Step 6: Serve new imported input data from OA into Excel then GDX through WaMDaM

By Adel M. Abdallah, Jan 2022

Execute the following cells by pressing `Shift-Enter`, or by pressing the play button on the toolbar above.

Overview

You'll use the downloaded models (now in SQLite WaMDaM database) and this Jupyter Notebook to run both WEAP and Wash models for the new scenarios defined in OpenAgua GUI. I published the SQLite WaMDaM database in HydroShare so its easier to query it in this script directly from there.

The script here will also read the models' results and visualize them or export the results to CSV or Excel file that can be loaded back to WaMDaM. Later in the next steps, those results can be uploaded back to OpenAgua to view them there in a dashboard.

You will need to have both WEAP and GAMS software installed on your machine

1. Import python libraries

In [ ]:

```
# 1. Import python libraries
### set the notebook mode to embed the figures within the cell

import sqlite3
import numpy as np
import pandas as pd
import getpass
from hs_restclient import HydroShare, HydroShareAuthBasic
import os

import plotly
plotly.__version__
import plotly.offline as offline
import plotly.graph_objs as go
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
offline.init_notebook_mode(connected=True)
from plotly.offline import init_notebook_mode, iplot
from plotly.graph_objs import *


init_notebook_mode(connected=True)         # initiate notebook for offline plot
import sys
from shutil import copyfile

import os
import csv
from collections import OrderedDict
import sqlite3
import pandas as pd
import numpy as np
from IPython.display import display, Image, SVG, Math, YouTubeVideo

import urllib.request as url

import calendar

import sqlite3
import pandas as pd
# https://github.com/NREL/gdx-pandas
from IPython.display import display, Image, SVG, Math, YouTubeVideo
from openpyxl import load_workbook
import logging
import inspect
import gdxpds

# ! pip install gdxpds
# import gdxpds
# import gams
# from gams import *
#from ctypes import c_bool
print ('The needed Python libraries have been imported')
```

2. Connect to the WaMDaM SQLite on HydroShare

Provide the HydroShare ID for your resource

Example
https://www.hydroshare.org/resource/af71ef99a95e47a89101983f5ec6ad8b/

In [ ]:

```
# enter your HydroShare username and password here between the quotes

username = ''
password = ''

auth = HydroShareAuthBasic(username=username, password=password)

hs = HydroShare(auth=auth)


# Then we can run queries against it within this notebook :)
resource_url='https://www.hydroshare.org/resource/af71ef99a95e47a89101983f5ec6ad8b/'


resource_id= resource_url.split("https://www.hydroshare.org/resource/",1)[1]
resource_id=resource_id.replace('/','')

print (resource_id)

print ('Connected to HydroShare')

resource_md = hs.getSystemMetadata(resource_id)
# print resource_md

print ('Resource title')
print(resource_md['resource_title'])
print ('----------------------------')

resources=hs.resource(resource_id).files.all()

file = ""

for f in hs.resource(resource_id).files.all():

    file += f.decode('utf8')

import json

file_json = json.loads(file)

for f in file_json["results"]:

    FileURL= f["url"]

    SQLiteFileName=FileURL.split("contents/",1)[1]

cwd = os.getcwd()
print (cwd)

fpath = hs.getResourceFile(resource_id, SQLiteFileName, destination=cwd)
conn = sqlite3.connect(SQLiteFileName,timeout=10)

print ('done')
# Test if the connection works
conn = sqlite3.connect(SQLiteFileName)

df = pd.read_sql_query("SELECT ResourceTypeAcronym   FROM ResourceTypes Limit 1 ", conn)
print (df)

print ('--------------------')
print ('\n Connected to the WaMDaM SQLite file called' + ': ' + SQLiteFileName)
```

Query the seasonal data from WaMDaM into a dataframe

In [ ]:

```
# The query has hard coded input parameters

import requests

Query_UseCase_URL='https://raw.githubusercontent.com/WamdamProject/WaMDaM_JupyterNotebooks/master/3_Visualize

# Read the query text inside the URL

page = requests.get(Query_UseCase_URL)

Query_UseCase_text=page.text

# # return query result in a pandas data frame
df_Seasonal_WaMDaM= pd.read_sql_query(Query_UseCase_text, conn)


display (df_Seasonal_WaMDaM)
```

Make copies of the origianl input file to the WASH Model

In [ ]:

```
# Demand conservation file
copyfile("WASH_1yr_InputData_original.xlsx", "WASH_1yr_InputData_conserve.xlsx")

# Demand Increase file
copyfile("WASH_1yr_InputData_original.xlsx", "WASH_1yr_InputData_increase.xlsx")
```

