0_database-generation

February 14, 2020

1 Database generation

In this notebook we process the data into a database we can later query to make plots/do analysis etc. ### OBS: This notebook is only slightly edited from Zebedee Nicholls notebook, see here

1.1 Imports

[1]:
```
from ar6_ch6_rcmipfigs.constants import INPUT_DATA_DIR

__depends__ = []
__dest__ = [INPUT_DATA_DIR+
    "/data/database-results/phase-1/timestamp.txt",
    INPUT_DATA_DIR+"/data/database-observations/timestamp.txt",
]
```

```
/home/sarambl/PHD/IPCC/public/AR6_CH6_RCMIPFIGS/ar6_ch6_rcmipfigs
/home/sarambl/PHD/IPCC/public/AR6_CH6_RCMIPFIGS/ar6_ch6_rcmipfigs/data_in
```

[2]:
```
%load_ext nb_black
```

```
<IPython.core.display.Javascript object>
```

[ ]:
```
import logging
import os.path
import re
from pathlib import Path
from distutils.util import strtobool

import pandas as pd
import tqdm
from scmdata import ScmDataFrame, df_append
```

[ ]:
```
from ar6_ch6_rcmipfigs.utils.database_generation import␣
↪check_all_variables_and_units_as_in_protocol, \
    check_all_scenarios_as_in_protocol, unify_units, save_into_database,␣
↪mce_get_quantile, hector_get_quantile
```

```
TEST_RUN = strtobool(os.getenv("CI", "False")) or False
TEST_RUN
```

[ ]:
```
logger = logging.getLogger()
```

1.2 Constants

[ ]:
```
from ar6_ch6_rcmipfigs.constants import  INPUT_DATA_DIR
OUTPUT_DATABASE_PATH = os.path.join(INPUT_DATA_DIR, "database-results",␣
 ↪"phase-1/")

OBS_DATABASE_PATH = os.path.join(INPUT_DATA_DIR, "database-observations/")
```

[ ]:
```
from ar6_ch6_rcmipfigs.utils.misc_func import make_folders

if not os.path.isdir(OUTPUT_DATABASE_PATH):
    make_folders(OUTPUT_DATABASE_PATH)

if not os.path.isdir(OBS_DATABASE_PATH):
    make_folders(OBS_DATABASE_PATH)
```

1.3 Protocol

[ ]:
```
SCENARIO_PROTOCOL = os.path.join(INPUT_DATA_DIR, "data", "protocol",␣
 ↪"rcmip-emissions-annual-means.csv"
)
```

[ ]:
```
protocol_db = ScmDataFrame(SCENARIO_PROTOCOL)
protocol_db.head()
```

[ ]:
```
protocol_db["scenario"].unique()
```

[ ]:
```
DATA_PROTOCOL = os.path.join(INPUT_DATA_DIR,
    "data",
    "submission-template",
    "rcmip-data-submission-template.xlsx",
)
```

[ ]:
```
protocol_variables = pd.read_excel(DATA_PROTOCOL,␣
 ↪sheet_name="variable_definitions")
protocol_variables.columns = protocol_variables.columns.str.lower()
protocol_variables.head()
```

[ ]:
```
protocol_scenarios = pd.read_excel(
    DATA_PROTOCOL, sheet_name="scenario_info", skip_rows=2
)
protocol_scenarios.columns = protocol_scenarios.columns.str.lower()
protocol_scenarios.head()
```

1.4 Model output

[ ]:
```
RESULTS_PATH = os.path.join(INPUT_DATA_DIR, "data", "results", "phase-1")
```

[ ]:
```
_results_files = list(Path(RESULTS_PATH).rglob("*.csv")) + list(
    Path(RESULTS_PATH).rglob("*.xlsx")
)
print(len(_results_files))
sorted(_results_files)
```

