University of Zurich[UZH]

Department of Geography

UNIVERSITY OF WOLLONGONG

RSL measurements | products | policy

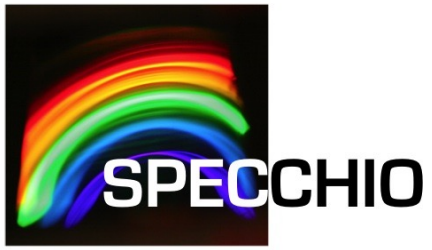
Matlab accessing SPECCHIO

Andy Hueni[1]

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

SPECCHIO Programming Course

Status: 27.3.2023



OPTI ISE

COST is supported by the EU Framework Programme Horizon 2020

cost EUROPEAN COOPERATION IN SCIENCE AND TECHNOLOGY

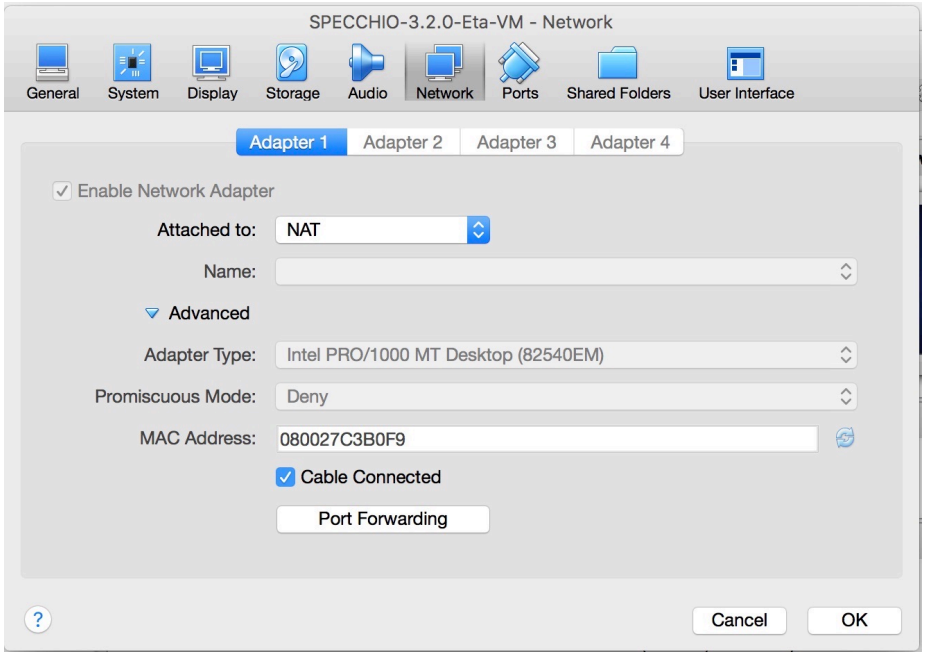
EARSeL

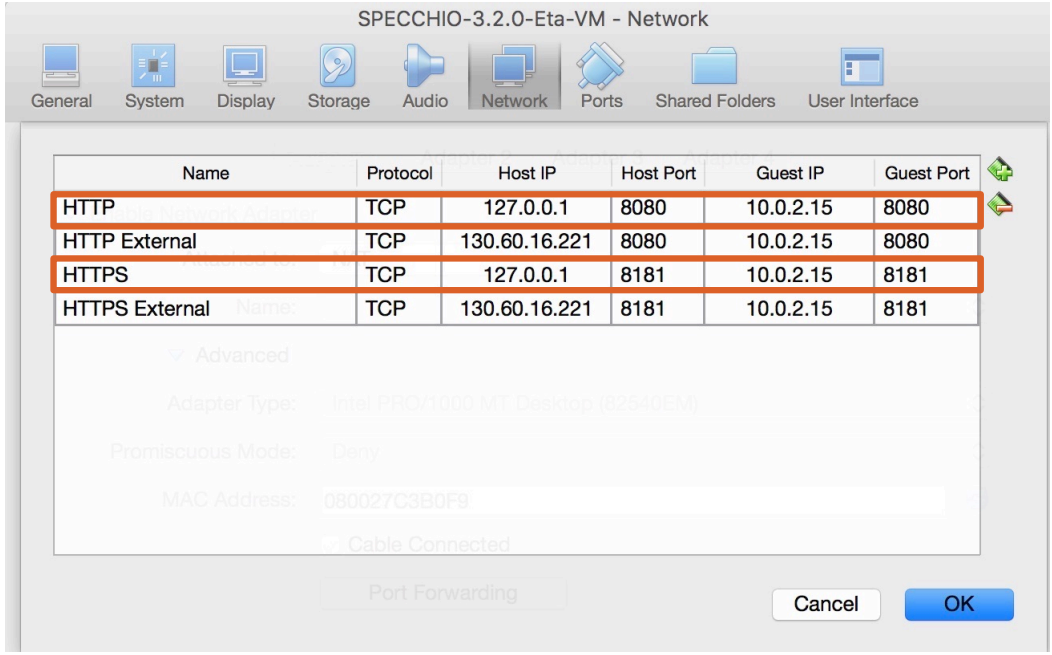
Agenda

- Configuring your machine for SPECCHIO access from Matlab (for a local VM)
- Connecting to SPECCHIO
- Selecting data
- Load spectral data from database
- Load metadata from database
- Adding metadata
- Inserting spectra and metadata from a file
  - Creating a new campaign
  - Creating a hierarchy
  - Creating new attributes in the database
  - Querying campaign and hierarchy IDs
  - Inserting spectral data and metadata using a spectral file object
- Spatial Operations (SPECCHIO V3.3.0.0)
- Matlab GUI demonstration
- Resources: example code
- Data Selection: Auto-generated Code

