University of Zurich[UZH]

Department of Geography

COST is supported by the EU Framework Programme Horizon 2020

cost EUROPEAN COOPERATION IN SCIENCE AND TECHNOLOGY

UNIVERSITY OF WOLLONGONG

RSL measurements | products | policy

Python accessing SPECCHIO

Andy Hueni[1]

[1] Remote Sensing Laboratories, University of Zurich, Switzerland

OPTIMISE SPECCHIO Programming Course

SPECCHIO

Agenda

- Configuring your machine for SPECCHIO access from Python
- Connecting to SPECCHIO
- Selecting data
- Load spectral data from database
- Load metadata from database

Java – Python Bridge

Two bridging packages have been tested, but only JPype is currently supported:

- Py4J: https://www.py4j.org
- JPype: https://jpype.readthedocs.io/en/latest/#

Install JPype packages on your operating system.

Based on tests, **JPype is currently the preferred option**, as transfers of Java arrays (i.e. spectral matrices) to Python arrays are much faster.

* Py4J is partly supported from client build 160 onwards

Connecting to SPECCHIO – Python – JPype

- Create a new Python script, test each of the lines below as you enter them.
- Import jpype, start JVM with clas path pointing to specchio client jar file, get specchio client package:

```
from jpype import *
startJVM('/Library/Java/JavaVirtualMachines/jdk1.7.0_79.jdk/Contents/MacOS/libjli.dylib', "-ea",
"-Djava.class.path=/Applications/SPECCHIO/SPECCHIO.app/Contents/Java/specchio-client.jar")

SPECCHIO = JPackage('ch').specchio.client
```

- Create a client factory instance and get a list of all connection details:

```
cf = SPECCHIO.SPECCHIOClientFactory.getInstance();
db_descriptor_list = cf.getAllServerDescriptors();
```

- Create client for the first entry in the database connection list:

```
specchio_client = cf.createClient(db_descriptor_list.get(0)) # zero indexed
```

- If no error appears then you are connected successfully to the SPECCHIO database running in your SPECCHIO VM

Connecting to SPECCHIO – Python – JPype

- If you want to see what server you just connected to, type:

```
db_descriptor = db_descriptor_list.get(0)
db_descriptor.getDataSourceName()
```

Selecting data – Python – JPype

The SPECCHIO tutorial dataset contains some spectra that have a spatial position.
We are now going to select spectra based on their altitude above sea level. Our condition is that the altitude must be 50m or higher.

•Create a query object:

```
QUERIES = JPackage('ch').specchio.queries
query = QUERIES.Query();
```

•Create query condition for the altitude attribute and configure it:

```
attr = specchio_client.getAttributesNameHash().get('Altitude')
cond =QUERIES.EAVQueryConditionObject(attr)
cond.setValue('50.0');
cond.setOperator('>=');
query.add_condition(cond);
```

•Get spectrum ids that match the query:

```
ids = specchio_client.getSpectrumIdsMatchingQuery(query);
```

Selecting data – Python – JPype

- Check how many spectra we found

```
ids.size()
```

- Add another condition to restrict the maximum altitude:

```
cond = QUERIES.EAVQueryConditionObject(attr);
cond.setValue('55.0');
cond.setOperator('<');
query.add_condition(cond);
```

- Get spectrum ids that match the query

```
ids = specchio_client.getSpectrumIdsMatchingQuery(query);
```

- Check the size of the ids variable again – there should be less spectra selected than before

Load spectral data from database – Python – JPype

Refer to the general presentation on SPECCHIO explaining the Spectral Space concept and the Space Factory; the following statements are using the implementation of these concepts.

•Create spectral spaces and order the spectra by their acquisition time (note: this does not load the spectral vectors yet but only prepares the 'containers') :

```
spaces = specchio_client.getSpaces(ids, 'Acquisition Time');
```

•Find out how many space we have received:

```
len(spaces)
```

•Get the first space:

```
space = spaces[0]
```

•The order of the ids may have changed because the vectors in the space are ordered by their Acquisition Time; therefore get the ids in their correct sequence to match the vector sequence:

```
ids = space.getSpectrumIds(); # get them sorted by 'Acquisition Time'
```

•Load the spectral data from the database into the space:

```
space = specchio_client.loadSpace(space);
```

Load spectral data from database – Python – JPype

- Get the spectral vectors, wavelengths and measurement unit, the plot using matplotlib.

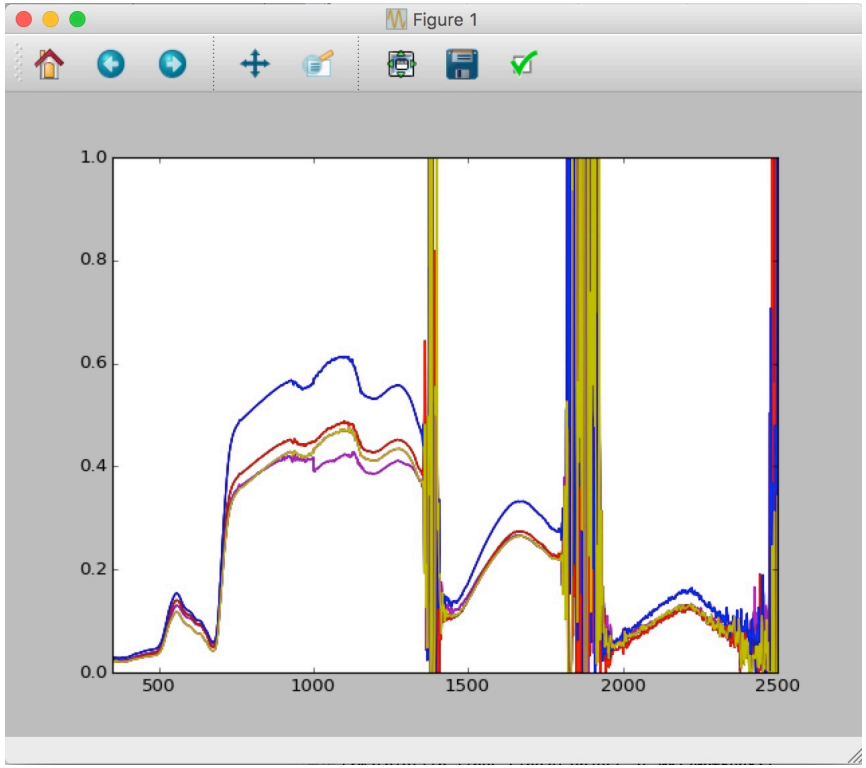
```
vectors = space.getVectorsAsArray();
wvl = space.getAverageWavelengths();
unit = space.getMeasurementUnit().getUnitName();


import numpy


import matplotlib.pyplot as mp
axes = mp.gca()
axes.set_xlim([350,2500])
axes.set_ylim([0,1])
y = numpy.rot90(vectors,3)
mp.plot(wvl, y)



   # end connection to Java
shutdownJVM()
```



