Dynamo

Tutorial on tools for Classification and Visualization

20/January/2012

GOALS

In this tutorial we present a extremely simple artificial data set. We will use it to get familiar with:

- general classification tools
  - creation of distance matrices
  - using PCA analysis
- visualization tools for large data sets (`dynamo_gallery`)
- visualization tools for small sets of volumes (`dynamo_mapview, dynamo_slices`)
- visualization tools for alignment and classification results coded in “table” files (`dynamo_tableview`)

SYNTAX

This tutorial assumes that you have already followed the basic tutorial and that you are familiar with the basic concepts of *Dynamo*, as projects, or basic syntax in Matlab or in the Linux/MacOS/Windows command shell.

Normally we will indicate commands in their Matlab version.
You can also use the Linux/MacOS/Windows corresponding commands, with the obvious adaptions

Creating the data set

Type in Matlab:

```
>> t=dynamo_tutorial('tutorial_ccmatrix','M',8,'N',8,'project','project_for_tutorial_ccmatrix');
```

This creates:
* 8 particles of one class (M) and 8 particles of another class (N)
* a *Dynamo* project with numerical settings for the alignment procedure
   together with auxiliary files (as masks, templates, etc)

NOTE:

In standalone modus, the Linux/MacOS shell syntax for this command would read:

```
$ dynamo tutorial tutorial_ccmatrix -M 8 -N 8 -project project_for_tutorial_ccmatrix
```

and in the Windows cmd shell:

```
> dynamo.exe tutorial tutorial_ccmatrix -M 8 -N 8 -project project_for_tutorial_ccmatrix
```

Inspecting the data set

We could use the data browser `dynamo_gallery` tool for a closer examination (later).

For a quick view we can type in Matlab:

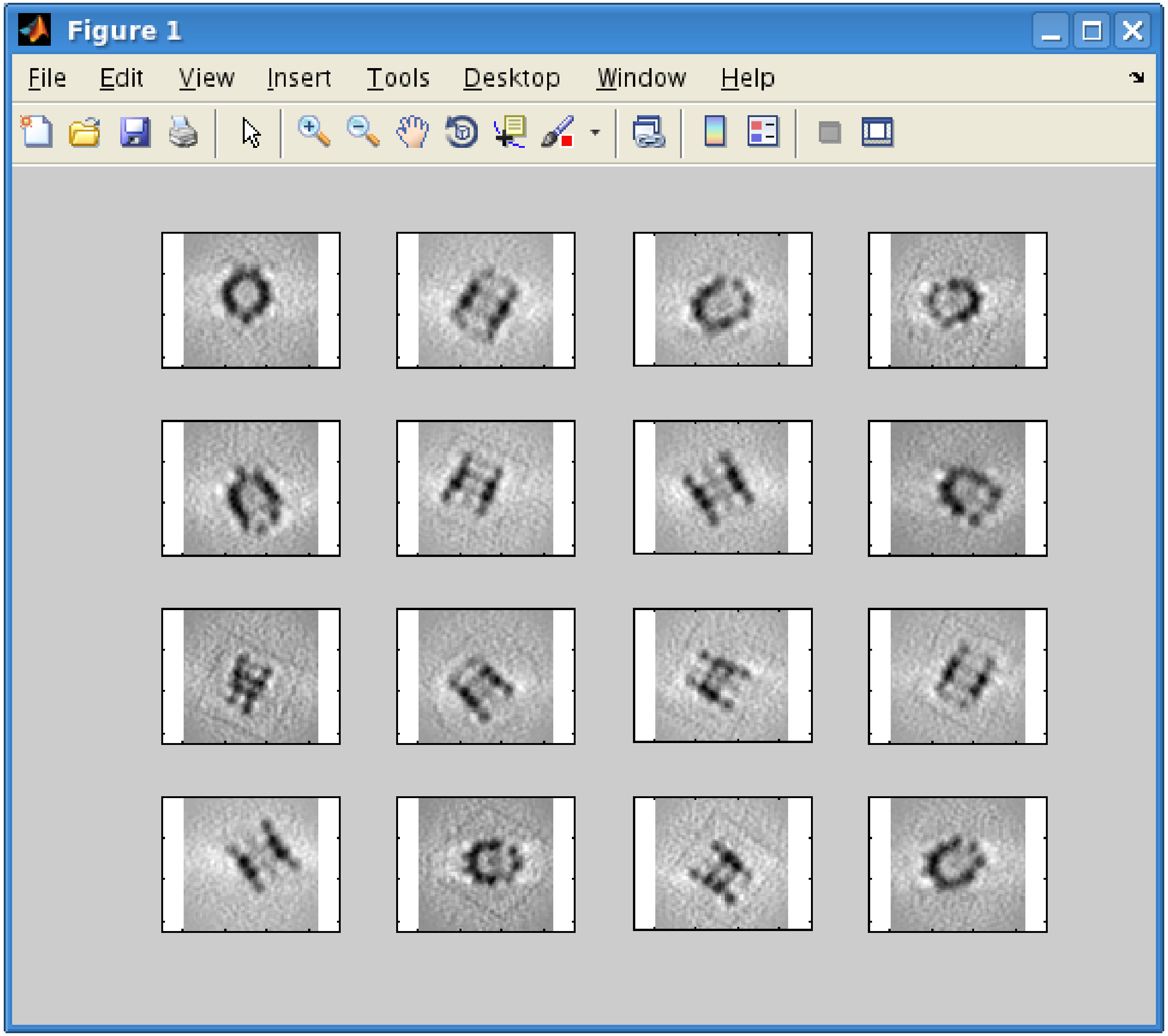
```
>> p0=dynamo_particle('project_for_tutorial_ccmatrix','tag','*');
```

This writes to the Matlab workspace variable `p0` all the particles (`'tag', '*'`) associated to the project.