Making the SPECCHIO VM server accessible from the host machine

- **Generally, your VM should be preconfigured to support accessibility out of the box.**
- Open the ‘settings’ of your VM and select the ‘Network’ tab.
- Add two entries in the ‘Port Forwarding’ table to map HTTPS and HTTP ports of the host to the VM

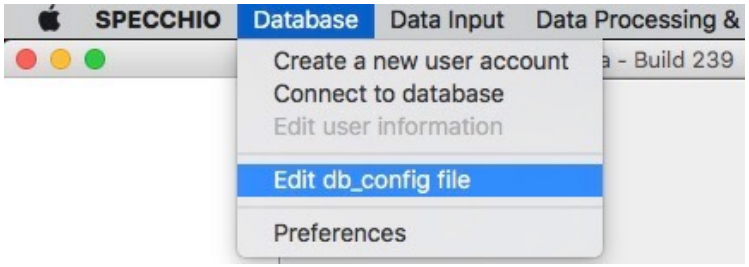


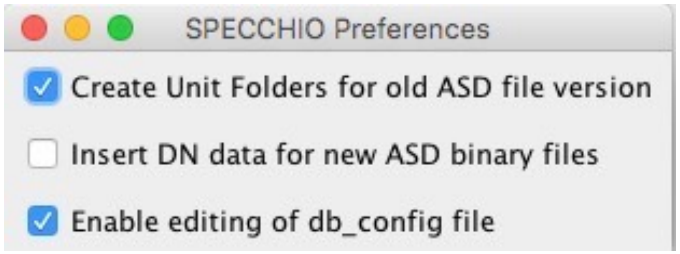


Install the SPECCHIO Client on your Host Machine

To access SPECCHIO from the host machine the SPECCHIO client must be installed on the host machine.
Download the SPECCHIO client from the SPECCHIO web page
(https://specchio.ch/downloads/) or from GitHub.

Add an entry into the SPECCHIO client configuration file to access the SPECCHIO VM as admin user. In case the menu item is greyed out, enable it first in the SPECCHIO Preferences.





Then copy and modify an existing connection to use HTTP and port 8080 (Otherwise Matlab cannot connect):
SPECCHIO VM:

```
http, localhost, 8080, /specchio_service, sdb_admin, 5p3cch10, jdbc/specchio
```

For a SPECCHIO instance running on a server, e.g. the test server:

```
http, sc22.geo.uzh.ch, 8080, /specchio_service, JDoe, the_password, jdbc/specchio_test
```

Now try connecting to the SPECCHIO database from your host machine using the new account details you just added.

Connecting to SPECCHIO - Matlab

- Create a new Matlab script, test each of the lines below as you enter them.
- Add the path to the SPECCHIO jar files installed on your host machine (example given for MacOS): this not only defines where the SPECCHO classes are but also gives automatically access to the connection details stored in the db_config file.

```
% dynamic classpath setting
    javaaddpath ({'/Applications/SPECCHIO/SPECCHIO.app/Contents/Java/specchio-client.jar', ...
        '/Applications/SPECCHIO/SPECCHIO.app/Contents/Java/specchio-types.jar'});
```

- Import the classes of the SPECCHIO client package:

```
import ch.specchio.client.*;
```

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

```
cf = 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 - Matlab

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

```
db_descriptor_list.get(0)
```

Selecting data - Matlab

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.

- Import the query package:

```
import ch.specchio.queries.*;
```

- Create a query object:

```
query = Query();
```

- Get the attribute that we want to restrict:

```
attr = specchio_client.getAttributesNameHash().get('Altitude');
```

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

```
cond = 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 - Matlab

- Check how many spectra we found

```
ids.size()
```

- List the spectrum ids:

```
ids.toArray
```

- Add another condition to restrict the maximum altitude:

```
cond = 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 - Matlab

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:

```
spaces.length()
```

•Get the first space (note: the spaces are now a Matlab array and are therefore indexed starting at 1):

```
space = spaces(1);
```

•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 - Matlab

- Get the spectral vectors as Matlab array:

```
vectors = space.getVectorsAsArray();
```

- Get the wavelengths are a vector:

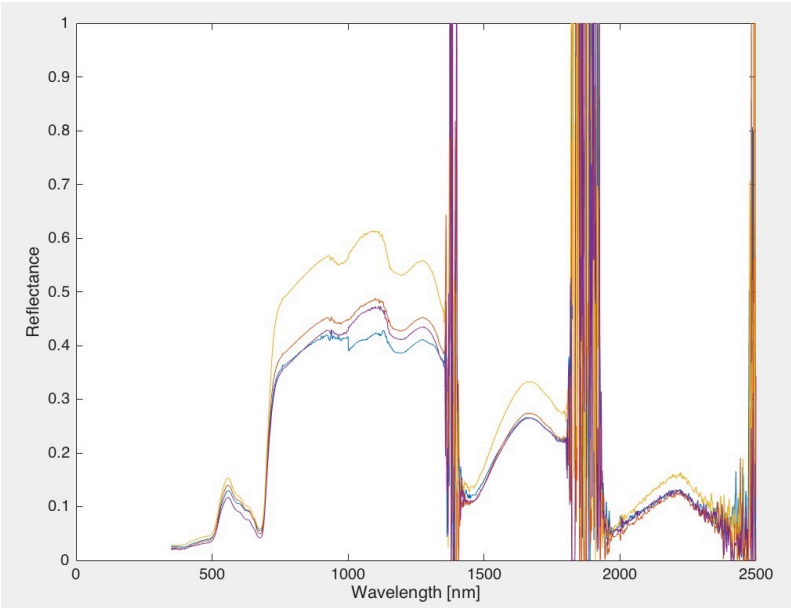
```
wvl = space.getAverageWavelengths();
```

- Get the unit of the spectra:

```
unit = char(space.getMeasurementUnit.getUnitName);
```

- Plot the spectra:

```
figure
plot(wvl, vectors)
xlabel('Wavelength [nm]')
ylabel(unit)
ylim([0 1])
```



Load metadata from database - Matlab

- Get the file names of the spectra held by the current space:

