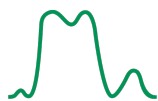# R accessing SPECCHIO

Andy Hueni[1]

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

SPECCHIO Programming Course

Zurich 2017

SPECCHIO

OPTIMISE

EUROPEAN COOPERATION
IN SCIENCE AND TECHNOLOGY

COST is supported by the EU Framework
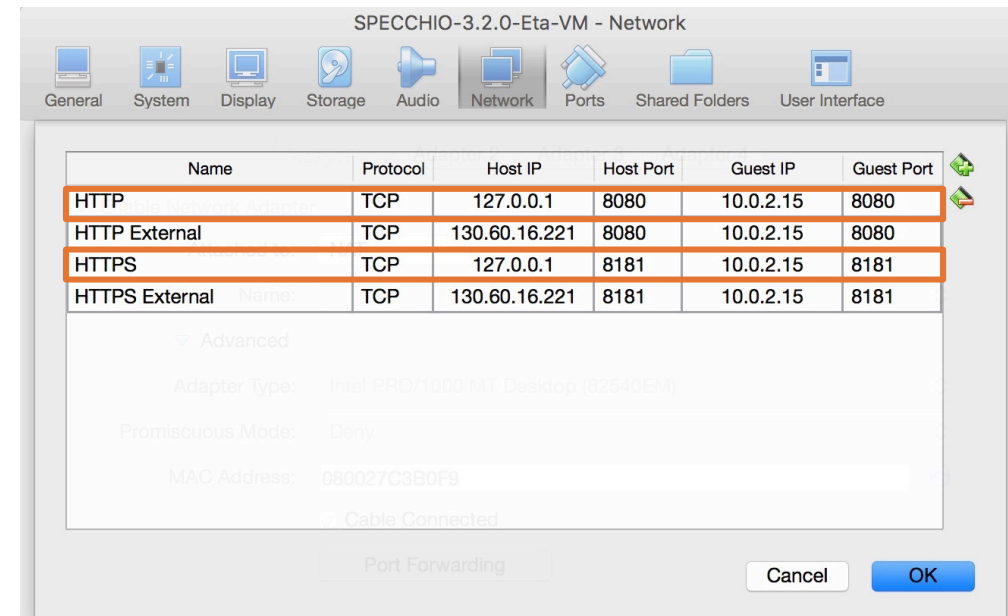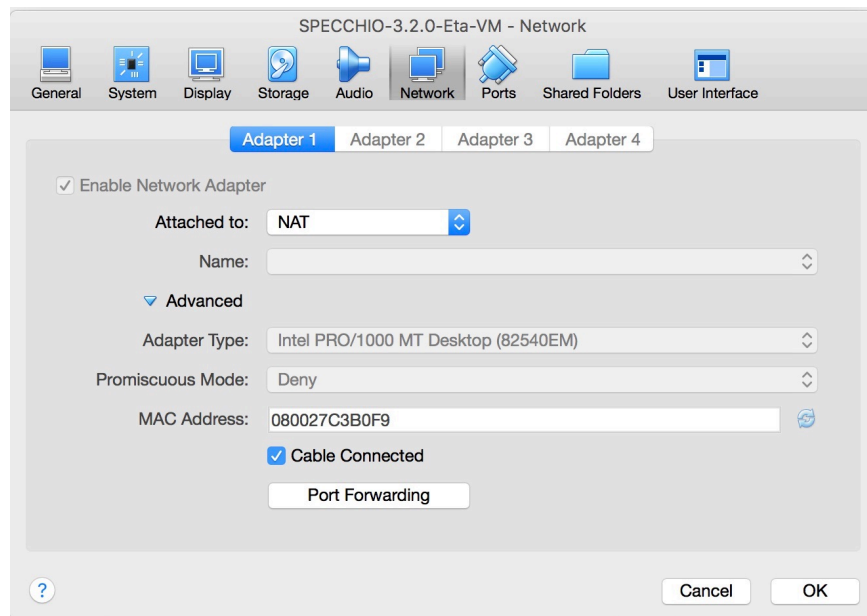Programme Horizon 2020

EARSeL

Agenda

- Configuring your machine for SPECCHIO access from R

- 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

- Matlab GUI demonstration

- Resources: example code

- Data Selection: Auto-generated Code

RSL
measurements | products | policy

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

SPECCHIO-3.2.0-Eta-VM - Network

| General | System | Display | Storage | Audio | Network | Ports | Shared Folders | User Interface |

