Step 9: Query data for Monterrey Mexico from OpenAgua published in HydroShare

By Adel M. Abdallah, Jan 2022

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

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 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
import calendar
# import datetime
from datetime import date
print 'The needed Python libraries have been imported'
```

2. Connect to the WaMDaM SQLite on HydroSahre

Provide the HydroShare ID for your resource

Example
https://www.hydroshare.org/resource/e29c9283305045338be24a495c781ec9/

In [ ]:

```
# enter your HydroShare user name
username = 'amabdallah'

# enter your HydroShare password
password = 'HydroShare123'

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

hs = HydroShare(auth=auth)

print 'Connected to HydroShare'

# Then we can run queries against it within this notebook :)
resource_url='https://www.hydroshare.org/resource/e29c9283305045338be24a495c781ec9/'


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

print resource_id

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 'Connected to the SQLite file= '+ SQLiteFileName
print 'done'
```

Query delivery target and obseved flow at DR Bajo Rio San Juan demand site

The data comes from OpenAgua

In [ ]:

```
# Use Case 3.1Identify_aggregate_TimeSeriesValues.csv
# plot aggregated to monthly and converted to acre-feet time series data of multiple sources



# 2.2Identify_aggregate_TimeSeriesValues.csv
Query_UseCase3_1_URL="""
https://raw.githubusercontent.com/WamdamProject/WaMDaM_JupyterNotebooks/master/3_VisualizePublish/SQL_queries/
"""

# Read the query text inside the URL
Query_UseCase3_1_text = urllib.urlopen(Query_UseCase3_1_URL).read()


# return query result in a pandas data frame
result_df_UseCase3_1= pd.read_sql_query(Query_UseCase3_1_text, conn)

# uncomment the below line to see the list of attributes
display (result_df_UseCase3_1)

# print result_df_UseCase3_1.keys()
print "Query is done"
```

Connect to the WaMDaM SQLite on HydroSahre

Provide the HydroShare ID for your resource

In [ ]:

```
# provide your HydroShare credentials

username = 'amabdallah'
password = 'HydroShare123'

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

hs = HydroShare(auth=auth)

print 'Connected to HydroShare'

# 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

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 demand data at the Logan Irrigation site

In [ ]:

```
# Use Case 3.1Identify_aggregate_TimeSeriesValues.csv
# plot aggregated to monthly and converted to acre-feet time series data of multiple sources



# 2.2Identify_aggregate_TimeSeriesValues.csv
Query_UseCase3_URL="""
https://raw.githubusercontent.com/WamdamProject/WaMDaM_JupyterNotebooks/master/3_VisualizePublish/SQL_queries/
"""

# Read the query text inside the URL
Query_UseCase3_text = urllib.urlopen(Query_UseCase3_URL).read()


# return query result in a pandas data frame
result_df_UseCase3= pd.read_sql_query(Query_UseCase3_text, conn)

# uncomment the below line to see the list of attributes
# display (result_df_UseCase3)

# print result_df_UseCase3_1.keys()
print "Query is done"

# generate time series

month=result_df_UseCase3['SeasonName']
SeasonNumericValue=result_df_UseCase3['SeasonNumericValue']

# print month
# print SeasonNumericValue

# 1 (acre foot) per month =0.00046936 cubic meter per second
# val=result_df_UseCase3['SeasonNumericValue']*0.00046936
# print val
# 1990-2014
result_data=OrderedDict()
result_data = {'date':[], 'value':[]}
day = 1
for year in range(2000,2005):
    # generate a date by using the year here and the month from the list. The day is always 1
    a=[10,11,12,1,2,3,4,5,6,7,8,9]
    b=range (12)
    for month_int,indx in zip(a, b):
        # Create two dates
        if month_int in [10,11,12]:
            calyear=year
        else:
            calyear=year+1

        date_val = date(calyear, month_int, day)


        Value=  SeasonNumericValue[indx]# get the value for the month
        print date_val
        print Value
#         print month
#         print Value
        result_data['date'].append(date_val)
        result_data['value'].append(float(Value)*0.00046936)


# display(result_data)
```

Plot and compare demand and observed delievery for the baseline and calibration scenariosin Monterrey model