```
filenames_ = specchio_client.getMetaparameterValues(ids, 'File Name');
```

- Create legend with filenames and add it to plot:

```
for i=1:size(filenames_)
    legend_str{i} = filenames_.get(i-1);
end

legend (legend_str)
```

- Get lat/lon as vectors and plot them:

```
positions =  specchio_client.getMetaparameters(ids, 'Spatial Position', java.lang.Boolean(false));
lat = zeros(positions.size(),1);
lon = zeros(positions.size(),1);

for i=0:positions.size() - 1
    lat(i+1) = positions.get(i).getPoint2D().getY();
    lon(i+1) = positions.get(i).getPoint2D().getX();
end
figure
plot(lon, lat, '*');
```

Adding metadata - Matlab

New metadata can be added easily to existing spectral records; assignment of the same value can be done to multiple spectra at once. Let us assume that we want to add a proper species name to all blackfern spectra. All these spectra have been given a filename of bfern.XXX.
In a first step we will select all spectra with that filename to get all spectrum ids, then we will add the Common Name of this plant.

•Get all ids that have a filename starting with bfern (note that the % character is the matching wildcard for one or more characters; as it is a special character in Matlab, one must use two % in a row):

```
query = Query();

attr = specchio_client.getAttributesNameHash().get('File Name');

cond = EAVQueryConditionObject(attr);
cond.setValue('bfern.%%');
cond.setOperator('like');
query.add_condition(cond);
```

•Get spectrum ids that match the query:

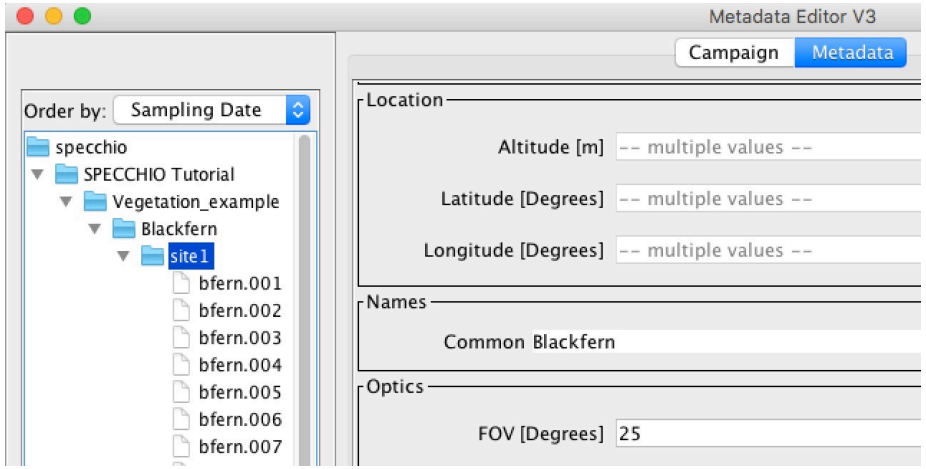
```
ids = specchio_client.getSpectrumIdsMatchingQuery(query);
```

Adding metadata - Matlab

- Create new metaparameter and attach it to the selected spectra:

```
import ch.specchio.types.*

common_name_attr = specchio_client.getAttributesNameHash().get('Common');

e = MetaParameter.newInstance(common_name_attr);
e.setValue('Blackfern');
specchio_client.updateEavMetadata(e, ids);
```

- Open the metadata editor and check the bfern spectra: they should now all have a ‘Common’ Name Field filled with ‘Blackfern’.

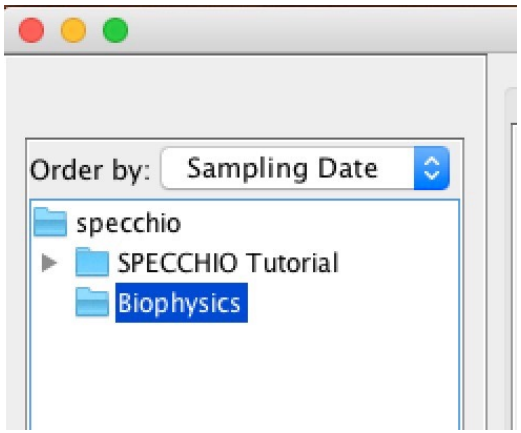


Creating a new campaign - Matlab

- Create a new campaign programmatically:

```
    c = SpecchioCampaign();
c.setName('Biophysics (Matlab)');

c_id = specchio_client.insertCampaign(c)
c.setId(c_id); % store the campaign id in our campaign object: required below ...
```

- The identifier of the new campaign is stored in the c_id variable
- Open the metadata editor and check the available campaigns, you should see your new campaign in the browser:

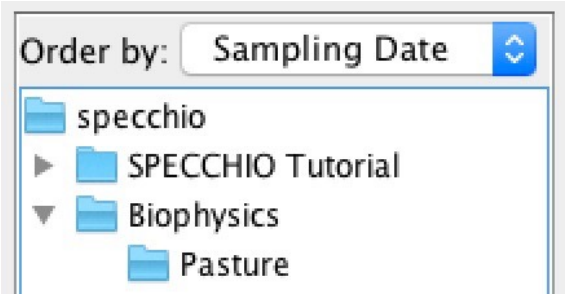


Creating a hierarchy - Matlab

- In this step we add a hierarchy to our new campaign (we could also choose to have no hierarchy and list all spectra just directly under the campaign, but hierarchies keep things tidy):