[ ]:
```
model_of_interest = [
#     ".*acc2.*v2-0-1.*",
    ".*rcmip_phase-1_cicero-scm.*v5-0-0.*",
#     ".*escimo.*v2-0-1.*",
    ".*fair-1.5-default.*v1-0-1.csv",
#     ".*rcmip_phase-1_gir.*",
#     ".*greb.*v2-0-0.*",
#     ".*hector.*v2-0-0.*",
#     ".*MAGICC7.1.0aX-rcmip-phase-1.*",
    ".*rcmip_phase-1_magicc7.1.0.beta_v1-0-0.*",
#     ".*MAGICC7.1.0aX.*",
#     ".*mce.*v2-0-1.*",
#     ".*oscar-v3-0*v1-0-1.*",
    ".*oscar-v3-0.*v1-0-1.*"
#     ".*wasp.*v1-0-1.*",
]
```

[ ]:
```
if TEST_RUN:
    model_of_interest = [
        ".*escimo-phase-1-v2-0-1.*",
        ".*greb.*",
        ".*rcmip_phase-1_cicero-scm.*v5-0-0.*",
    ]

results_files = [
    str(p)
    for p in _results_files
    if any([bool(re.match(m, str(p))) for m in model_of_interest]) and "$" not␣
 ↪in str(p)
```

```
]
print(len(results_files))
sorted(results_files)
```

[ ]:
```
[
    str(p)
    for p in results_files
    if 'magicc' in str(p)] #for m in model_of_interest]) and "$" not in str(p)
#]
```

[ ]:
```
db = []
for rf in tqdm.tqdm_notebook(results_files):
    if rf.endswith(".csv"):
        loaded = ScmDataFrame(rf)
    else:
        loaded = ScmDataFrame(rf, sheet_name="your_data")
    db.append(loaded)

db = df_append(db).timeseries().reset_index()
db["unit"] = db["unit"].apply(
    lambda x: x.replace("Dimensionless", "dimensionless") if isinstance(x, str)␣
 ↪else x
)
db = ScmDataFrame(db)
db.head()
```

[ ]:
```
db.filter(climatemodel="*cicero*").head()
```

[ ]:
```
db["climatemodel"].unique()
```

1.4.1 Minor quick fixes

We relabel all the ssp370-lowNTCF data to remove ambiguity.

[ ]:
```
db = db.timeseries().reset_index()
db["scenario"] = db["scenario"].apply(
    lambda x: "ssp370-lowNTCF-gidden" if x == "ssp370-lowNTCF" else x
)
db["scenario"] = db["scenario"].apply(
    lambda x: "esm-ssp370-lowNTCF-gidden" if x == "esm-ssp370-lowNTCF" else x
)
db["scenario"] = db["scenario"].apply(
    lambda x: "esm-ssp370-lowNTCF-gidden-allGHG"
    if x == "esm-ssp370-lowNTCF-allGHG"
    else x
)
```

```
db = ScmDataFrame(db)
```

[ ]:
```
assert "ssp370-lowNTCF" not in db["scenario"].unique().tolist()
assert "esm-ssp370-lowNTCF" not in db["scenario"].unique().tolist()
assert "esm-ssp370-lowNTCF-allGHG" not in db["scenario"].unique().tolist()
```

The Hector and MCE data is mislabelled so we do a quick fix here. I also have changed my mind about how to format the quantiles so tweak the FaIR and WASP data too.

[ ]:
```
mce_prob_data = db.filter(climatemodel="MCE*PROB*")
mce_prob_data["climatemodel"].unique()
if not mce_prob_data.timeseries().empty:
    mce_prob_data = mce_prob_data.timeseries().reset_index()

    mce_prob_data["variable"] = (
        mce_prob_data["variable"]
        + "|"
        + mce_prob_data["climatemodel"].apply(mce_get_quantile)
        + "th quantile"
    )

    mce_prob_data["climatemodel"] = mce_prob_data["climatemodel"].apply(
        lambda x: "-".join(x.split("-")[:-1])
    )

    db = db.filter(climatemodel="MCE*PROB*", keep=False).append(mce_prob_data)

db.filter(climatemodel="MCE*PROB").head(10)
```