Preapre input data and write it to Excel input to WASH

In [ ]:

```
Instance=df_Seasonal_WaMDaM['InstanceName'][1]
print (Instance)

column_name = ["ScenarioName"]
subsets = df_Seasonal_WaMDaM.groupby(column_name)

for subset in subsets.groups.keys():
    dt = subsets.get_group(name=subset)
    print (subset)

    df=dt.loc[:, 'SeasonName':'SeasonNumericValue'].T


    if subset=='ConsDemand':
        WASH_ExcelFile='WASH_1yr_InputData_conserve.xlsx'
        df=df.loc['SeasonNumericValue':, ]
        display(df)

    elif subset=='IncrDemand':
        WASH_ExcelFile='WASH_1yr_InputData_increase.xlsx'

        df=df.loc['SeasonNumericValue':, ]
        columns = dict(map(reversed, enumerate(df.columns)))
        df = df.rename(columns=columns)
        df.head()
        display(df)
    else:
        continue

    sheetname='demandReq'
    #
    book = load_workbook(WASH_ExcelFile)

    UpdateDemand = book[sheetname]
    i =0
    for index, row in df.iterrows():
        for j, column in row.iteritems():
            UpdateDemand.cell(row=i+2, column=j+2, value=float(column))
        i += 1
    book.save(WASH_ExcelFile)

    print ('Done writing the value to WASH excel file')
```

Follow these steps to run the WASH Models in GAMS

1. Download and install GAMS V.24.2.3 and have Excel 2007 onward

You will need a license to run GAMS solvers.

2. Open GAMS software. Go to File-> Project -> New Project. Navigate to the folder you created and create a new project file with any name you want.

3. Double click on the gams file name: WASH-WaMDaM.gms to open the GAMS file

4. Comment out the lines in 218-230 to choose one input file at a time. Then comment out the lines in 590-597 to write the results to the GDX file one at a time

WASH_1yr_InputData_original.xlsx

WASH_1yr_InputData_conserve.xlsx

WASH_1yr_InputData_increase.xlsx

WASH-solution-original.gdx

WASH-solution-conserve.gdx

WASH-solution-increase.gdx

Read the WASH Area result objective function value from GDX

First, read the three result .gdx files.

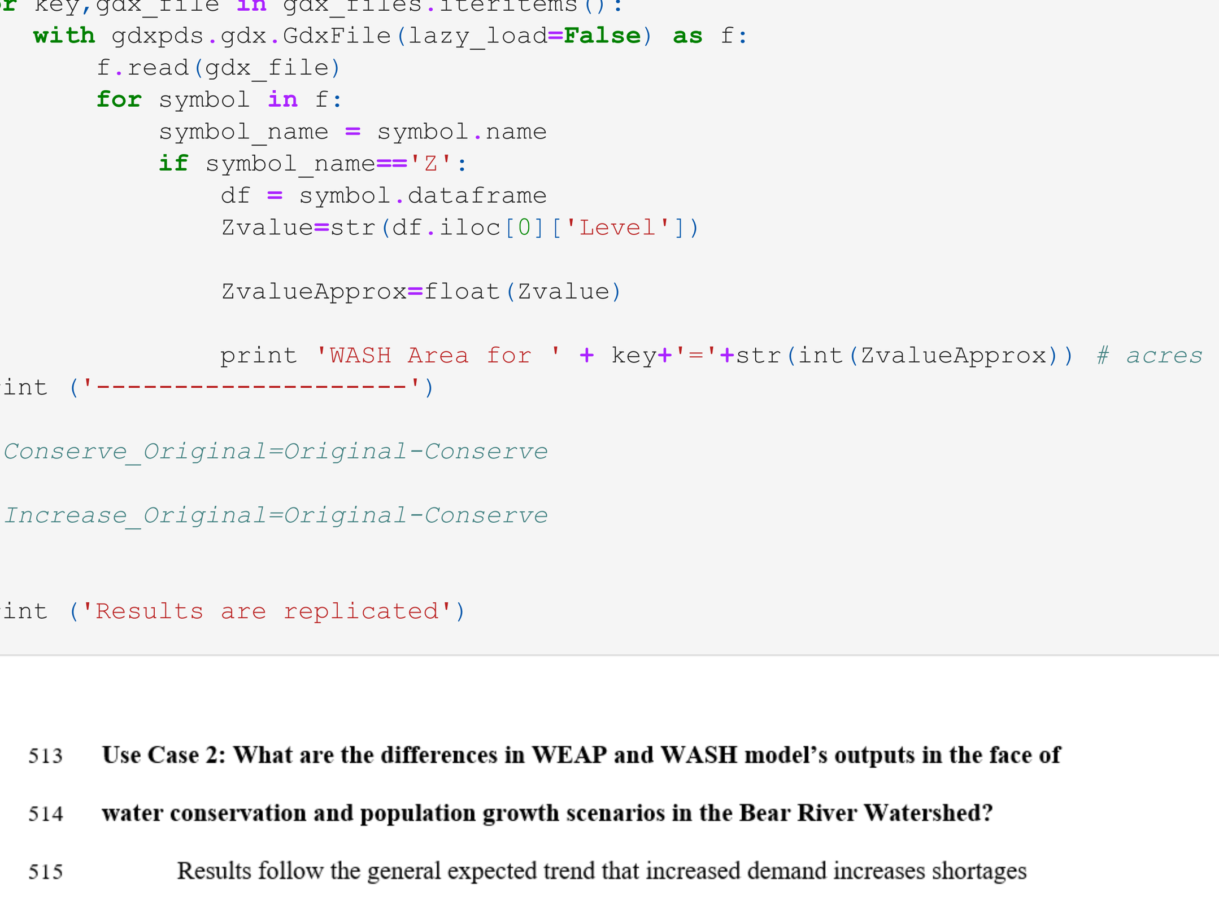
If you have issues in running GAMS scenarios, I already posted the soultion .gdx files here
https://github.com/WamdamProject/WaMDaM_JupyterNotebooks/tree/master/3_VisualizePublish