In [ ]:

```
df_TimeSeries=result_df_UseCase3_1
# identify the data for four time series only based on the DatasetAcronym column header
column_name = ["ScenarioName","AttributeName"]
subsets = df_TimeSeries.groupby(column_name)
data = []

subsets_settings = {
    ('Baseline','Observed Delivery'): {
        'dash': 'solid',
        'legend_index': 0,
        'legend_name': 'Baseline: Observed Delivery',
        'width':2,
        'color':'rgb(0, 0, 0)',
        'size':10,
        'symbol':'square'

        },
    ('Baseline','Demand'): {
        'dash': 'solid',
        'legend_index': 2,
        'legend_name': 'DR Bajo Rio San Juan, Mexico (OpenAqua)',
        'width':2,
        'color':'#0099CC',
        'size':10,
        'symbol':'square'

        },

    }

# This dict is used to map legend_name to original subset name
subsets_names = {y['legend_name']: x for x,y in subsets_settings.iteritems()}

for subset in subsets.groups.keys():
    if subset== ('Calibration','Demand') or subset == ('Calibration','Observed Delivery') or subset==('Baselir

        continue
    dt = subsets.get_group(name=subset)
    print subset

    s = go.Scatter(
                    x=dt['DateTimeStamp'],
                    y=dt['DataValue'],
                    mode='lines+markers',
                    marker=dict(size=10 ),

                    name = subsets_settings[subset]['legend_name'],

                        line = dict(
                        color =subsets_settings[subset]['color'],
                        width =subsets_settings[subset]['width'],
                        dash=subsets_settings[subset]['dash']
                               ),
                        opacity = 1
                   )
    data.append(s)



data.sort(key=lambda x: subsets_settings[subsets_names[x['name']]]['legend_index'])




utah = go.Scatter(

      x=result_data['date'],
      y=result_data['value'],
      name = 'Logan Irrigation, Utah (WEAP)',
    mode='lines+markers',
    marker=dict(symbol='square',size=10),
    line = dict(color ='#333333')
                   )
data.append(utah)


layout = dict(
#title = "Use Case 3.3",
yaxis = dict(
    title = "Delivery target <br> (cubic meter/second)",            automargin=True,

#     tickformat= ',',

        dtick='0.5',
        ticks='outside',
        ticklen=10,

            range = ['0', '4'],
            showline=True,
            linewidth=1, linecolor='black',

            zerolinecolor='#00000f',
            tickcolor='#00000f',
            showgrid=True,
            gridcolor='#dddddd'),

xaxis = dict(
#         title = "Updated input parameters in the <br>Bear_River_WEAP_Model_2017",

    ticks='outside',
    automargin=True,
         range = ['2001-11-01','2005'],
                tick0='2000-01-01',
         zeroline=False,
         zerolinecolor='#00000f',
         zerolinewidth=4,
         tickcolor='#00000f',
         showgrid=True,
    gridcolor='#dddddd',
        showline=True,

#          tickfont=dict(size=22),
        ticklen=20
                ),
legend=dict(
        x=0.2,y=1.2,
      bordercolor='#00000f',
        borderwidth=2, traceorder="reversed"),
    width=1100,
    height=700,

 margin=dict(l=200,pad=4),
font=dict(size=25,family='arial',color='#00000f'),
showlegend=True,
#     paper_bgcolor='rgba(0,0,0,0)',
    plot_bgcolor='#FFFFFF'

)
# create a figure object
fig = dict(data=data, layout=layout)
#py.iplot(fig, filename = "2.3Identify_SeasonalValues")

import plotly.express as px

## it can be run from the local machine on Pycharm like this like below
## It would also work here offline but in a seperate window
offline.iplot(fig,filename = 'Monterrey' )
# fig.update_xaxes(showline=True, linewidth=2, linecolor='black')
# fig.update_yaxes(showline=True, linewidth=2, linecolor='black')
print "Figure x is replicated!!"
```

7. Close the SQLite connection

In [ ]:

```
# 9. Close the SQLite and WEAP API connections
conn.close()

print 'connection disconnected'
```

The End :) Congratulations!