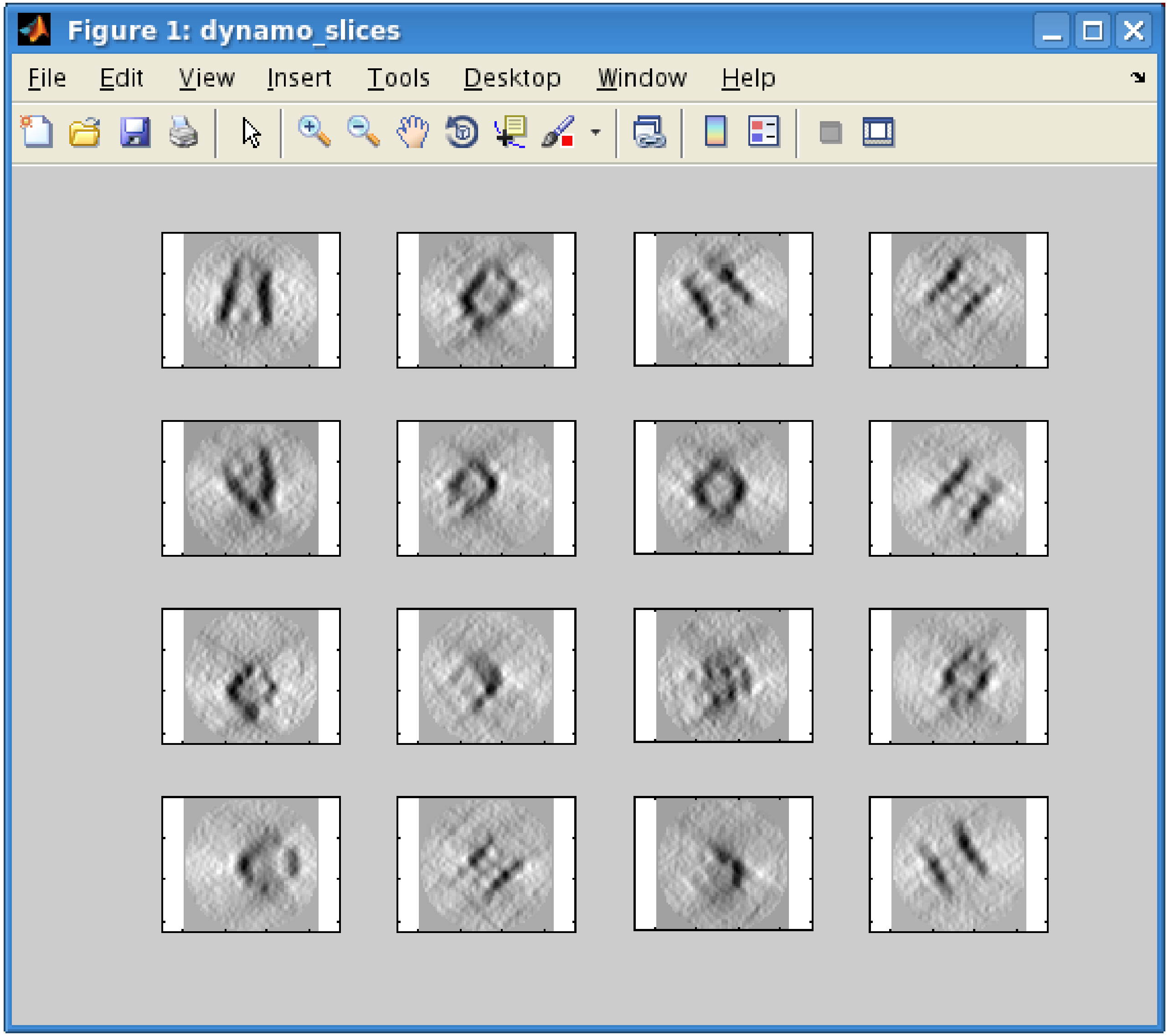
Now, we can visualize them with:

```
>> dynamo_slices(p0,'range',33,'panel',1);
```

```
>>dynamo_slices(p0,'range',33,'panel',1);
```



This is the z-view of all the particles assigned to the project (showing the section #33 of the cube)
Particles are initially randomly oriented.



```
>>dynamo_slices(p0,'y','range',33,'panel',1);
```

Same particles viewed from the 'y' direction, showing the effect of the missing wedge

Ok, so in theory the tutorial generated data belonging to “two classes” of particles....

but...