In [ ]:

```
os.getcwd()
```

In [ ]:

```
gdx_files = {'Original':'WASH-solution-original.gdx',
             'Conserve':'WASH-solution-conserve.gdx',
             'Increase':'WASH-solution-increase.gdx'}


for key,gdx_file in gdx_files.iteritems():
    with gdxpds.gdx.GdxFile(lazy_load=False) as f:
        f.read(gdx_file)
        for symbol in f:
            symbol_name = symbol.name
            if symbol_name=='Z':
                df = symbol.dataframe
                Zvalue=str(df.iloc[0]['Level'])

                ZvalueApprox=float(Zvalue)

                print 'WASH Area for ' + key+'='+str(int(ZvalueApprox)) # acres
print ('--------------------')

# Conserve_Original=Original-Conserve

# Increase_Original=Original-Conserve


print ('Results are replicated')
```



Use Case 2: What are the differences in WEAP and WASH model's outputs in the face of water conservation and population growth scenarios in the Bear River Watershed?

Results follow the general expected trend that increased demand increases shortages while water conservation reduces shortages (Figure 6). There are four years, 1970, 1976, 1993, 1996, where water conservation completely eliminates shortages while shortages persist for the base case and increased demand scenario. In dryer years (e.g., 1987 to 1992 and 2000 to 2004 where there is not enough water to meet site demand), the conservation scenario reduces the magnitude of shortages compared to the baseline scenario. These results are also available online in OpenAgua for stakeholders to view and discuss.

For the WASH model, the watershed area for suitable habitat for native vegetation, birds and fish in the baseline scenario (2003 hydrologic year) is estimated at 121,526 acres. Reducing Cache County urban demand by 25% would increase the WASH area by 144 acres while a 25% increase in the site's demand would decrease the WASH area by 142 acres.

This small increase or decrease in WASH area is because of the small influence of Cache County urban site which represents about 18% of the total annual agriculture and urban demand in this watershed of 415 million cubic meters (336,446 acre-feet). These results show the potential role of targeted urban water conservation and growth in improving or degrading suitable habitat areas in the watershed.

The End :) Congratulations!

The code below is part of the trial to run GAMS from here but didnt work for me. Feel free to give it a try

Execute GAMS (Before and after update)

https://www.gams.com/latest/docs/API_PY_TUTORIAL.html

In [ ]:

```
import os
# os.path.dirname(os.path.abspath(__file__))

command="""start cmd cd C:\GAMS\win64/24.7 & gams.exe WASH-CEE6410"""

var=os.system(command)
print var
```

In [ ]:

```
from gams import *
```

In [ ]:

```
! conda install -c goop gams
```

In [ ]:

```
version = GamsWorkspace.api_version
print version
ws = GamsWorkspace('Test')
# ws = GamsWorkspace(debug=DebugLevel.KeepFiles)
print ws
job = ws.add_job_from_file("C:\Users\Adel\Documents\GitHub\WEAP_WASH_OA\WASH-CEE6410.gms")
job.run()
GamsDatabase = job.out_db
```

In [ ]:

```
ws = GamsWorkspace()
job = ws.add_job_from_file("C:\Users\Adel\Documents\GitHub\WEAP_WASH_OA\WASH-CEE6410.gms")
job.run()
GamsDatabase = job.out_db
```

In [ ]:

```
if len(sys.argv) > 1:
    ws = GamsWorkspace(system_directory = sys.argv[1])
else:
    ws = GamsWorkspace()
```

In [ ]:

```
GAMSWorkspace.GAMSWorkspace(workingDirectory = null,
systemDirectory = null,
DebugLevel  = DebugLevel.Off
)
```