 df = pd.concat(dfs)
338 print('Total deaths attributable for 11 combos overlap with Cassini in Central and Western
Europe')
339
340 df = aggregate_summarize_draws(df, to_aggregate=['pathogen', 'abx_class', 'location_id'])
341 print_out_cf_lower_mean_upper(df, 'deaths')
342
343 df = get_results_wrapper('nonfatal', 2, 1, syndrome='all', pathogen=overlap_pathogens,
344     counterfactual='susceptible_infection', draws=True, location_id=[42, 73])
345
346 dfs = []
347 for combo in cassini_overlap:
348     adf = df.loc[(df['abx_class'] == combo[0]) & (df['pathogen'] == combo[1]), ]
349     dfs.append(adf)
350 df = pd.concat(dfs)
351 print('Total DALYs attributable for 11 combos overlap with Cassini in Central and Western
Europe')

```

```
352 df = aggregate_summarize_draws(df, to_aggregate=['pathogen', 'abx_class', 'location_id'])
353 print_out_cf_lower_mean_upper(df, 'DALYs')
354
355
356 print()
357 print('check cardiac compared to total associated')
358 df = get_results_wrapper('fatal', 1, 1, syndrome='cardiac_infectious', pathogen='all',
359   abx_class='all_resistant',
360   counterfactual='no_infection', draws=True)
361 total = get_results_wrapper('fatal', 1, 1, syndrome='all', pathogen='all',
362   abx_class='all_resistant',
363   counterfactual='no_infection', draws=True)
364 df, pct = aggregate_summarize_draws(df, to_aggregate='infectious_syndrome', get_pct=True,
365   denominator=total)
366 print_out_cf_lower_mean_upper(pct, 'deaths')
```