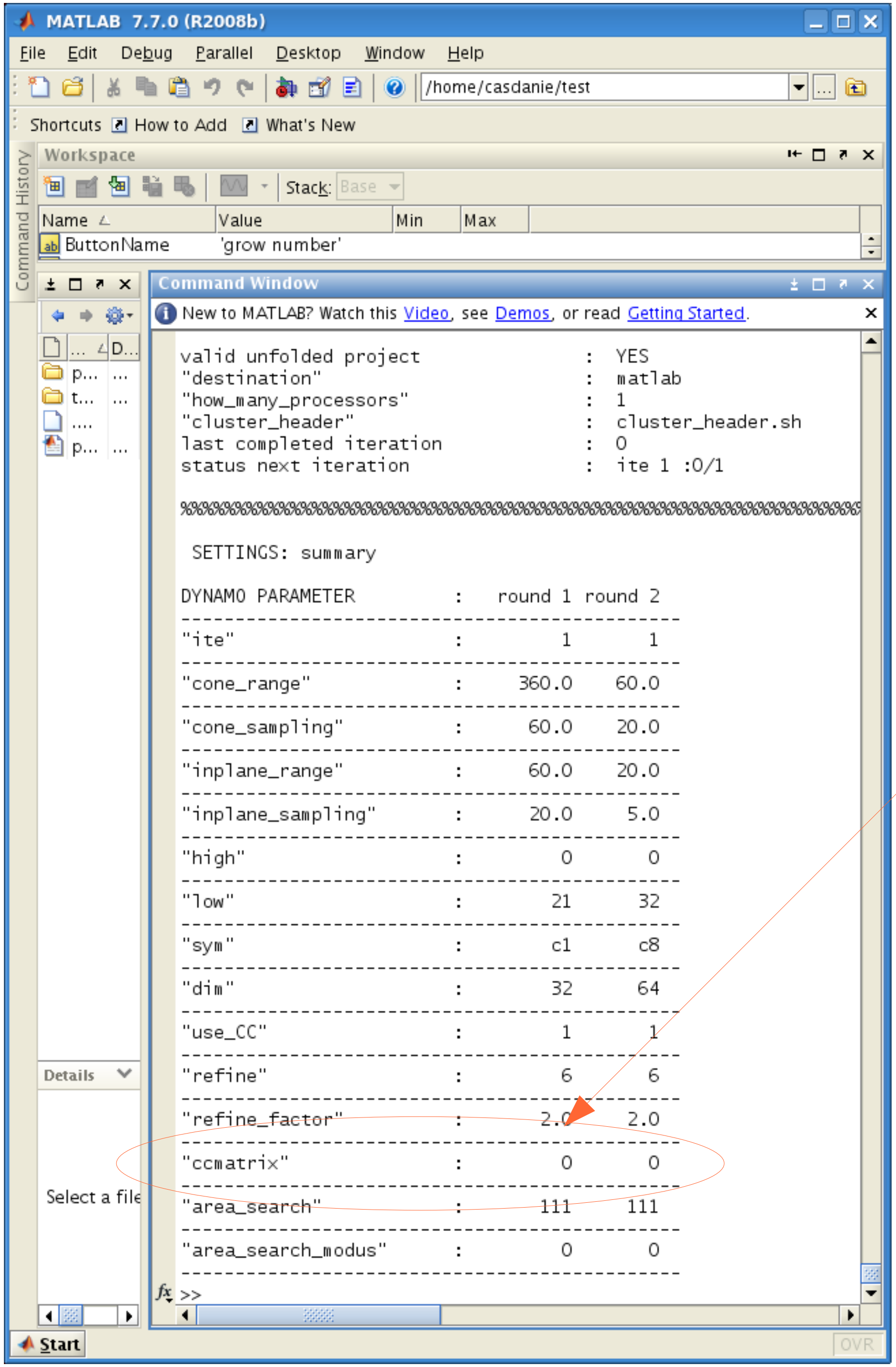
Do you recognize which particle is in which class?

Do you recognize which features define each class?

... probably not without aligning them first.

The tutorial that you just generated includes an alignment “project”

We will edit and use this project to align the particles and, simultaneously, provide us with ground information to perform afterwards a classification.



```
valid unfolded project                       :  YES
"destination"                                :  matlab
"how_many_processors"                        :  1
"cluster_header"                             :  cluster_header.sh
last completed iteration                     :  0
status next iteration                        :  ite 1 :0/1

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 SETTINGS: summary

DYNAMO PARAMETER            :    round 1  round 2
-----------------------------------------------------
"ite"                       :          1        1
-----------------------------------------------------
"cone_range"                :      360.0     60.0
-----------------------------------------------------
"cone_sampling"             :       60.0     20.0
-----------------------------------------------------
"inplane_range"             :       60.0     20.0
-----------------------------------------------------
"inplane_sampling"          :       20.0      5.0
-----------------------------------------------------
"high"                      :          0        0
-----------------------------------------------------
"low"                       :         21       32
-----------------------------------------------------
"sym"                       :         c1       c8
-----------------------------------------------------
"dim"                       :         32       64
-----------------------------------------------------
"use_CC"                    :          1        1
-----------------------------------------------------
"refine"                    :          6        6
-----------------------------------------------------
"refine_factor"             :        2.0      2.0
-----------------------------------------------------
"ccmatrix"                  :          0        0
-----------------------------------------------------
"area_search"               :        111      111
-----------------------------------------------------
"area_search_modus"         :          0        0
-----------------------------------------------------
>>
```

