Observation Analysis Tool (OAT)

Massimiliano Cannata, Mirko Cardoso, Jakob Neumann

FREEWAT
Free and Open Source Software Tools for Water Resource Management
EU HORIZON 2020 Project

SUPSI

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 642224

FREEWAT

This is a working version of the Observation Analysis Tool tutorial! Your feedback is much appreciated.

This tutorial corresponds to FREEWAT version 0.1 and may not be compatible with earlier plugin versions.

FREEWAT - Free and Open Source Software Tools for Water Resource Management

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 642224

OAT in a nutshell

Contents

OAT Theory and overview

Using OAT

- Adding sensors
- Managing Sensors
- Processing Sensors
- Comparing Sensors

OAT in a nutshell

Overview

Time series are a key aspect in environmental modelling, and more and more are getting important with the increasing establishment of diffuse, online and real-time monitoring networks.

Using OAT you can upload, explore, analyse and get the maximum value out of your observations.

In particular, they are important as a means of:

- understanding the system to be modelled and thus support the **preparation of model input data**
- verification of models results and thus help to **calibrate your model**.

OAT - Overview

Preparation of model input data

Any groundwater balance or numerical model requires quantitative data which:

1. Define the physical properties of the groundwater basin/study area
2. Describe its hydrological framework
    - Rate of recharge/discharge

Identifying data needs, as well as monitoring/collecting data is an essential part of any groundwater/water resources related task.

Input data for model conceptualization and implementation

Output data for validation

Observation data for sensitivity analysis, optimization and prediction

OAT - Overview

Groundwater models

General time series data needs:

- Climate:
  - Rainfall
  - Evaporation
  - Evapotranspiration
  - Other recharge
- Boundary condition changes:
  - Water body stages
  - Discharge
- Water management
  - Irrigation
  - Pumping
- Other:
  - Contamination & Transport

OAT in a nutshell

On-going process

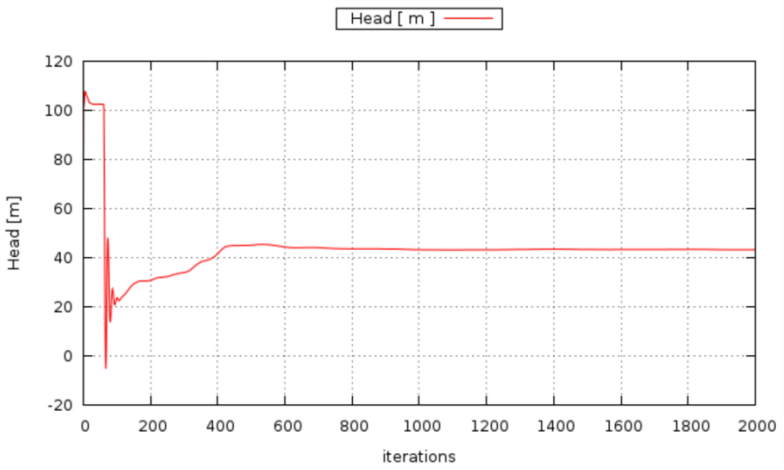
While the developement of the core of the library is completed, in the near future several upgrades and new features will be available, so be tuned for library upgrades!

Overall, this is an Open Source project that for its nature is continuosly evolving and growing thanks to the participatory contribution of the community in terms of feedbacks and coding !