Thank you for your attention!

For more information on the current version of SPECCHIO see: www.specchio.ch

https://twitter.com/SPECCHIO_DB

Australian Government
Department of Industry Innovation, Science, Research and Tertiary Education

UNIVERSITY OF WOLLONGONG

SPECCHIO

ands AUSTRALIAN NATIONAL DATA SERVICE

cost ES0903 EUROPEAN COOPERATION IN SCIENCE AND TECHNOLOGY

OPTIMISE Innovative Optical Tools for Proximal Sensing of Ecophysiological Processes

Appendix: Py4J Documentation

Note: support for Py4J is currently disabled!

Connecting to SPECCHIO – Python – Py4J

- Create a new Python script, test each of the lines below as you enter them.
- Start the SPECCHIO java client. This starts the Java Py4J Gateway. (If you start the SPECCHIO client from the terminal, you will see a message similar to:

```
Welcome to 3.2.1.3 Alpha
Specchio - Python Gateway Server Started
```

- Import the PyJ4 Gateway and get the gateway object:

```
from py4j.java_gateway import JavaGateway
gateway = JavaGateway()
```

- Create a client factory instance and get a list of all connection details:

```
cf=gateway.entry_point.getClientFactory()
db_descriptor_list = cf.getAllServerDescriptors()
```

- Create client for the first entry in the database connection list:

```
specchio_client = cf.createClient(db_descriptor_list.get(0)) # zero indexed
```

- If no error appears then you are connected successfully to the SPECCHIO database running in your SPECCHIO VM

Connecting to SPECCHIO – Python – Py4J

- If you want to see what server you just connected to, type:

```
db_descriptor = db_descriptor_list.get(0)
db_descriptor.getDataSourceName()
```

Selecting data – Python – Py4J

The SPECCHIO tutorial dataset contains some spectra that have a spatial position.
We are now going to select spectra based on their altitude above sea level. Our condition is that the altitude must be 50m or higher.

•Create a query object:

```
query = specchio_client.getQueryObject()
```

•Create query condition for the altitude attribute and configure it:

```
attr = specchio_client.getAttributesNameHash().get('Altitude')
cond =specchio_client.getEAVQueryConditionObject(attr)
cond.setValue('50.0');
cond.setOperator('>=');
query.add_condition(cond);
```

•Get spectrum ids that match the query:

```
ids = specchio_client.getSpectrumIdsMatchingQuery(query);
```

Selecting data – Python – Py4J

- Check how many spectra we found

```
ids.size()
```

- Add another condition to restrict the maximum altitude:

```
cond = specchio_client.getEAVQueryConditionObject(attr);
cond.setValue('55.0');
cond.setOperator('<');
query.add_condition(cond);
```

- Get spectrum ids that match the query

```
ids = specchio_client.getSpectrumIdsMatchingQuery(query);
```

- Check the size of the ids variable again – there should be less spectra selected than before

Load spectral data from database – Python – Py4J

Refer to the general presentation on SPECCHIO explaining the Spectral Space concept and the Space Factory; the following statements are using the implementation of these concepts.

•Create spectral spaces and order the spectra by their acquisition time (note: this does not load the spectral vectors yet but only prepares the 'containers') :

```
spaces = specchio_client.getSpaces(ids, 'Acquisition Time');
```

•Find out how many space we have received:

```
len(spaces)
```

•Get the first space:

```
space = spaces[0]
```

•The order of the ids may have changed because the vectors in the space are ordered by their Acquisition Time; therefore get the ids in their correct sequence to match the vector sequence:

```
ids = space.getSpectrumIds(); # get them sorted by 'Acquisition Time'
```

•Load the spectral data from the database into the space:

```
space = specchio_client.loadSpace(space);
```

Load spectral data from database – Python – Py4J

- Get the spectral vectors, wavelengths and measurement unit, the plot using matplotlib. Note: conversion of the vectors to the list data format is time consuming.

```
vectors = space.getVectorsAsArray();
wvl = space.getAverageWavelengths();
unit = space.getMeasurementUnit().getUnitName();

wavelengths = list(wvl)

import matplotlib.pyplot as mp
mp.hold
axes = mp.gca()
axes.set_xlim([350,2500])
axes.set_ylim([0,1])
for i, dataset in enumerate(vectors):
mp.plot(wavelengths, list(dataset))

    # further option using numpy, but also not faster …
import numpy
from numpy import array
x = array(wavelengths)
v=array(list(vectors))
mp.plot(x, numpy.rot90(v))
a = numpy.asarray(list(vectors))
```