```
hierarchy_id = specchio_client.getSubHierarchyId(c, 'Pasture', 0);
```

- The zero argument signifies that the hierarchy has no parent hierarchy, i.e. it is directly connected to the campaign
- Open the metadata editor and check the available campaigns, you should see your new campaign with a new hierarchy added in the browser:



Input Files of this Exercise

- We have received to files[1] in CSV format that contain spectral data and metadata (spectra.csv and metadata.csv). Open the files in Excel and have a look at the content.

- Quite a lot of parameters have been established by laboratory measurements
- We could of course add all of them, but for simplicity, lets add 'Plot' which is a plot number, Nitrate Nitrogen and Phosphorus.
- If you check the existing metaparameters in SPECCHIO, you will notice that pretty much all of these parameters do not yet exist.
- We will use the 'Target Id' for the plot number, but we need to insert new attributes for Nitrate Nitrogen and Phosphorus

Plot	1.4	1.5	5.1	5.2
% Crude Protein	15.8	17.2	14.7	17
Nitrogen (Kjeldahl) %	2.6	2.9	2.4	2.7
Nitrate Nitrogen Mg/Kg	100	210	86	80
Phosphorus %	0.24	0.33	0.21	0.21
Potassium %	2	1.6	2.4	1.1
Sulfur %	0.25	0.28	0.21	0.25
Calcium %	0.32	0.3	0.28	0.34
Magnesium%	0.25	0.27	0.2	0.3
Sodium %	0.67	0.83	0.55	0.79
Chloride %	2.1	1.6	2	1.4
Manganese Mg/Kg	190	190	130	150
Iron Mg/Kg	120	110	110	150
Copper Mg/Kg	7.7	9.4	6.7	9.2
Zinc Mg/Kg	34	32	29	37
Molybdenum Mg/Kg	0.52	0.95	0.37	0.41
Boron Mg/Kg	6.5	5.2	5.6	2.5
N/P Ratio	11	8.8	11	13
N/K Ratio	1.3	1.8	1	2.5
N/S Ratio	10	10	11	11
Ammonium Nitrogen	160	160	120	170

[1] Kindly contributed by L. Suarez, RMIT, Melbourne, Australia

Adding new Attributes

- New attributes must be defined in the database using SQL. Our statements for this are:

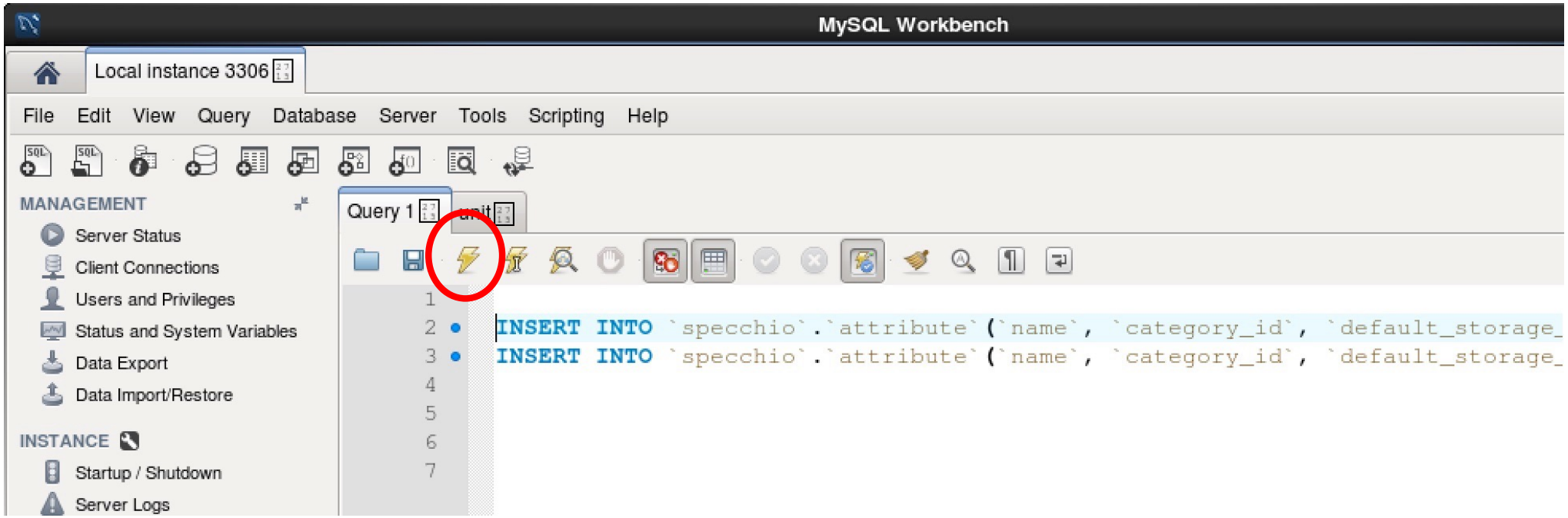
```
INSERT INTO `specchio`.`attribute`(`name`, `category_id`, `default_storage_field`, `default_unit_id`,
    `description`) VALUES ('Nitrate Nitrogen', (select category_id from `specchio`.category where name =
    'Vegetation Biophysical Variables'), 'double_val', (select unit_id from `specchio`.unit where short_name = '
'), 'Nitrate');