Understanding OAT

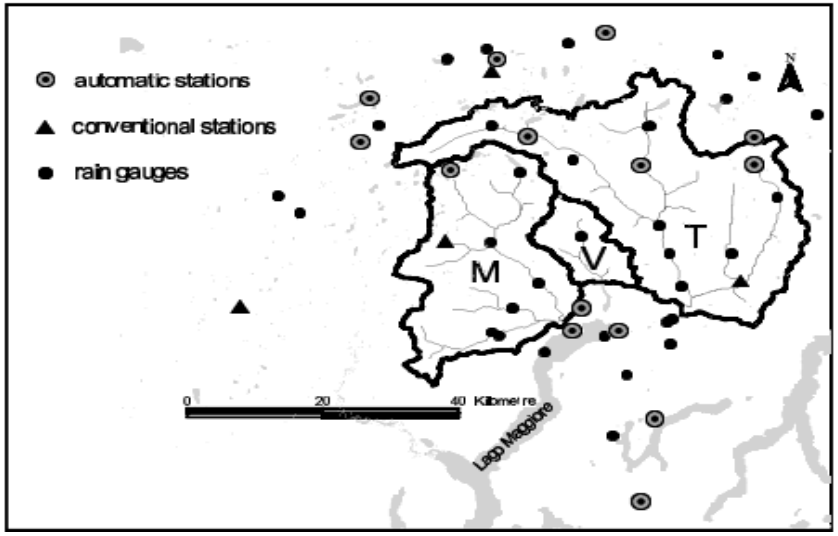
The basic concept

You have a **sensor** making some **observations**



…and then you have **some more information**…

Understanding OAT



2015-06-12 09:40:00, 100, 0.237, OBS1, True
2015-06-12 09:50:00, 100, 0.234, OBS2, True
2015-06-12 10:00:00, 100, 0.237, OBS3, True
2015-06-12 10:10:00, 100, 0.236, OBS4, True
2015-06-12 10:20:00, 100, 0.234, OBS5, True
2015-06-12 10:30:00, 100, 0.237, OBS6, False
2015-06-12 10:40:00, 200, 0.936, OBS7, True
2015-06-12 10:50:00, 200, 0.932, OBS8, True

Sensor location and metadata:

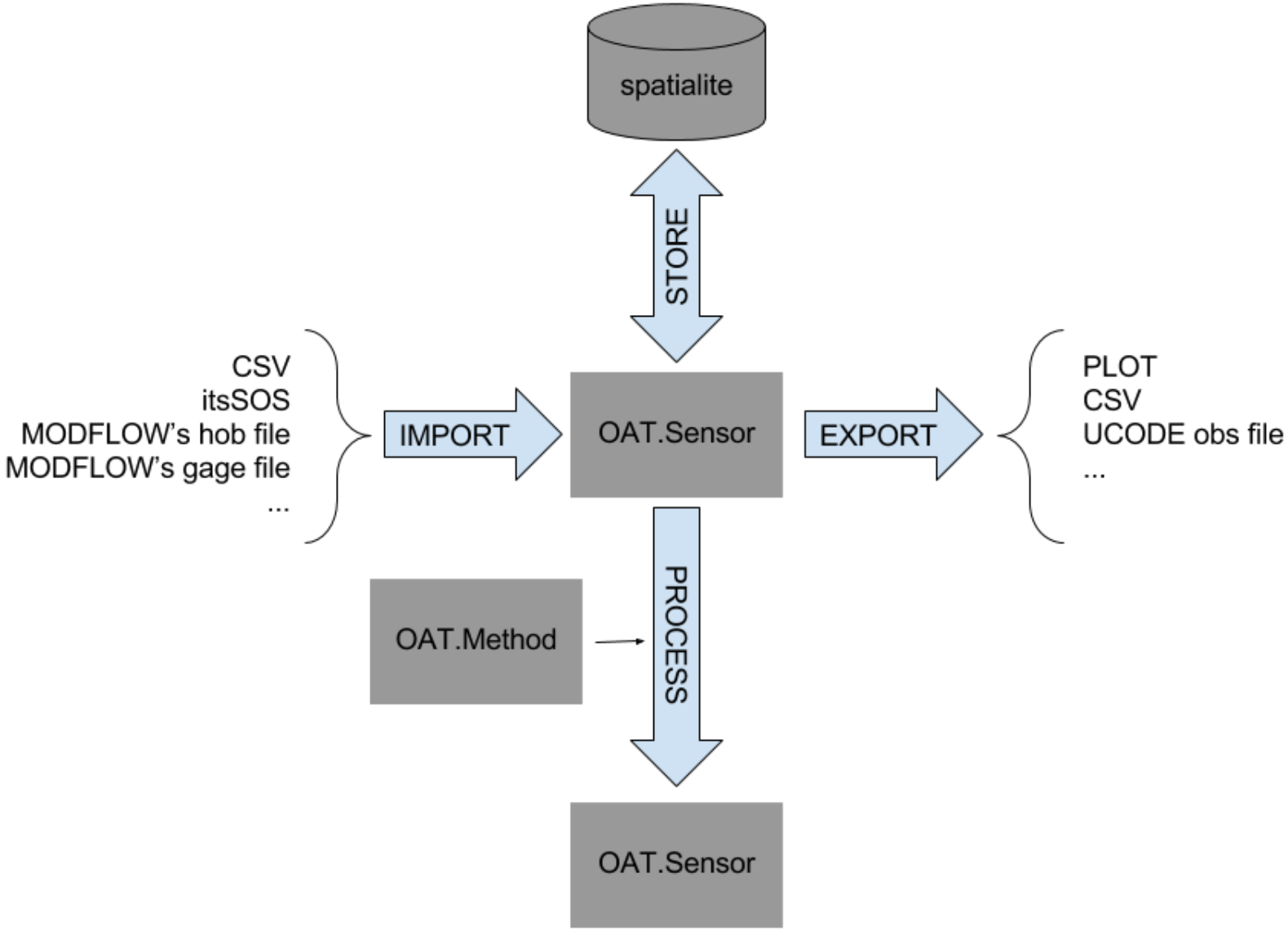
- Name
- Description
- Location (lat,lon,elev)
- Unit of measure
- Observed porperty
- Coordinate system
- Timezone
- Frequency (if regualr time series)
- Weight statistic
- Data availability (time interval)