Adapter 1 | Adapter 2 | Adapter 3 | Adapter 4

☑ Enable Network Adapter

Attached to: NAT

Name:

▽ Advanced

Adapter Type: Intel PRO/1000 MT Desktop (82540EM)

Promiscuous Mode: Deny

MAC Address: 080027C3B0F9

☑ Cable Connected

Port Forwarding

Cancel   OK

SPECCHIO-3.2.0-Eta-VM - Network

| Name	Protocol	Host IP	Host Port	Guest IP	Guest Port
HTTP	TCP	127.0.0.1	8080	10.0.2.15	8080
HTTP External	TCP	130.60.16.221	8080	10.0.2.15	8080
HTTPS	TCP	127.0.0.1	8181	10.0.2.15	8181
HTTPS External	TCP	130.60.16.221	8181	10.0.2.15	8181

Cancel   OK

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 (http://specchio.ch/app_download.php) 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.
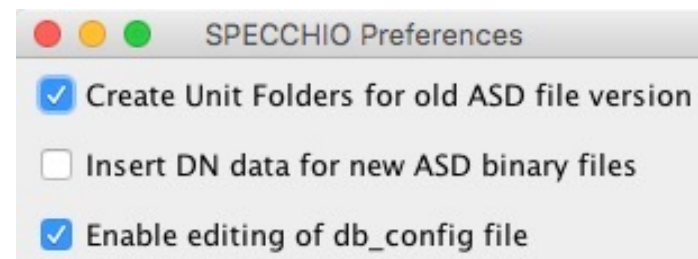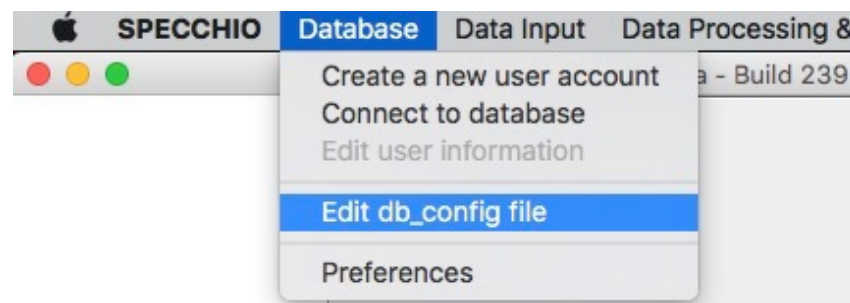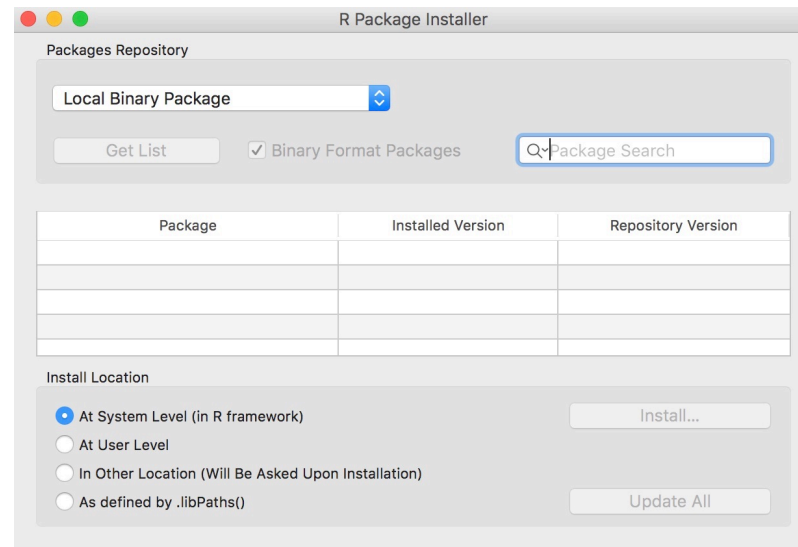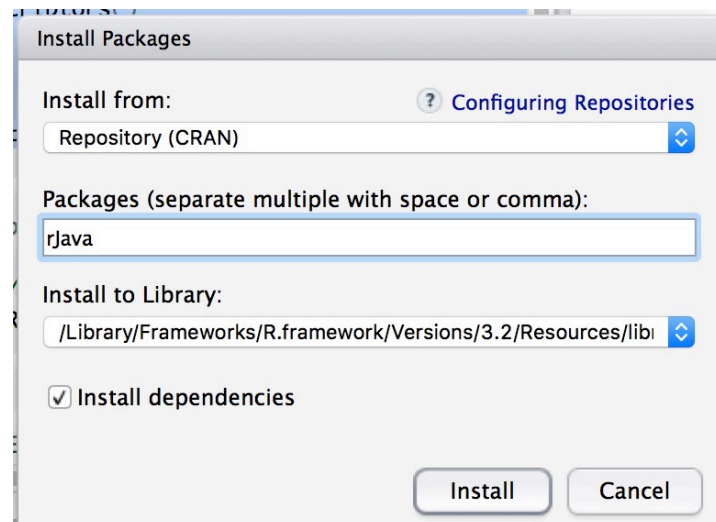


```
http, localhost, 8080, /specchio_service, sdb_admin, 5p3cch10, jdbc/specchio
```