INSERT INTO `specchio`.`attribute`(`name`, `category_id`, `default_storage_field`, `default_unit_id`,
    `description`) VALUES ('Phosphorus', (select category_id from `specchio`.category where name =
    'Vegetation Biophysical Variables'), 'double_val', (select unit_id from `specchio`.unit where name =
    'Percent'), 'Phosphorus');
```

- Note: to select ‘Percent’ we use the ‘name’ field of the unit table, finding it via the short_name would require escaping the % character, which is a wildcard in SQL.

Adding new Attributes

- Open your virtual machine as root user (password is ‘reverse’)
- Open the MySQL Workbench (Applications/Programming/MySQL Workbench)

- Copy the two insert statements into the Query tab and press the ‘Run’ button



Adding new Attributes

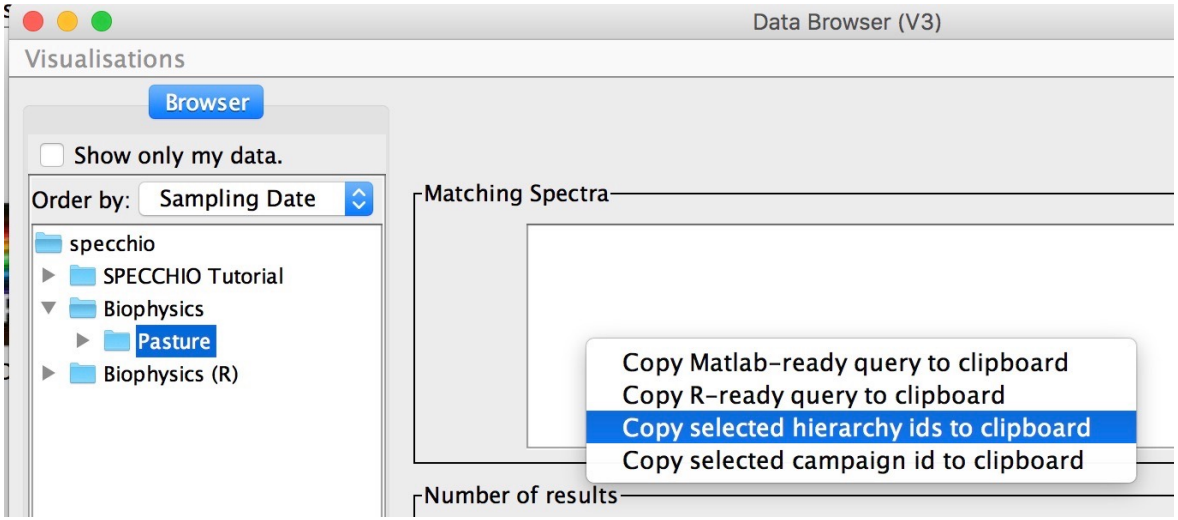
- To make the newly added attributes available a restart of the SPECCHIO server is required.
- You could simply close and open the VM; another solution is to stop and start the server using the command line:
  - Open a terminal in the VM and enter the following commands:
  - cd /opt/glassfish3/glassfish/bin
  - ./asadmin stop-domain
  - ./asadmin start-domain domain1
- Reconnect your SPECCHIO client to the database and the new attributes are now available

```
[root@SPECCHIOVM bin]# cd /opt/glassfish3/glassfish/bin
[root@SPECCHIOVM bin]# ./asadmin stop-domain
Waiting for the domain to stop ....
Command stop-domain executed successfully.
[root@SPECCHIOVM bin]# ./asadmin start-domain domain1
Waiting for domain1 to start ......
Successfully started the domain : domain1
domain  Location: /opt/glassfish3/glassfish/domains/domain1
Log File: /opt/glassfish3/glassfish/domains/domain1/logs/server.log
Admin Port: 4848
Command start-domain executed successfully.
[root@SPECCHIOVM bin]#
```

```
Vegetation Biophysical Variables
-----------------------------------------------
% Crown Cover  Crown Cover Percentage doub
Approx. Crown Diameter        double_val
Chlorophyll Content           double_val
Crown Class (FPMRIS)   SOP 13 Measuring a I
DBH     Diameter at breast height     doub
Dry Weight             double_val
Height  Height of vegetation   double_val
Leaf Area              double_val     cm2
Nitrate Nitrogen       Nitrate double_val
Phosphorus     Nitrate double_val     %
Specific Leaf Area     Calculated by: Leaf/
cm2/g
Water Content          double_val     g/cm
Wet Weight             double_val
-----------------------------------------------
```

Getting Campaign and Hierarchy IDs

- If you had to close you programming environment, then you may have forgotten the IDs of your new campaign and the hierarchy you just created.
- They can be easily retrieved using the SPECCHIO Query Browser: select the campaign or the hierarchy you require, then select the appropriate action from the pop-up menu in the 'Matching Spectra' text field:

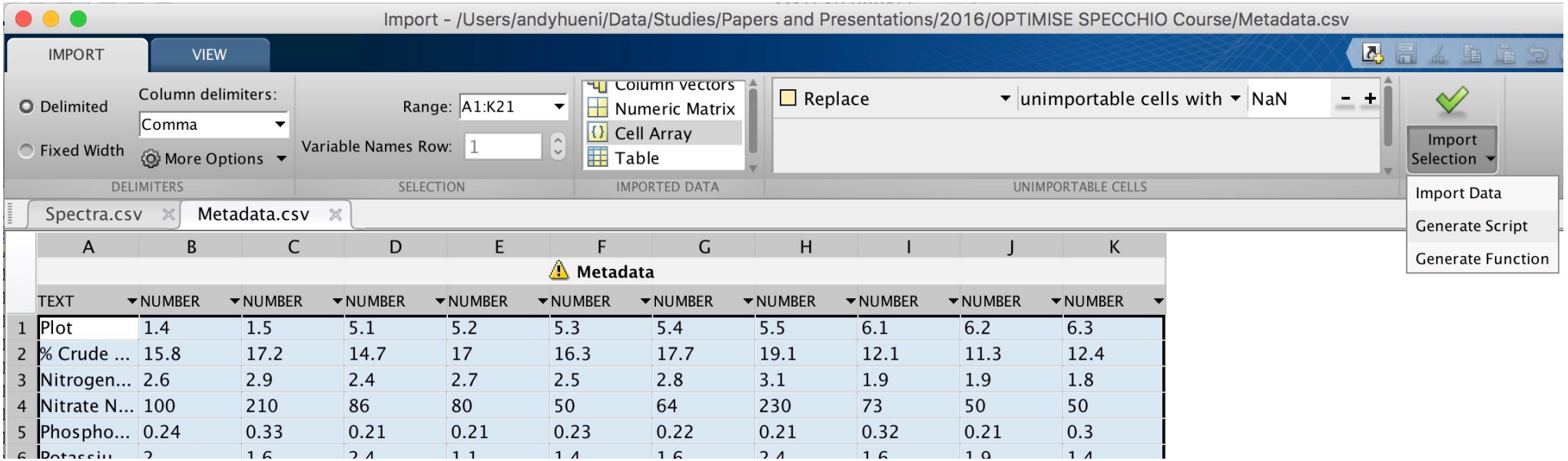


Reading and Processing the Input Files - Matlab

- Load the spectral CSV and the metadata CSV files into Matlab:

```
% load spectral file
filepath = '/Users/andyhueni/Data/Studies/Papers and Presentations/2016/OPTIMISE SPECCHIO Course';
filename = 'Spectra.csv';
wvls_and_spectra = csvread([filepath filesep filename]);
wvls = wvls_and_spectra(:,1);
spectra = wvls_and_spectra(:,2:end);
```

- To load the metadata file, the easiest is to use the 'Import Data' function in Matlab, telling it to create a cell array and create the code as well:



	A	B	C	D	E	F	G	H	I	J	K
	TEXT	NUMBER	NUMBER	NUMBER	NUMBER	NUMBER	NUMBER	NUMBER	NUMBER	NUMBER	NUMBER
1	Plot	1.4	1.5	5.1	5.2	5.3	5.4	5.5	6.1	6.2	6.3
2	% Crude ...	15.8	17.2	14.7	17	16.3	17.7	19.1	12.1	11.3	12.4
3	Nitrogen...	2.6	2.9	2.4	2.7	2.5	2.8	3.1	1.9	1.9	1.8
4	Nitrate N...	100	210	86	80	50	64	230	73	50	50
5	Phospho...	0.24	0.33	0.21	0.21	0.23	0.22	0.21	0.32	0.21	0.3
6	Potassiu...	2	1.6	2.4	1.1	1.4	1.6	2.4	1.6	1.9	1.4

Reading and Processing the Input Files - Matlab

There should now be three new variables in the Matlab workspace:
`wvls, spectra , metadata`

The following code generates a spectral file object, fills the spectral data into a Java Array and the metadata into a metadata object. The spectral file object is then stored in the database under the campaign and hierarchy we created.

```
%% create spectral file object and fill in some info …
s = ch.specchio.types.SpectralFile;
s.setNumberOfSpectra(size(spectra,2));
s.setPath(filepath);
s.setFilename(filename);
s.setCompany('ASD');

% set campaign and hierarchy to store in
s.setHierarchyId(hierarchy_id);
s.setCampaignId(c_id);

% create array for spectral data
spectra_array = javaArray('java.lang.Float',size(spectra,2),length(wvls));

% create wvls array and fill it
java_wvls = javaArray('java.lang.Float',length(wvls));

for w=1:length(wvls)
    java_wvls(w) = java.lang.Float(wvls(w));
end
```

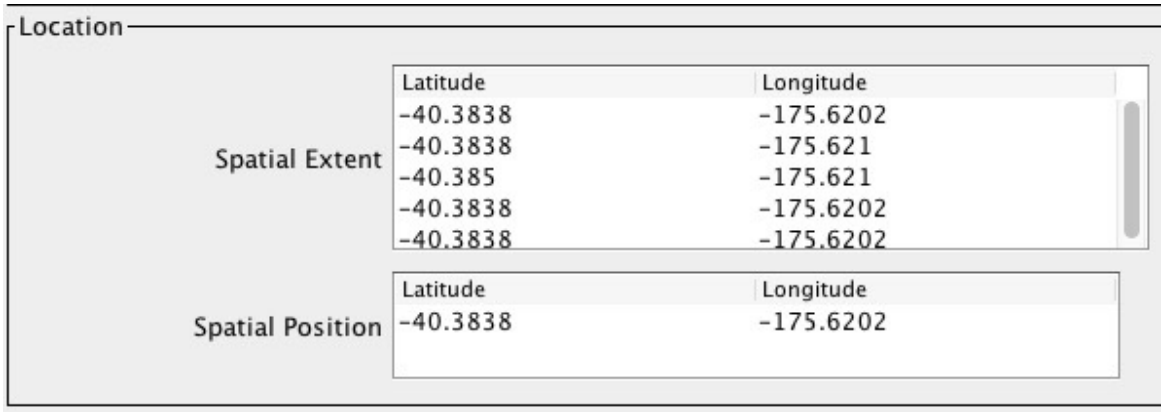
```
% add spectral data and metadata
for i=1:size(spectra,2)

    % fill vector in array
    vector = spectra(:,i);
    for w=1:length(wvls)
        spectra_array(i,w) = java.lang.Float(vector(w));
    end

    % add wvls (done for each spectrum individually)
    s.addWvls(java_wvls);

    % add filename: here we add an automatic number to make them distinct
    s.addSpectrumFilename([filename ' ' num2str(i)]);

    % add plot number
    smd = ch.specchio.types.Metadata();
    mp = MetaParameter.newInstance(specchio_client.getAttributesNameHash().get('Target ID'));
    mp.setValue(num2str(metadata{1,i+1})); % meta values start in the 2 column, Plot is in the 1 row
    smd.addEntry(mp);

    % add Nitrate
    mp = MetaParameter.newInstance(specchio_client.getAttributesNameHash().get('Nitrate Nitrogen'));
    mp.setValue(metadata{4,i+1}); % meta values start in the 2 column
    smd.addEntry(mp);