Seeing the project

In Matlab:

```
>>dynamo_vpr_info project_for_tutorial_ccmatrix;
```

In Linux/MacOS:

```
$ dynamo vpr_info project_for_tutorial_ccmatrix
```

You should see something similar to the image on the left.

The "`ccmatrix`" *Dynamo* parameter is set to zero.
this means that after each alignment iteration,
Dynamo will not classify the newly aligned particles

In this tutorial we want to create a project that computes
The basic piece of a classification procedure, the "ccmatrix"
(constrained covariance matrix) after each alignment iteration.

We will edit the project to include these computations.

Editing the project

We want to change the values of parameter “ccmatrix” to 1 in rounds 1 and 2.

We operate on a virtual project that copies the contents of the existing project:

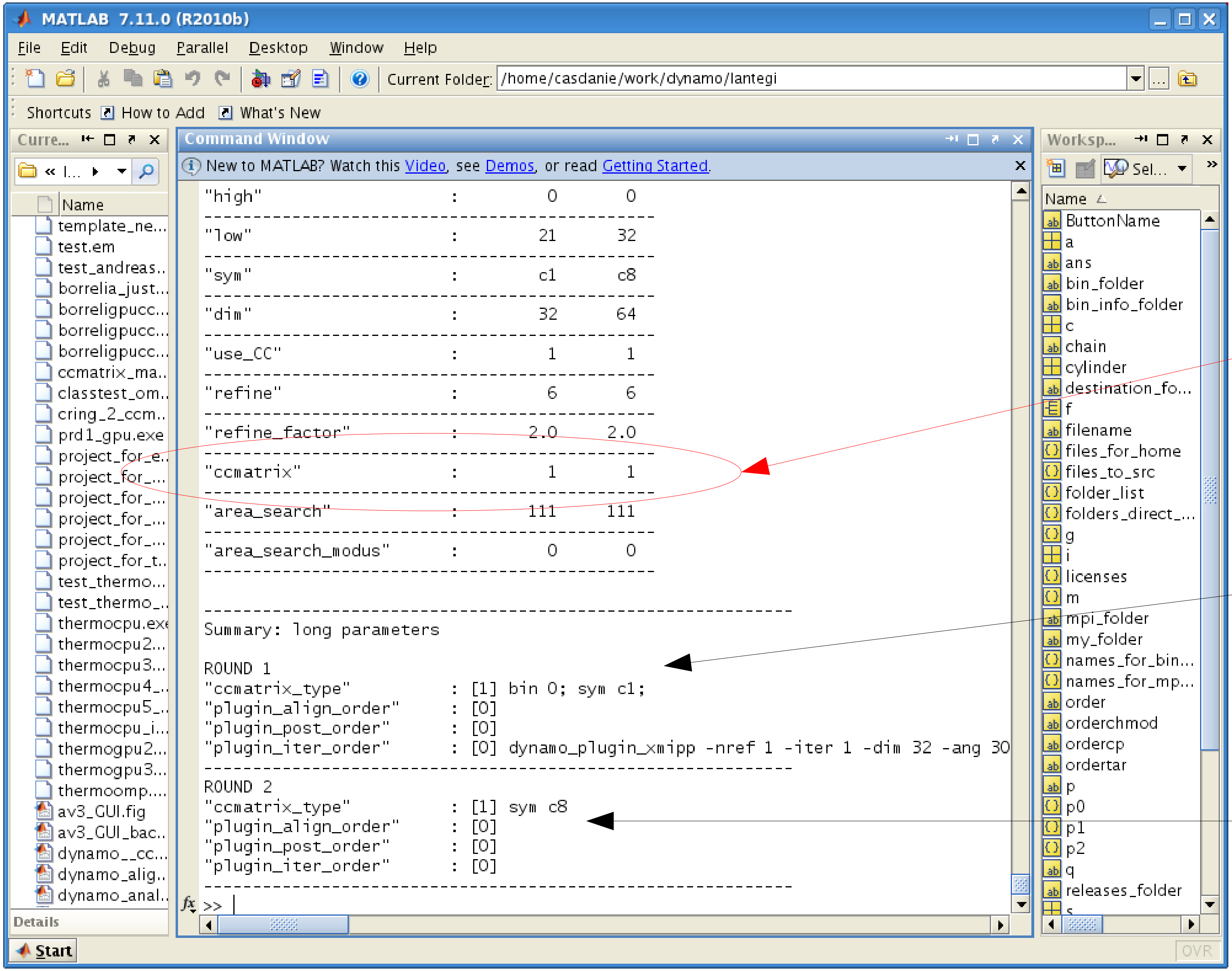
```
>> vpr=dynamo_vpr_modify('project_for_tutorial_ccmatrix','ccmatrix_r1',1,'ccmatrix_r2',1);
```