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

Connecting to SPECCHIO - R

Download rJava and install it.

Connecting to SPECCHIO - R

- Create a new R script in R Studio, test each of the lines below as you enter them.
- Load the rJava library, init rJava, ddd 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.

```
library(rJava)

.jinit()
.jaddClassPath(path="/Applications/SPECCHIO/SPECCHIO.app/Contents/Java/specchio-client.jar")
```

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

```
cf <- J("ch/specchio/client/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(as.integer(0)))
```

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

Connecting to SPECCHIO - R

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

```
db_descriptor_list$get(as.integer(0))
```

- Note: the index must be explicitly given as integer, otherwise the method is not recognised!

Selecting data - R

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 specchio types package:
```
.jaddClassPath(path="/Applications/SPECCHIO/SPECCHIO.app/Contents/Java/specchio-types.jar")
```

- Create a query object:
```
query <- .jnew("ch/specchio/queries/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 <- .jnew("ch/specchio/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 - R

- Check how many spectra we found

```
ids$size()
```

- List the spectrum ids:

```
ids$toString()
```

- Add another condition to restrict the maximum altitude:

```
cond <- .jnew("ch/specchio/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 - R

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 an R array and are therefore indexed starting at 1):

```
space <- spaces[[1]] # first element of returned array
```

- 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 properly ordered ids!
```

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

```
space <- specchio_client$loadSpace(space);
```

Load spectral data from database - R

- Get the spectral vectors as R array:

```
vectors <- space$getVectorsAsArray()
# convert java array to R matrix
r_matrix <- .jevalArray(vectors, simplify=TRUE)
```

- Get the wavelengths are a vector:

```
wvls <- space$getAverageWavelengths()
```

- Get the unit of the spectra:

```
unit = space$getMeasurementUnit()$getUnitName()
```

- Plot the spectra:

```
plot(r_matrix[1,]~wvls,type="n", ylim=c(0, 1), xlab="Wavelength [nm]" , ylab=unit) # create empty plot
col = rainbow(space$getNumberOfDataPoints())


for (i in 1:space$getNumberOfDataPoints() ) {
  lines(r_matrix[i,]~wvls, col=col[i])
}
```

Load metadata from database - R

- 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 (not really nice, but those using R should know how to do it properly):

```
# fill legend string
legendstr <- c(length = filenames_$size())

for (i in 1:filenames_$size() ) {
  legendstr[i]=filenames_$get(as.integer(i-1))
}

legend(500, 0.8, legendstr)
```

- Get lat/lon as vectors and plot them:

```
lat <- specchio_client$getMetaparameterValues(ids, 'Latitude')$get_as_double_array()
lon <- specchio_client$getMetaparameterValues(ids, 'Longitude')$get_as_double_array()

plot(lat~lon)
```

Adding metadata - R

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:

```
query <- .jnew("ch/specchio/queries/Query")
attr <- specchio_client$getAttributesNameHash()$get("File Name")

cond <- .jnew("ch/specchio/queries/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 - R