[ ]:
```
hector_prob_data = db.filter(climatemodel="hector*HISTCALIB*")
if not hector_prob_data.timeseries().empty:
    hector_prob_data = hector_prob_data.timeseries().reset_index()

    hector_prob_data["variable"] = (
        hector_prob_data["variable"]
        + "|"
        + hector_prob_data["climatemodel"].apply(hector_get_quantile)
    )

    hector_prob_data["climatemodel"] = hector_prob_data["climatemodel"].apply(
        lambda x: x.split("-")[0]
    )

    db = db.filter(climatemodel="hector*HISTCALIB*", keep=False).append(
        hector_prob_data
    )
```

```
db.filter(climatemodel="hector*HISTCALIB").head(10)
```

[ ]:
```
fair_prob_data = db.filter(climatemodel="*FaIR*")
if not fair_prob_data.timeseries().empty:
    fair_prob_data = fair_prob_data.timeseries().reset_index()

    fair_prob_data["variable"] = fair_prob_data["variable"].apply(
        lambda x: x.replace("|00th", "|0th").replace("|05th", "|5th")
    )

    db = db.filter(climatemodel="*FaIR*", keep=False).append(
        ScmDataFrame(fair_prob_data)
    )

db.filter(climatemodel="*FaIR*").head(10)
```

[ ]:
```
wasp_prob_data = db.filter(climatemodel="*WASP*")
if not wasp_prob_data.timeseries().empty:
    wasp_prob_data = wasp_prob_data.timeseries().reset_index()

    wasp_prob_data["variable"] = wasp_prob_data["variable"].apply(
        lambda x: x.replace("|00th", "|0th").replace("|05th", "|5th")
    )

    db = db.filter(climatemodel="*WASP*", keep=False).append(
        ScmDataFrame(wasp_prob_data)
    )

db.filter(climatemodel="*WASP*").head(10)
```

1.5 Unify units and check names

Here we loop over the submissions and unify their units as well as checking their naming matches what we expect.

[ ]:
```
base_df = db.timeseries()
any_failures = False

clean_db = []
for climatemodel, cdf in tqdm.tqdm_notebook(
    base_df.groupby("climatemodel"), desc="Climate model"
):
    print(climatemodel)
    print("-" * len(climatemodel))

    any_failures_climatemodel = False
```

```
    cdf = ScmDataFrame(cdf)
    cdf_converted_units = unify_units(cdf, protocol_variables)
    try:
        check_all_scenarios_as_in_protocol(cdf_converted_units,␣
↪protocol_scenarios)
        check_all_variables_and_units_as_in_protocol(
            cdf_converted_units, protocol_variables
        )
    except AssertionError as e:
        print(e)
        any_failures_climatemodel = True
    #     # currently not possible as groups weren't told to obey variable␣
↪hierarchy,
    #     # add this in phase 2
    #     for v_top in cdf_converted_units.filter(level=0)["variable"].unique():
    #         print(v_top)
    #         cdf_pyam = cdf_converted_units.filter(variable="{}*".
↪format(v_top)).timeseries()
    #         cdf_pyam.columns = cdf_pyam.columns.map(lambda x: x.year)

    #         cdf_consistency_checker = pyam.IamDataFrame(cdf_pyam)
    #         if cdf_consistency_checker.check_internal_consistency() is not␣
↪None:
    #             print("Failed for {}".format(v_top))
    #             any_failures_climatemodel = True
    #             failing_set = cdf_consistency_checker.copy()

    print()
    if not any_failures_climatemodel:
        clean_db.append(cdf_converted_units)
        print("All clear for {}".format(climatemodel))
    else:
        print("Failed {}".format(climatemodel))
        print("X" * len("Failed"))
        any_failures = True

    print()
    print()

if any_failures:
    raise AssertionError("database isn't ready yet")
else:
    clean_db = df_append(clean_db)
    clean_db.head()
```

[ ]:
```
clean_db.head()
```

Notes whilst doing this:

- I wasn't clear that the variable hierarchy needs to be obeyed, hence doing internal consistency checks isn't going to work

For phase 2:

- checking internal consistency super slow, worth looping over top level variables when doing this to speed up filtering
- need to decide what a sensible tolerance is
- might have to go back to model notes to work out why there are inconsistencies
- will have to implement a custom hack to deal with the double counting in the direct aerosol forcing hierarchy

1.6 Creating a database

[ ]:
```
save_into_database(clean_db, OUTPUT_DATABASE_PATH, "rcmip-phase-1")
```