And then save the modified virtual project into the hard disk as a new project

```
>> dynamo_vpr_unfold(vpr);
```

Note that we have not changed the name of the project, so we can check again the status of the project in the disk:

```
>>dynamo_vpr_info project_for_tutorial_ccmatrix;
```



```
"high"                  :         0        0
-----------------------------------------------
"low"                   :        21       32
-----------------------------------------------
"sym"                   :        c1       c8
-----------------------------------------------
"dim"                   :        32       64
-----------------------------------------------
"use_CC"                :         1        1
-----------------------------------------------
"refine"                :         6        6
-----------------------------------------------
"refine_factor"         :       2.0      2.0
-----------------------------------------------
"ccmatrix"              :         1        1
-----------------------------------------------
"area_search"           :       111      111
-----------------------------------------------
"area_search_modus"     :         0        0
-----------------------------------------------

-------------------------------------------------------------
Summary: long parameters

ROUND 1
"ccmatrix_type"         : [1] bin 0; sym c1;
"plugin_align_order"    : [0]
"plugin_post_order"     : [0]
"plugin_iter_order"     : [0] dynamo_plugin_xmipp -nref 1 -iter 1 -dim 32 -ang 30
-------------------------------------------------------------
ROUND 2
"ccmatrix_type"         : [1] sym c8
"plugin_align_order"    : [0]
"plugin_post_order"     : [0]
"plugin_iter_order"     : [0]
-------------------------------------------------------------
>> |
```

Now, “ccmatrix” is active in both rounds 1 and 2 ...

...BUT in different forms:

In round 1, particles will not be symmetrized when computing the ccmatrix

In round 2, particles will be symmetrized.

```
>>dynamo_vpr_info('project_for_tutorial_ccmatrix');
```

Running the project

A simple way to run the project is just executing the *execution script:*

```
 >> run project_for_tutorial_ccmatrix.m
```

or

```
>> project_for_tutorial_ccmatrix
```

If the project was to be executed in the linux shell, the execution script will be invoked as:

```
$ ./project_for_tutorial_ccmatrix.exe
```

In this case, you might need to change your permisions on this file before you actually execute it:

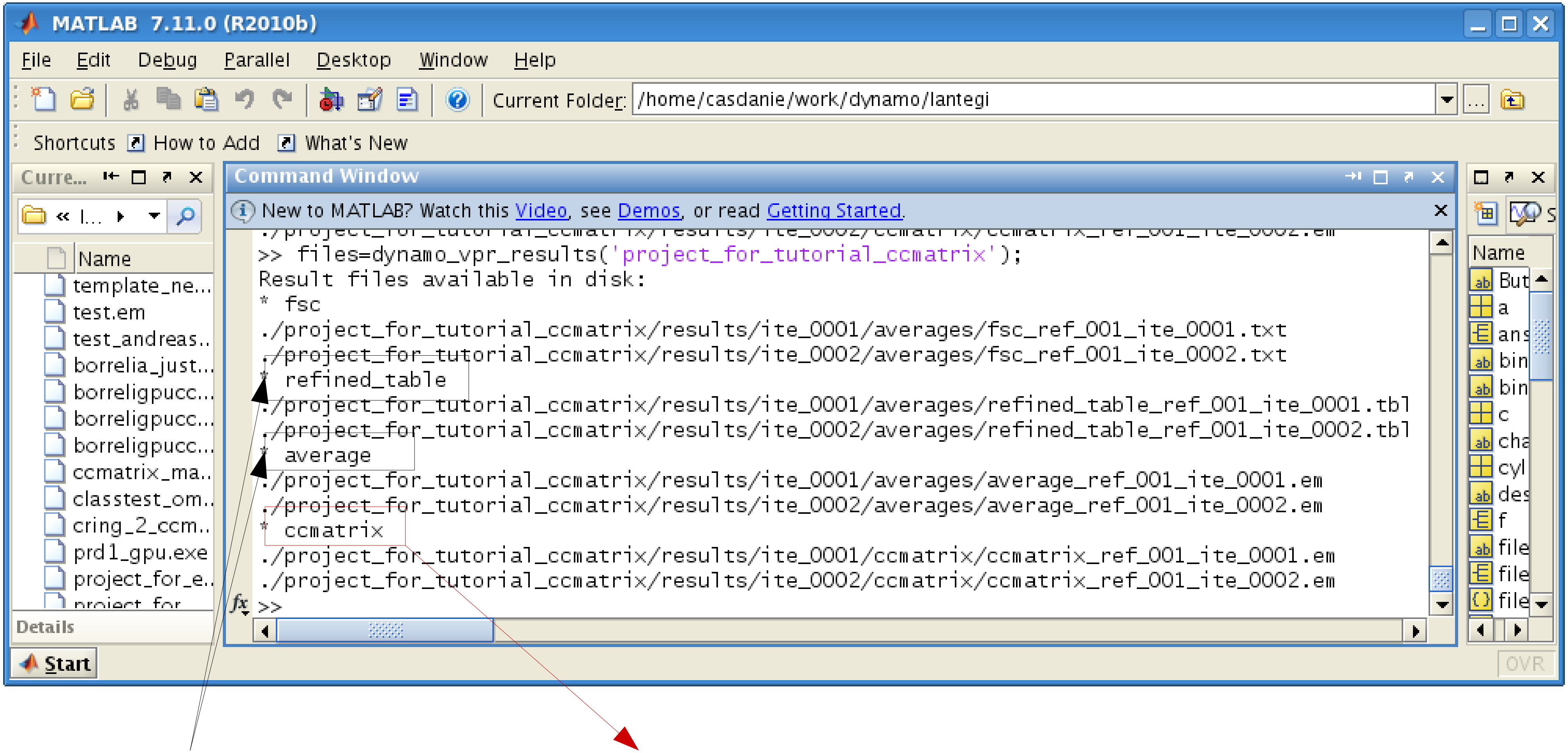
```
$ chmod u=rwx project_for_tutorial_ccmatrix.exe
```