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

```
common_name_attr  <- specchio_client$getAttributesNameHash()$get("Common")


e <- J("ch/specchio/types/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'.
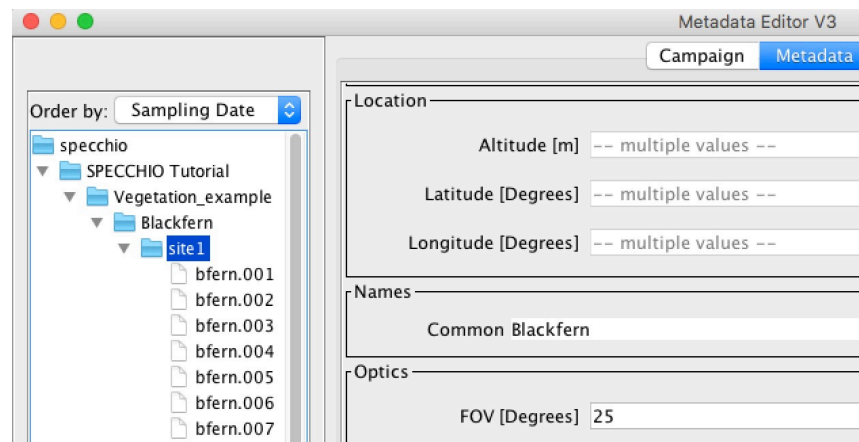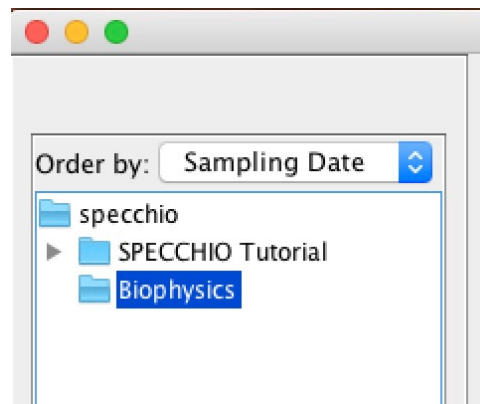
Creating a new campaign - R

- Create a new campaign programmatically:

```
c <- .jnew("ch/specchio/types/SpecchioCampaign")
c$setName('Biophysics (R)');
c_id <- specchio_client$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 - R

- 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', as.integer(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:
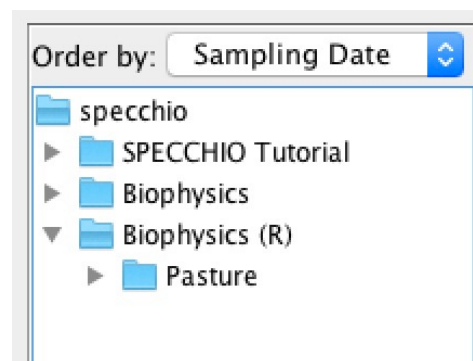
Input Files of this Exercise

- 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
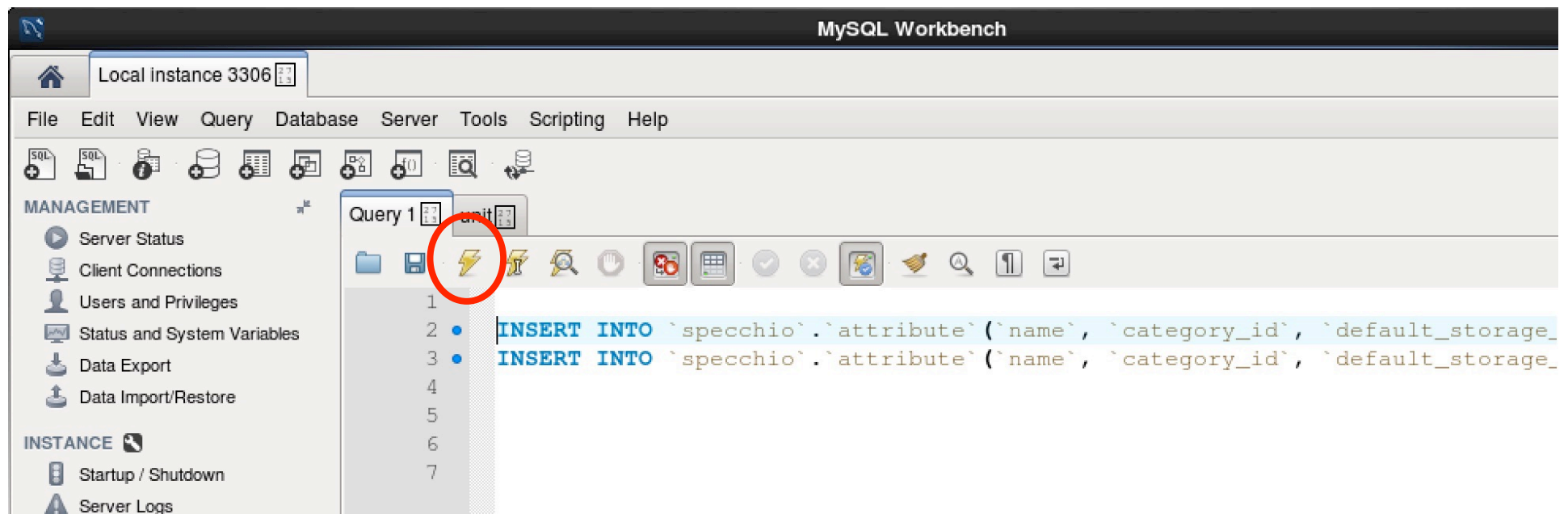
Adding new Attributes