Time series of observations

- Time
- Quality*
- Values
- Obs. Index
- Use

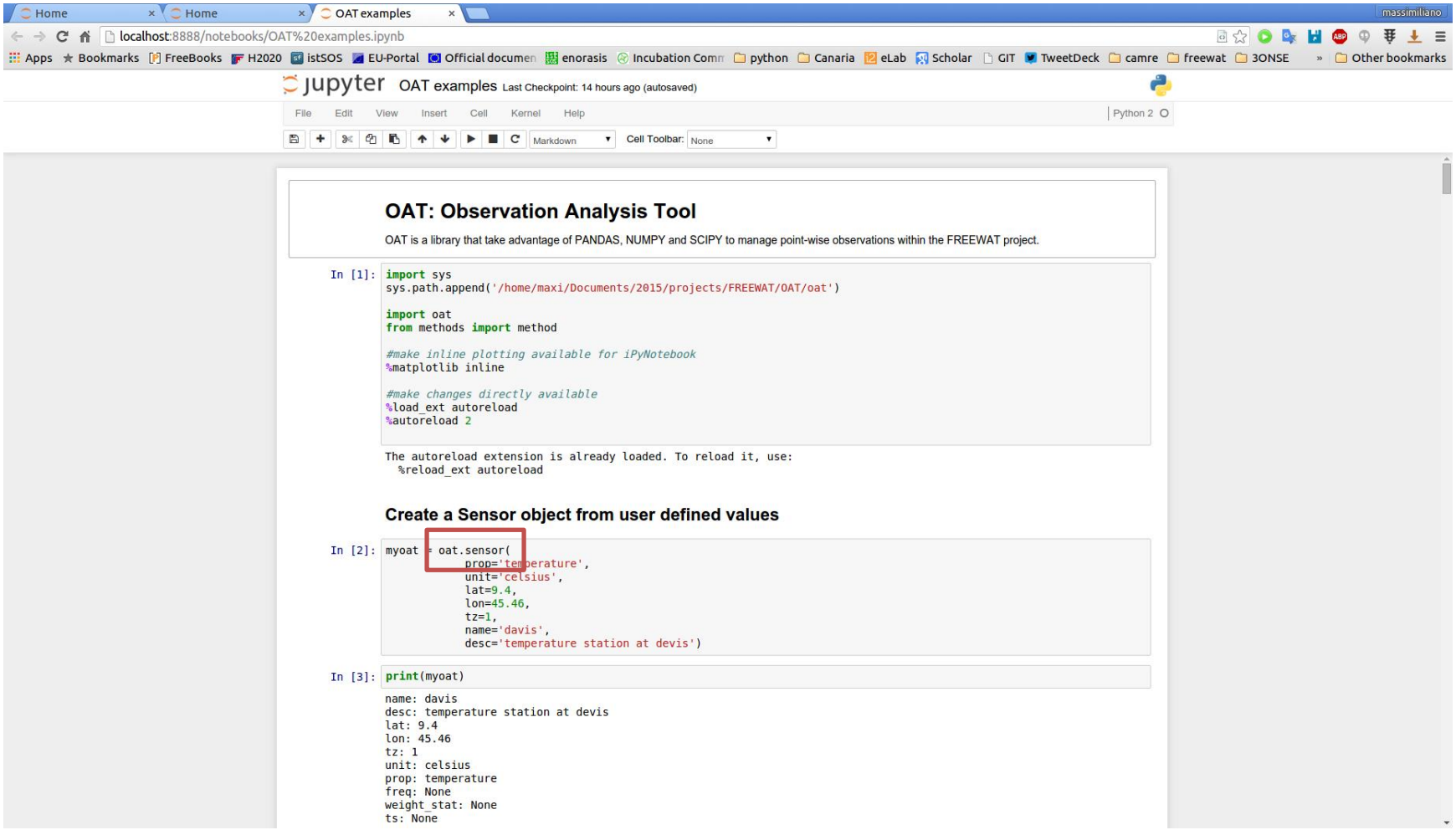
*Quality could be an identifying index (e.g. 100 for raw data, 200 for statistically sound data) or a statistical index used to calculate the weight of each observation. See FREEWAT manual Vol. 5 & 6 for further information.