.... NOW WAIT FOR DYNAMO TO COMPLETE THE PROJECT

Results of the project

To scan the progress of the project you can use:
`dynamo_vpr_results` (Matlab) or
`dynamo_vprf_results` (Linux/MacOS)

You should see this when the project is complete:



Current Folder: /home/casdanie/work/dynamo/lantegi

New to MATLAB? Watch this Video, see Demos, or read Getting Started.

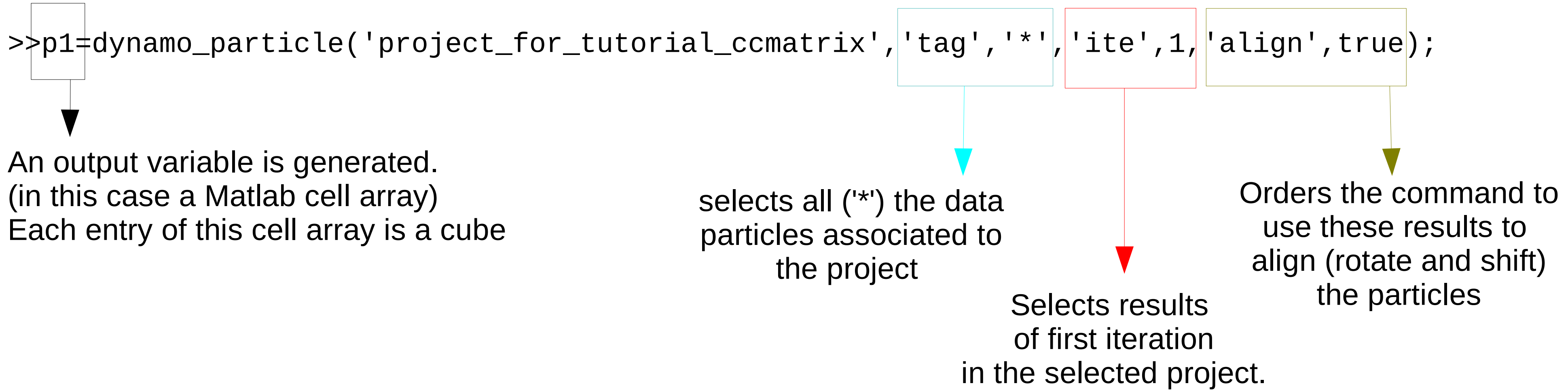
```
./project_for_tutorial_ccmatrix/results/ite_0002/ccmatrix/ccmatrix_ref_001_ite_0002.em
>> files=dynamo_vpr_results('project_for_tutorial_ccmatrix');
Result files available in disk:
* fsc
./project_for_tutorial_ccmatrix/results/ite_0001/averages/fsc_ref_001_ite_0001.txt
./project_for_tutorial_ccmatrix/results/ite_0002/averages/fsc_ref_001_ite_0002.txt
* refined_table
./project_for_tutorial_ccmatrix/results/ite_0001/averages/refined_table_ref_001_ite_0001.tbl
./project_for_tutorial_ccmatrix/results/ite_0002/averages/refined_table_ref_001_ite_0002.tbl
* average
./project_for_tutorial_ccmatrix/results/ite_0001/averages/average_ref_001_ite_0001.em
./project_for_tutorial_ccmatrix/results/ite_0002/averages/average_ref_001_ite_0002.em
* ccmatrix
./project_for_tutorial_ccmatrix/results/ite_0001/ccmatrix/ccmatrix_ref_001_ite_0001.em
./project_for_tutorial_ccmatrix/results/ite_0002/ccmatrix/ccmatrix_ref_001_ite_0002.em
>>
```

Results of the alignment

Result of the project that we need to post-process now to produce a classification.

Want to take a glance on how the aligned particles look like?