- 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
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:
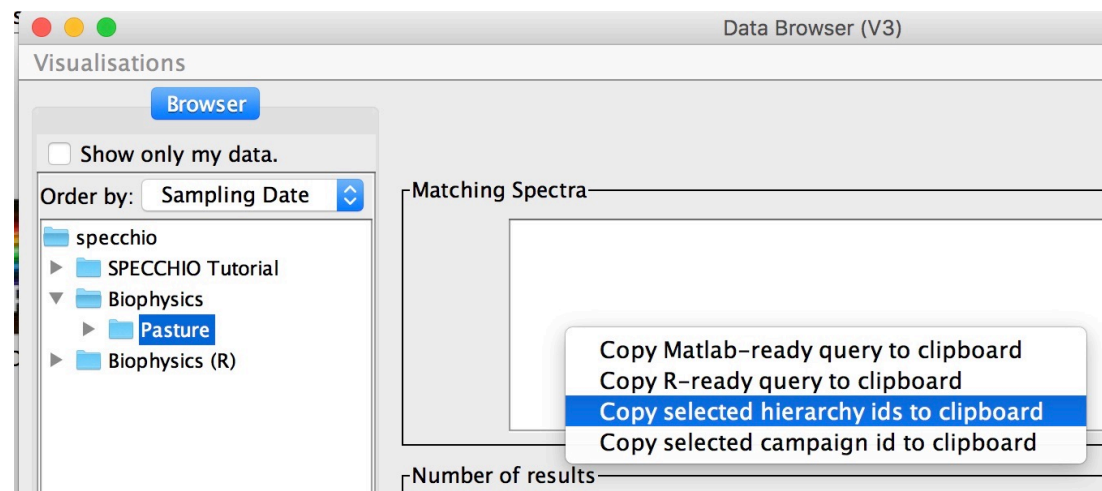
Reading and Processing the Input Files - R