    % add Phosphorus
    mp = MetaParameter.newInstance(specchio_client.getAttributesNameHash().get('Phosphorus'));
    mp.setValue(metadata{5,i+1}); % meta values start in the 2 column
    smd.addEntry(mp);

    s.addEavMetadata(smd); % add the metadata structure to the spectral file object
end

s.setMeasurements(spectra_array); % set the measurement array (all spectral data)

specchio_client.insertSpectralFile(s);
```

Spatial Operations (SPECCHIO V3.3.0.0)

Spatial operations are supported from SPECCHIO V3.3.0.0 onwards, basing on the MySQL spatial extension (Note: MySQL may not have the best spatial support but at this point we have no resources to port to e.g. PostGIS).

How spatial are handled in SPECCHIO:

- Supports point, polyline and polygon geometries. These are implemented as new attributes in the SPECCHIO metadata system: 'Spatial Position', 'Spatial Transect' and 'Spatial Extent'
- Supports spatial searches by polygon
- New entry fields and report fields for spatial data in the Metadata Editor and Spectrum Report:



Inserting Spatial Metadata: select spectra to be updated

```
import ch.specchio.types.*;
 % Query pasted from SPECCHIO Query builder and manually edited to just set conditions for the File
    Name attribute:
  %>>>>>>>>>>>>>>>>>>>>>
    query = Query('spectrum');
    query.setQueryType(Query.SELECT_QUERY);

    query.addColumn('spectrum_id')

    cond = EAVQueryConditionObject('eav', 'spectrum_x_eav', 'File Name', 'string_val');
    cond.setValue('%%cabbage%%'); % select spectra with a filename like 'cabbage'
    cond.setOperator('like');
    query.add_condition(cond);


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

Inserting Spatial Metadata: spatial position

```
%% Spatial Position insert for all selected spectra
   % create new coordinate object point: (lat, lon)
   coord = Point2D(-40.3838, -175.6202);

   % create point metaparameter to add to spectra
   spatial_point_attribute = specchio_client.getAttributesNameHash().get('Spatial Position');
   mp = MetaParameter.newInstance(spatial_point_attribute);
   mp.setValue(coord);

% insert metaparameter
   specchio_client.updateEavMetadata(mp, ids);

   % see how the point is represented in MySQL syntax
   disp(mp.getEAVValue());
```

Inserting Spatial Metadata: spatial extent

```
%% create a polygon and add to spectra
   % create list of coordinates to define polygon (polygon gets closed
   % automatically by SPECCHIO before insert)
   coords = java.util.ArrayList();
   coord = Point2D(-40.3838, -175.6202);
   coords.add(coord);
   coord = Point2D(-40.3838, -175.6210);
   coords.add(coord);
   coord = Point2D(-40.3850, -175.6210);
   coords.add(coord);
   coord = Point2D(-40.3838, -175.6202);
   coords.add(coord);

   % create point metaparameter to add to spectra
   spatial_extent_attribute = specchio_client.getAttributesNameHash().get('Spatial Extent');
   mp = MetaParameter.newInstance(spatial_extent_attribute);
   mp.setValue(coords);
   % insert metaparameter
   specchio_client.updateEavMetadata(mp, ids);
   % see how the point is represented in MySQL syntax
   disp(mp.getEAVValue());
```

Spatial Selection

```
% Query pasted from SPECCHIO Query builder and manually edited to just set conditions for the File
   Name attribute:
   % >>>>>>>>>>>>>>>>>>>>>>
   query = Query('spectrum');
   query.setQueryType(Query.SELECT_QUERY);


   query.addColumn('spectrum_id')


   % select spectra with spatial position inside polygon
   cond = EAVQueryConditionObject('eav', 'spectrum_x_eav', 'Spatial Position', 'spatial_val');
   cond.setValue('ST_GeomFromText(''Polygon((-40 -175.6,-40 -176.8,-41.2 -176.8,-41.2 -176.8,-40 -
   175.6))'')');
   cond.setOperator('MBRContains'); % MySQL spatial operator
   query.add_condition(cond);


   ids = specchio_client.getSpectrumIdsMatchingQuery(query);


   %<<<<<<<<<<<<<<<<<<<<<<<<


   disp(ids.size());
```

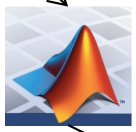
General Processing Workflow

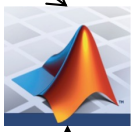
Spectral Files

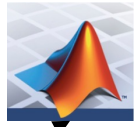
Meta Files

Data Augmentation: add metadata to spectra: Matlab/R are the most flexible solutions. SPECCHIO GUI can also work.