A compact way to access the particles from the command line is with “`dynamo_particle`”



```
>>p1=dynamo_particle('project_for_tutorial_ccmatrix','tag','*','ite',1,'align',true);
```

An output variable is generated.
(in this case a Matlab cell array)
Each entry of this cell array is a cube

selects all ('*') the data particles associated to the project

Selects results of first iteration in the selected project.

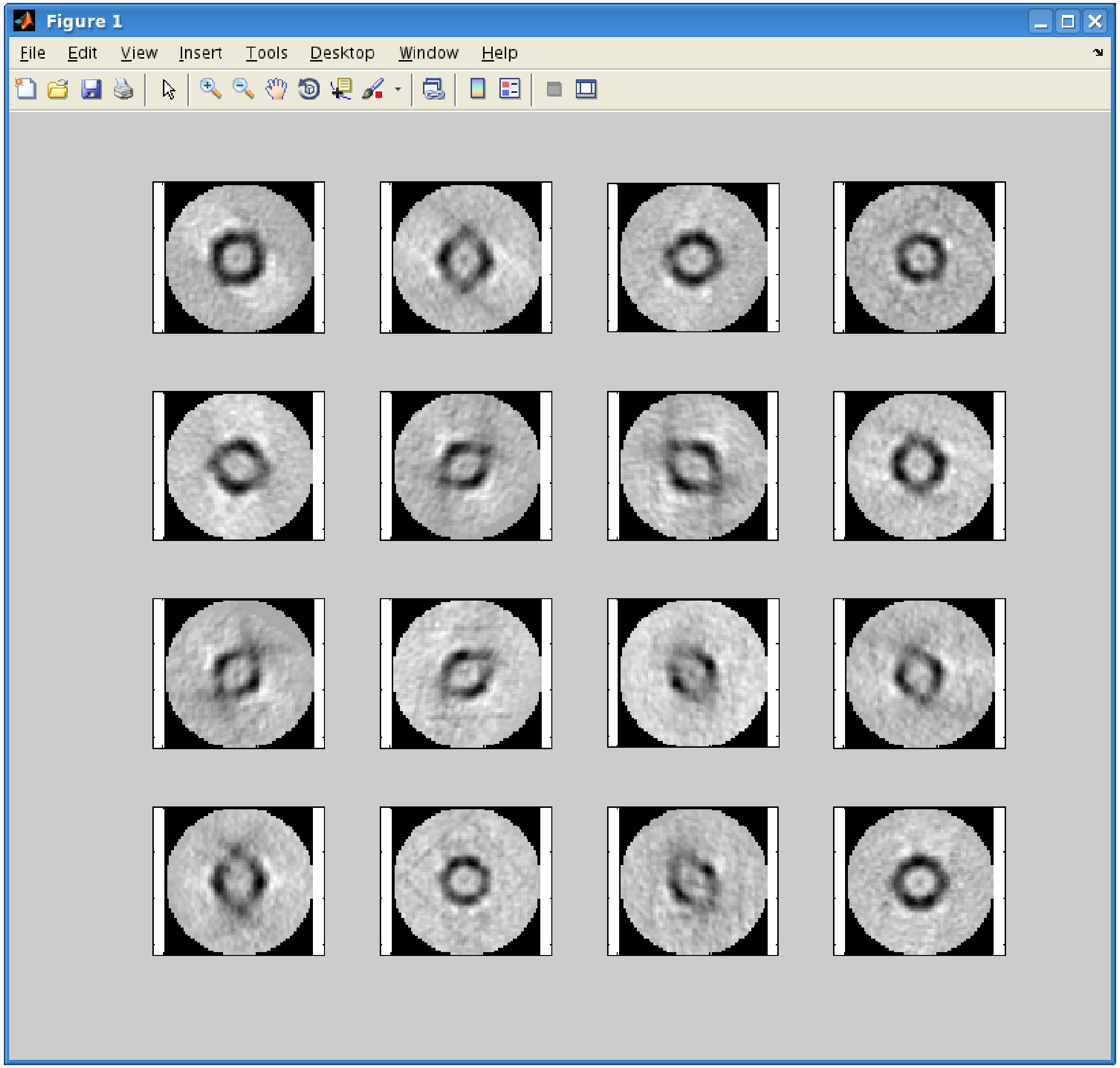
Orders the command to use these results to align (rotate and shift) the particles

Now we depict this array of particles in panel format.

```
>>dynamo_slices(p1,'range',33,'panel',1,'dim',[4,4]);
```

We depict only (z) slice #33 on each particle

You can see the different effects of the missing wedge, depending on the initial orientation of the particles...



...but do you see different classes yet?

A long intermezzo: `dynamo_gallery`

We will take now some time (and a lot of slides) to play with the data browser `dynamo_gallery`.

The main functionality of the browser is to allow the user a flexible control on which particles on a project need to be load in memory for a resonable depiction.