- To be done by someone proficient in R – if you got a nice solution please send it to ahueni@geo.uzh.ch
- For SPECCHIO functions to be used please see the Matlab version of this presentation
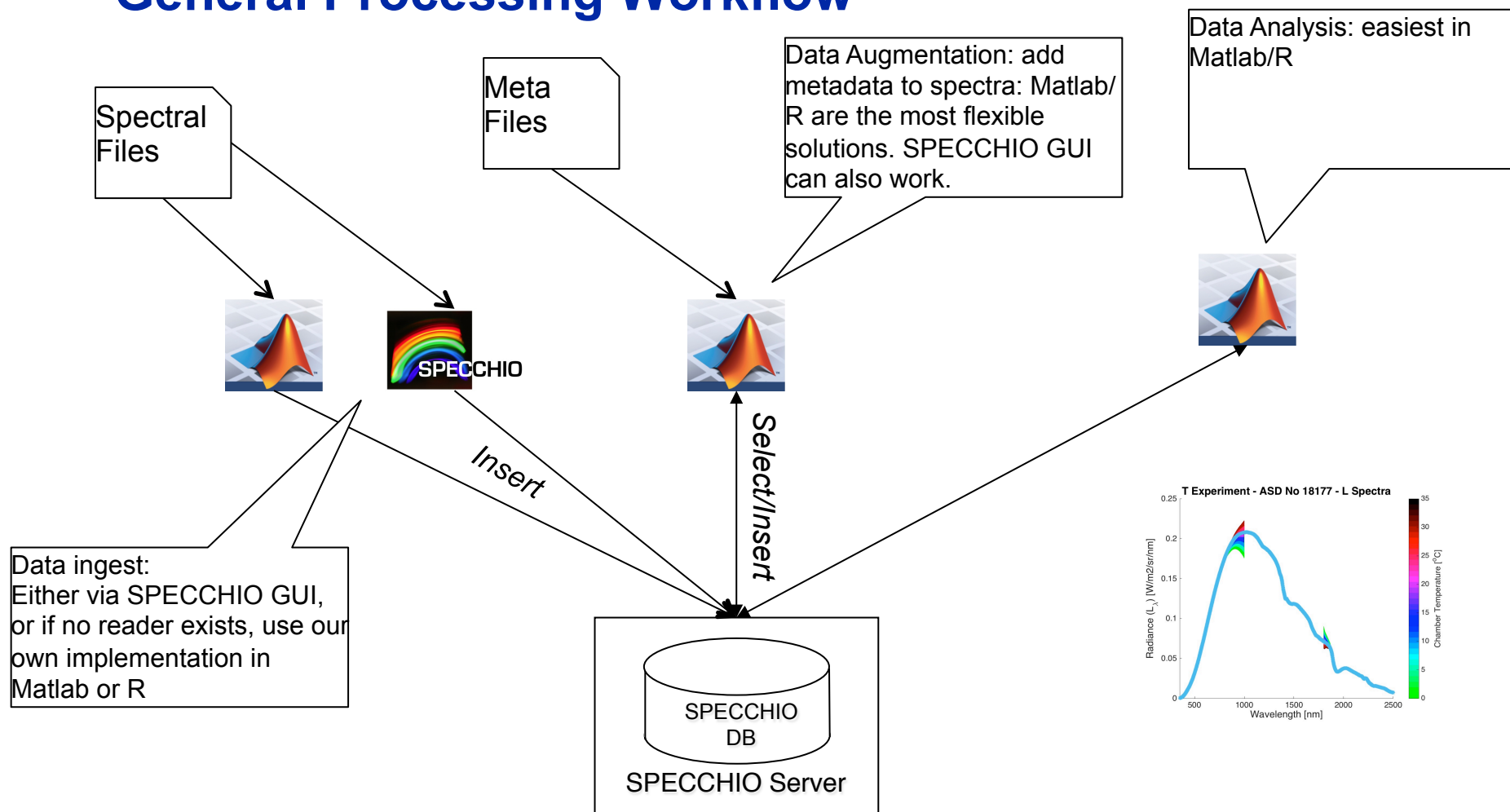
Reading and Processing the Input Files - R

- Code yet to be written; essentially a copy of the Matlab version …
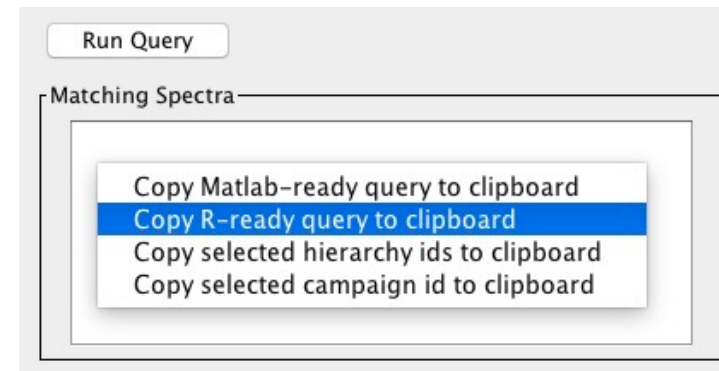
General Processing Workflow

Data Selection: Auto-generated Code


```
query <- .jnew("ch/specchio/queries/Query", "spectrum")
query$setQueryType(query$SELECT_QUERY)

query$addColumn("spectrum_id")

cond <- .jnew("ch/specchio/queries/EAVQueryConditionObject", "eav", "spectrum_x_eav", "Instrument Temperature", "double_val");
cond$setValue("15.0");
cond$setOperator(">=");
query$add_condition(cond);

ids <- specchio_client$getSpectrumIdsMatchingQuery(query);
```

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