OAT libraray concept



OAT: Python library

This preview shows the OAT code. Important is the separation into **.sensor** objects…



Home | Home | OAT examples — localhost:8888/notebooks/OAT%20examples.ipynb

jupyter OAT examples Last Checkpoint: 14 hours ago (autosaved)

File Edit View Insert Cell Kernel Help — Python 2

OAT: Observation Analysis Tool

OAT is a library that take advantage of PANDAS, NUMPY and SCIPY to manage point-wise observations within the FREEWAT project.

In [1]:
```
import sys
sys.path.append('/home/maxi/Documents/2015/projects/FREEWAT/OAT/oat')

import oat
from methods import method

#make inline plotting available for iPyNotebook
%matplotlib inline

#make changes directly available
%load_ext autoreload
%autoreload 2
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

Create a Sensor object from user defined values

In [2]:
```
myoat = oat.sensor(
            prop='temperature',
            unit='celsius',
            lat=9.4,
            lon=45.46,
            tz=1,
            name='davis',
            desc='temperature station at devis')
```

In [3]:
```
print(myoat)
```

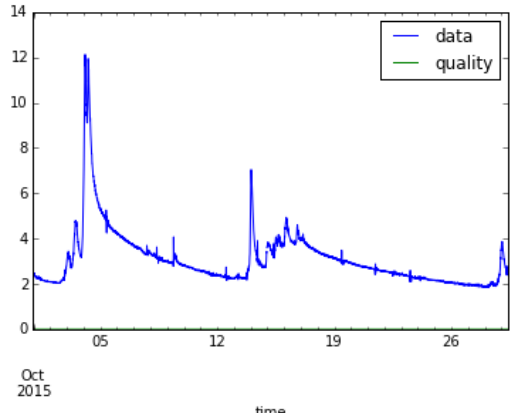
```
name: davis
desc: temperature station at devis
lat: 9.4
lon: 45.46
tz: 1
unit: celsius
prop: temperature
freq: None
weight_stat: None
ts: None
```

OAT features

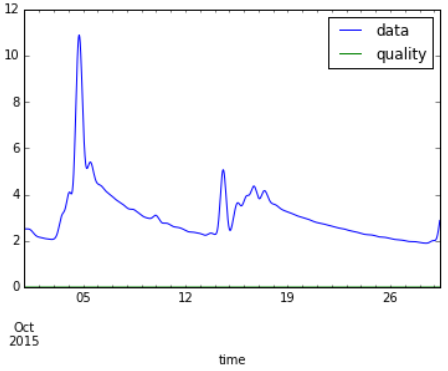
…and **.method** objects

Here: Digital filter, exceedance probability, baseflow separation

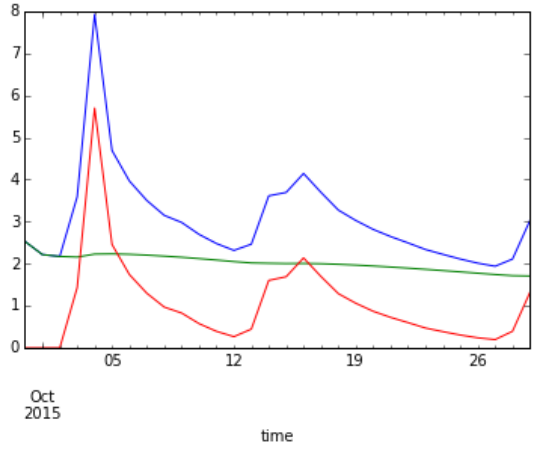
```
: #orgiginal data
CUC2.plot(quality=True)
```




