08_Step8_Run Bear and Weber WEAP Models. Estimate System's Reliability due to changes in input

By Adel M. Abdallah, Feb 2022

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

1. Import python libraries

```python
# 1. Import python libraries
### set the notebook mode to embed the figures within the cell
import numpy
import sqlite3
import numpy as np
import pandas as pd
import getpass
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

print 'The needed Python libraries have been imported'
```

2. Run WEAP

Please wait, it will take ~1-3 minutes to finish calcualting the two WEAP Areas with their many scenarios

2.1-A Bear River Model Scenarios (Headflow + Demand + Evaporation)

```python
# this library is needed to connect to the WEAP API
import win32com.client

# this command will open the WEAP software (if closed) and get the last active model
# you could change the active area to another one inside WEAP or by passing it to the command here
#WEAP.ActiveArea = "BearRiverFeb2017_V10.9"


WEAP=win32com.client.Dispatch("WEAP.WEAPApplication")

#-----------------------------------------------------------------
# I'm calling the active area many times because sometimes it didnt work to call it one time
print WEAP.ActiveArea.Name
WEAP.ActiveArea = "Bear_River_WEAP_Model_2017_scenarios"
print WEAP.ActiveArea.Name

WEAP.Areas("Bear_River_WEAP_Model_2017_scenarios").Open
WEAP.ActiveArea = "Bear_River_WEAP_Model_2017_scenarios"
print WEAP.ActiveArea.Name
#-----------------------------------------------------------------


print 'Please wait 1-3 min for the calculation to finish'
WEAP.Calculate(2006,10,True)
WEAP.SaveArea

print '\n \n The calculation has been done and saved'
print WEAP.CalculationTime

print 'Done'

SelectDemandSites=['Montpelier Irr','Highline Canal','Hyrum Canal','Bird Refuge','Logan Potable']



Result= OrderedDict()
# columns=['ScenarioName','BranchName','Reliability']
# Result=OrderedDict()

# group:ScenarioName
# Key=BranchName
# value:BranchName

for ScenarioName in WEAP.Scenarios:
    print ScenarioName
    WEAP.ActiveScenario=ScenarioName

    #print str(ScenarioName)
    scenarioKey = str(ScenarioName)
    Result[scenarioKey] = OrderedDict()

    BranchNames = []
    Reliabilities = []

    for Branch in WEAP.Branches:
        if Branch.TypeName=='Demand Site' and Branch.IsNode:
            FullBranchName=Branch.FullName
            BranchName=Branch.Name
            for SelectSite in SelectDemandSites:
                if SelectSite==BranchName:
                    if WEAP.Branch(FullBranchName).Variables.Exists("Reliability"):
                        BranchNames.append(BranchName)
                        Reliabilities.append(WEAP.Branch(FullBranchName).Variables("Reliability").Value)

    NewReliabilities = []
    for site in SelectDemandSites:
        index = BranchNames.index(site)
        NewReliabilities.append(Reliabilities[index])

#     print BranchNames
    Result[scenarioKey]['BranchName'] = SelectDemandSites
    Result[scenarioKey]['Reliability'] = NewReliabilities

print 'done'
print Result
```

2.1-B Bear River Model Scenarios (Sedeimtation)

```python
WEAP=win32com.client.Dispatch("WEAP.WEAPApplication")

#-----------------------------------------------------------------
print WEAP.ActiveArea.Name
WEAP.ActiveArea = "Bear_River_WEAP_Model_2017_sedimentation"
print WEAP.ActiveArea.Name

WEAP.Areas("Bear_River_WEAP_Model_2017_sedimentation").Open
WEAP.ActiveArea = "Bear_River_WEAP_Model_2017_sedimentation"
print WEAP.ActiveArea.Name
#-----------------------------------------------------------------


print 'Please wait 1-3 min for the calculation to finish'
WEAP.Calculate(2006,10,True)
WEAP.SaveArea

print '\n \n The calculation has been done and saved'
print WEAP.CalculationTime

print 'Done'

# WEAP.Visible = 'FALSE'

SelectDemandSites=['Montpelier Irr','Highline Canal','Hyrum Canal','Bird Refuge','Logan Potable']



Result2= OrderedDict()

for ScenarioName in WEAP.Scenarios:
    print ScenarioName

#     if ScenarioName=='Reference':
    WEAP.ActiveScenario=ScenarioName

    #print str(ScenarioName)
    scenarioKey = str(ScenarioName)
    Result2[scenarioKey] = OrderedDict()

    BranchNames = []
    Reliabilities = []

    for Branch in WEAP.Branches:
        if Branch.TypeName=='Demand Site' and Branch.IsNode:
            FullBranchName=Branch.FullName
            BranchName=Branch.Name
            for SelectSite in SelectDemandSites:
                if SelectSite==BranchName:
                    if WEAP.Branch(FullBranchName).Variables.Exists("Reliability"):
                        BranchNames.append(BranchName)
                        Reliabilities.append(WEAP.Branch(FullBranchName).Variables("Reliability").Value)
#     NewBranchNames = []
    NewReliabilities = []
    for site in SelectDemandSites:
        index = BranchNames.index(site)
        NewReliabilities.append(Reliabilities[index])
    Result2[scenarioKey]['BranchName'] = SelectDemandSites
    Result2[scenarioKey]['Reliability'] = NewReliabilities

print Result2

print 'done'
```