Data Analysis: easiest in Matlab/R







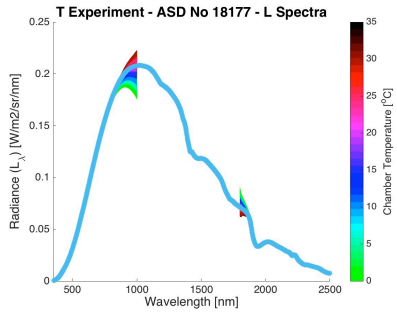
Insert

Select/Insert

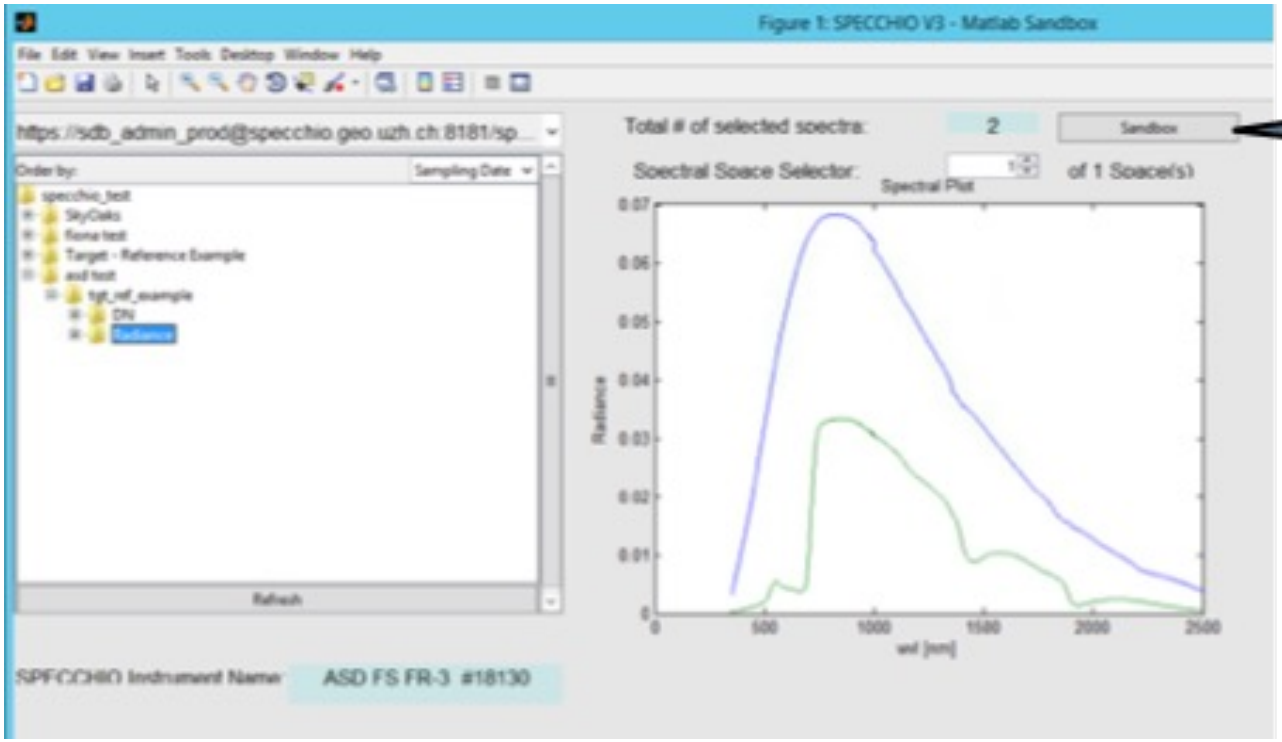
Data ingest:
Either via SPECCHIO GUI, or if no reader exists, use our own implementation in Matlab or R

SPECCHIO DB

SPECCHIO Server



Matlab Interactive GUI Sandbox



Sandbox function

```
function SPECCHIO_Sandbox(hObject, EventData)

    fh = ancestor(hObject.hghandle, 'figure');
    user_data = get(fh, 'UserData');


    %% Do whatever needs doing with the data ....


    % e.g. get the mean:

    avg_spectrum = mean(user_data.current_spectra.vectors);

    % do a plot
    figure
    plot(user_data.current_spectra.wvl, user_data.current_spectra.vectors);

end
```

Matlab Central

File Exchange

specchio_interactive_sandbox_GUI()

by Andy Hueni
11 Oct 2014

Interactive GUI for Data Selection from the SPECCHIO Spectral Information System

Watch this File

Be the first to rate this file!

15 Downloads (last 30 days)
File Size: 4.85 KB
File ID: #48082
Version: 1.0

Download Zip

Code covered by the BSD License

File Information

Description Provides the basic functionality to interactively browse the data provided by a SPECCHIO Spectral Information System. Server connections are automatically loaded into the Matlab GUI when the class path is set up as detailed in the SPECCHIO Matlab Guide. For more details see the documentation on www.specchio.ch

Acknowledgements Using Java Swing Components In Matlab inspired *this file*.

MATLAB release MATLAB 7.12 (R2011a)

MATLAB Search Path
```
/
/SPECCHIO_Matlab_GUI_Sandbox
```

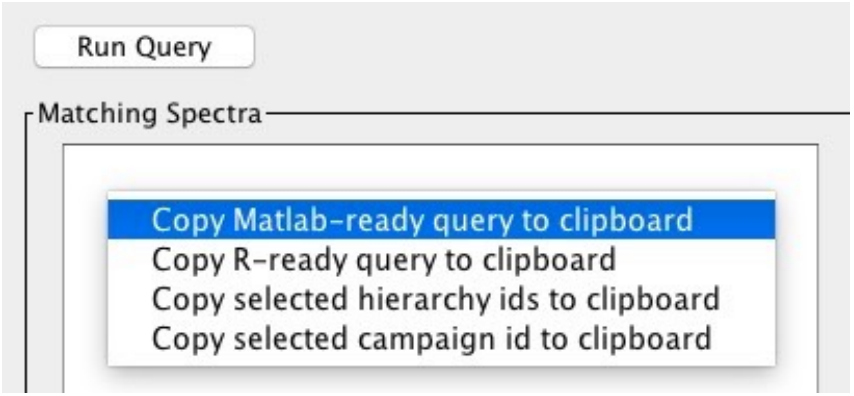
Other requirements SPECCHIO Java Application must be installed and one or more accounts must have been created on one or more SPECCHIO servers.

Highlights from specchio_interactive_sandbox_GUI()

fx plot_2d(figure_handle, da...
fx specchio_interactive_sand...

» View all files

Data Selection: Auto-generated Code from SPECCHIO Query Builder GUI




```
query = Query('spectrum');
query.setQueryType(Query.SELECT_QUERY);

query.addColumn('spectrum_id')

cond = EAVQueryConditionObject('eav', 'spectrum_x_eav', 'Instrument Temperature', 'double_val');
cond.setValue('15.0');
cond.setOperator('>=');
query.add_condition(cond);

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

Exporting data to CSV

Data can be exported from code by calling SPECCHIO classes and using the export GUI:

Participants Feedback

- How was the course?
- What should be added or changed?
- Was the course too short, too long, just right?
- What programming user help would you like to see?

Please send me your feedback by email before end of course

ahueni@geo.uzh.ch

Some of our ideas what could be done:

- Have a project with input data and analysis goals to work through as a team
- Exercise on program structuring and where to use which SPECCHIO function

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 EUROPEAN COOPERATION IN SCIENCE AND TECHNOLOGY

OPTIMISE

cost ES0903 EUROPEAN COOPERATION IN SCIENCE AND TECHNOLOGY