```
#Lowpass filter
CUC4.process(
    method.DigitalFilter(
                1, 0.01, order=3, btype='lowpass')
).plot(quality=True)
```




```
#Hydrograph separation
#with two parameter digital filter
#=================================
CUCd = CUC2.process(
    method.Resample(freq='1D',how='mean',
                    fill='ffill', how_quality='sum'))
base, runoff = CUCd.process(method.Hysep(mode='TPDF'))

CUCd.plot()
base.plot()
runoff.plot()
```



Calculate exceedance time

```
In [25]: #Exceedance values from exceedance percentage
A = CUC2.process(method.Exceedance(perc=[5, 50, 95]))
print "----------------------------"
print "Percentage\t: %s\t%%" % A[0][0]
print "Value\t\t: %s\tm3/s"  % A[0][1]
print "----------------------------"
print A
```

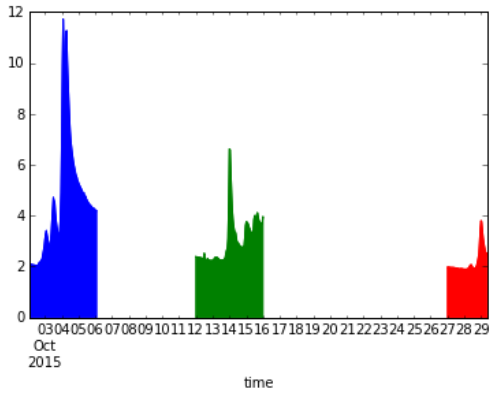
```
----------------------------
Percentage      : 5.0   %
Value           : 1.967 m3/s
----------------------------
[[  5.      1.967]
 [ 50.      2.761]
 [ 95.      4.968]]
```

OAT features

Extract events, calculate volumes, hydro-indices, goodness of fit, ...

```
# Extract peak events
events = CUCh.process(
    method.Hydro_events(rise_lag=2,fall_lag=2,window=4,min_peak=3.8))

# Extract periods
periods = [ e.period() for e in events ]

# Calculate volumes
volumes = CUCh.process(method.Integrate(periods=periods, astext=True))

#print the calculate vlumes
print "Cubic Meters:", [i[2] for i in volumes]

#Print the areas of the events
for e in events:
    e.plot(kind='area')

#Calculate with/without factor and tunit
#print CUCh.process(method.Integrate(periods=periods, factor=0.001, astext=True))
#print CUCh.process(method.Integrate(periods=periods, tunit="hours", astext=True))
```

```
Cubic Meters: [1606704.6000000001, 1046764.8600000001, 449653.20000000001]
```




```
#Hydrological indexes
for c in range(1,46):
    val = myOat.process(
        method.Hydrologic_indices(
            htype='MA', code=c, drain_area=100))
    print "MA%s: %s" %(c,val)
```

```
MA1: -0.120394010305
MA2: -0.046214
MA3: 9.25136773865
MA4: -1.20345986548
MA5: 2.60514152216
MA6: -1.37611514673
MA7: -1.41286013942
MA8: -1.39819099706
```

```
#Goodness of fit
# CUCh is the Observation
# rand_CUCh is the Simulation
CUCh.process(method.Compare(rand_CUCh,
        stats=['BIAS', 'SANDARD_ERROR', 'RELATIVE_BIAS', 'RELATIVE_STANDARD_ERROR',
               'NASH_SUTCLIFFE', 'COEFFICIENT_OF_EFFICIENCY',
               'INDEX_OF_AGREEMENT', 'VOLUMETRIC_EFFICIENCY'], exponent=1))
```

```
{u'BIAS': -3.1010381729736666,
 u'COEFFICIENT_OF_EFFICIENCY': -2.7652755213651776,
 u'INDEX_OF_AGREEMENT': 0.20950312134658267,
 u'NASH_SUTCLIFFE': -6.5619543988222517,
 u'RELATIVE_BIAS': -1.0073674194139706,
 u'RELATIVE_STANDARD_ERROR': 1.256717402443948,
 u'SANDARD_ERROR': 1.5933059782834331,
 u'VOLUMETRIC_EFFICIENCY': -0.0079295228874634027}
```