2.2-A: Run the Weber River Model (Headflow + Demand + Evaporation)

```python
import win32com.client

WEAP=win32com.client.Dispatch("WEAP.WEAPApplication")
#-----------------------------------------------------------------
print WEAP.ActiveArea.Name
WEAP.ActiveArea = "WeberOgdenRiversLab-3_scenarios"
print WEAP.ActiveArea.Name

WEAP.Areas("WeberOgdenRiversLab-3_scenarios").Open
WEAP.ActiveArea = "WeberOgdenRiversLab-3_scenarios"
print WEAP.ActiveArea.Name
#-----------------------------------------------------------------


print 'Please wait 1-3 min for the calculation to finish'
WEAP.Calculate(2006,10,True)
WEAP.SaveArea

print '\n \n The calculation has been done and saved'
print WEAP.CalculationTime

print 'Done'

SelectDemandSites=['Weber Basin Proj. Ogd Valley','Wanship to Echo']



Result3= OrderedDict()
# columns=['ScenarioName','BranchName','Reliability']
# Result=OrderedDict()

# group:ScenarioName
# Key=BranchName
# value:BranchName

for ScenarioName in WEAP.Scenarios:
    print ScenarioName
    WEAP.ActiveScenario=ScenarioName

    #print str(ScenarioName)
    scenarioKey = str(ScenarioName)
    Result3[scenarioKey] = OrderedDict()

    BranchNames = []
    Reliabilities = []

    for Branch in WEAP.Branches:
        if Branch.TypeName=='Demand Site' and Branch.IsNode:
            FullBranchName=Branch.FullName
            BranchName=Branch.Name
            for SelectSite in SelectDemandSites:
                if SelectSite==BranchName:
                    if WEAP.Branch(FullBranchName).Variables.Exists("Reliability"):
                        #Result['ScenarioName'] = ScenarioName
                        BranchNames.append(BranchName)
                        Reliabilities.append(WEAP.Branch(FullBranchName).Variables("Reliability").Value)

    NewReliabilities = []
    for site in SelectDemandSites:
        index = BranchNames.index(site)
        NewReliabilities.append(Reliabilities[index])

    Result3[scenarioKey]['BranchName'] = SelectDemandSites
    Result3[scenarioKey]['Reliability'] = NewReliabilities

print 'done'

print Result3
```

2.1-B Weber River Model Scenario (Sedeimtation)

```python
WEAP=win32com.client.Dispatch("WEAP.WEAPApplication")

#-----------------------------------------------------------------
print WEAP.ActiveArea.Name
WEAP.ActiveArea = "WeberOgdenRiversLab-3_sedimentation"
print WEAP.ActiveArea.Name

WEAP.Areas("WeberOgdenRiversLab-3_sedimentation").Open
WEAP.ActiveArea = "WeberOgdenRiversLab-3_sedimentation"
print WEAP.ActiveArea.Name
#-----------------------------------------------------------------


print 'Please wait 1-3 min for the calculation to finish'
WEAP.Calculate(2006,10,True)
WEAP.SaveArea

print '\n \n The calculation has been done and saved'
print WEAP.CalculationTime

print 'Done'

# WEAP.Visible = 'FALSE'

SelectDemandSites=['Weber Basin Proj. Ogd Valley','Wanship to Echo']



Result4= OrderedDict()

for ScenarioName in WEAP.Scenarios:
    print ScenarioName

#     if ScenarioName=='Reference':
    WEAP.ActiveScenario=ScenarioName

    #print str(ScenarioName)
    scenarioKey = str(ScenarioName)
    Result4[scenarioKey] = OrderedDict()

    BranchNames = []
    Reliabilities = []

    for Branch in WEAP.Branches:
        if Branch.TypeName=='Demand Site' and Branch.IsNode:
            FullBranchName=Branch.FullName
            BranchName=Branch.Name
            for SelectSite in SelectDemandSites:
                if SelectSite==BranchName:
                    if WEAP.Branch(FullBranchName).Variables.Exists("Reliability"):
                        #Result['ScenarioName'] = ScenarioName
                        BranchNames.append(BranchName)
                        Reliabilities.append(WEAP.Branch(FullBranchName).Variables("Reliability").Value)


    NewReliabilities = []
    for site in SelectDemandSites:
        index = BranchNames.index(site)
        NewReliabilities.append(Reliabilities[index])


    Result4[scenarioKey]['BranchName'] = SelectDemandSites
    Result4[scenarioKey]['Reliability'] = NewReliabilities




print Result['Reference']['BranchName'],
print Result['Reference']['Reliability'],


print 'done'
print Result4
```

Plot: System reliability to meet demand targets across scenarios in the Bear (blue) and Weber (red) Rivers WEAP models.

```python
scenario1 = go.Bar(
    x=Result['Demand']['BranchName'],
    y=Result['Demand']['Reliability'],
    name = 'Reduce demand by 10%',

    marker = dict(
        color = '#290AD8'
))


scenario2 = go.Bar(
    x=Result['Reference']['BranchName'],
    y=Result['Reference']['Reliability'],
    name = 'Base Case',
    marker = dict(
        color = '#000000',

))

scenario3 = go.Bar(
    x=Result['Evaporation']['BranchName'],
    y=Result['Evaporation']['Reliability'],
    name = 'Increase evaporation by 10%',

    marker = dict(
        color = '#3FA0FF'
))



scenario4 = go.Bar(
    x=Result2['Reference']['BranchName'],
    y=Result2['Reference']['Reliability'],
    name = 'Reduce reservoir capacities by 10%',

    marker = dict(
        color = '#72D9FF'
))



scenario5 = go.Bar(
    x=Result['Headflow']['BranchName'],
    y=Result['Headflow']['Reliability'],
    name = 'Reduce headflows by 10%',

    marker = dict(
        color = '#AAF7FF'
))


#$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$

scenario11 = go.Bar(
    x=Result3['Demand']['BranchName'],
    y=Result3['Demand']['Reliability'],
    name = 'Reduce demand by 10%',
#     showlegend=False,

    marker = dict(
        color = '#A50021'
))


scenario12 = go.Bar(
    x=Result3['Reference']['BranchName'],
    y=Result3['Reference']['Reliability'],
    name = 'Base Case',
#     showlegend=False,

    marker = dict(
        color = '#000000',

))

scenario13 = go.Bar(
    x=Result3['Evaporation']['BranchName'],
    y=Result3['Evaporation']['Reliability'],
    name = 'Increase evaporation by 10%',
#     showlegend=False,

    marker = dict(
        color = '#D82632'
))



scenario14 = go.Bar(
    x=Result4['Reference']['BranchName'],
    y=Result4['Reference']['Reliability'],
    name = 'Reduce reservoir capacities by 10%',
#     showlegend=False,

    marker = dict(
        color = '#F76D5E'
))



scenario15 = go.Bar(
    x=Result3['Headflow']['BranchName'],
    y=Result3['Headflow']['Reliability'],
    name = 'Reduce headflows by 10%',
#     showlegend=False,
    marker = dict(
        color = '#FFAD72'
))




layout = dict(
    #title = "Use Case 3.3",
    yaxis = dict(
        title = "Demand reliability (%)",
        tickformat= ',',
        showline=True,
        tick0=40,

        dtick='10',
        ticks='outside',
        range = [40, 105],
        ticklen=10,
        tickcolor='#000',
        gridwidth=1,
        showgrid=True,

                ),

    xaxis = dict(
#         title = "Updated input parameters in the <br>Bear_River_WEAP_Model_2017",
#         showline=True,
        ticks='inside',
        tickfont=dict(size=22),
            tickcolor='#000',
        gridwidth=1,
        showgrid=True,

        ticklen=10
                    ),
        legend=dict(
        x=1,y=.56,
          bordercolor='#00000f',
            borderwidth=2

               ),
    width=1100,
    height=700,
    #paper_bgcolor='rgb(233,233,233)',
    #plot_bgcolor='rgb(233,233,233)',
    margin=go.Margin(l=130,b=200),
    font=dict(size=25,family='arial',color='#00000f'),

    showlegend=True,
    shapes=[{'type': 'line',
                                          'x0': 0, 'x1': 1, 'xref': 'paper',
                                          'y0': 40, 'y1': 40, 'yref': 'y'}]
                )


data = [scenario1, scenario2,scenario3,scenario4,scenario5,scenario11, scenario12,scenario13,scenario14,scenar


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


## 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 = 'jupyter/UnmentDemand@BirdRefuge' )

print "Figure 5 is replicated!!"
```

7. Close WEAP API connection

```python
# 9. Close the WEAP API connection


print 'connection disconnected'

# Uncomment
WEAP.SaveArea



# this command will close WEAP
WEAP.Quit

print 'Connection with WEAP API is disconnected'
```

The End :) Congratulations!