This is a trade off:

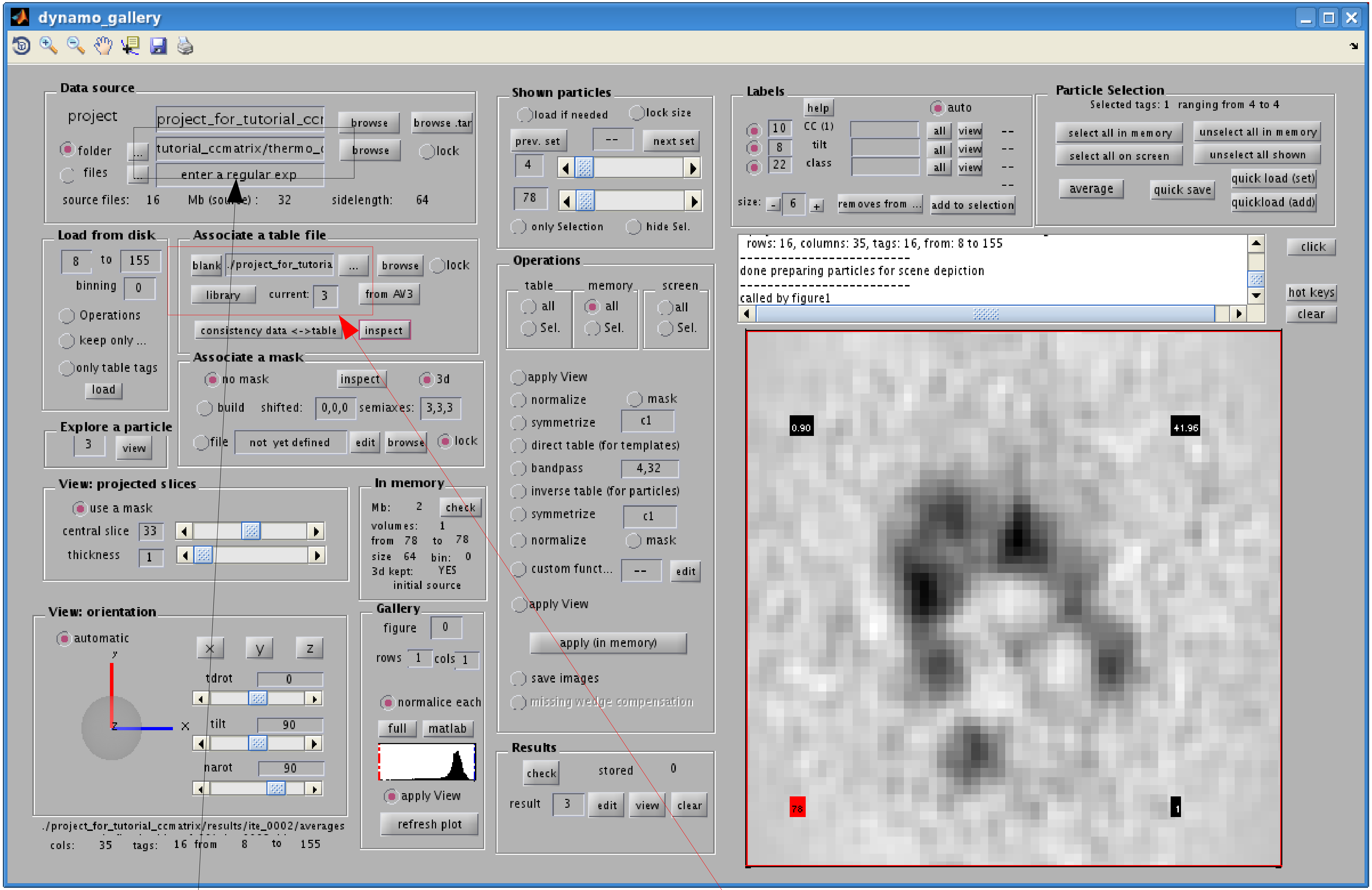
- Simultaneous loading of many or very large subtomograms can freeze your computer.
- ... but constant access to the disk can slow down the interactivity.

Invoking dynamo_gallery

The most compact way from the command line would be:

```
>>dynamo_gallery('project','project_for_tutorial_ccmatrix');
```

As freshly opened, the gallery will show you just one particle of the ones associated to the project.



dynamo_gallery

```
rows: 16, columns: 35, tags: 16, from: 8 to 155
--------------------------
done preparing particles for scene depiction
--------------------------
called by figure1
```

Data source: project: project_for_tutorial_ccr; ● folder: tutorial_ccmatrix/thermo_; ○ files: enter a regular exp; source files: 16; Mb (source): 32; sidelength: 64

Load from disk: 8 to 155; binning: 0

Associate a table file: ./project_for_tutoria; current: 3

Associate a mask: ● no mask; ● 3d; shifted: 0,0,0; semiaxes: 3,3,3; file: not yet defined

Explore a particle: 3

View: projected slices: ● use a mask; central slice: 33; thickness: 1

View: orientation: ● automatic; tdrot: 0; tilt: 90; narot: 90

./project_for_tutorial_ccmatrix/results/ite_0002/averages

cols: 35 tags: 16 from 8 to 155

Shown particles: 4; 78

Operations: memory: ● all; bandpass: 4,32; symmetrize: c1

In memory: Mb: 2; volumes: 1; from 78 to 78; size 64 bin: 0; 3d kept: YES; initial source

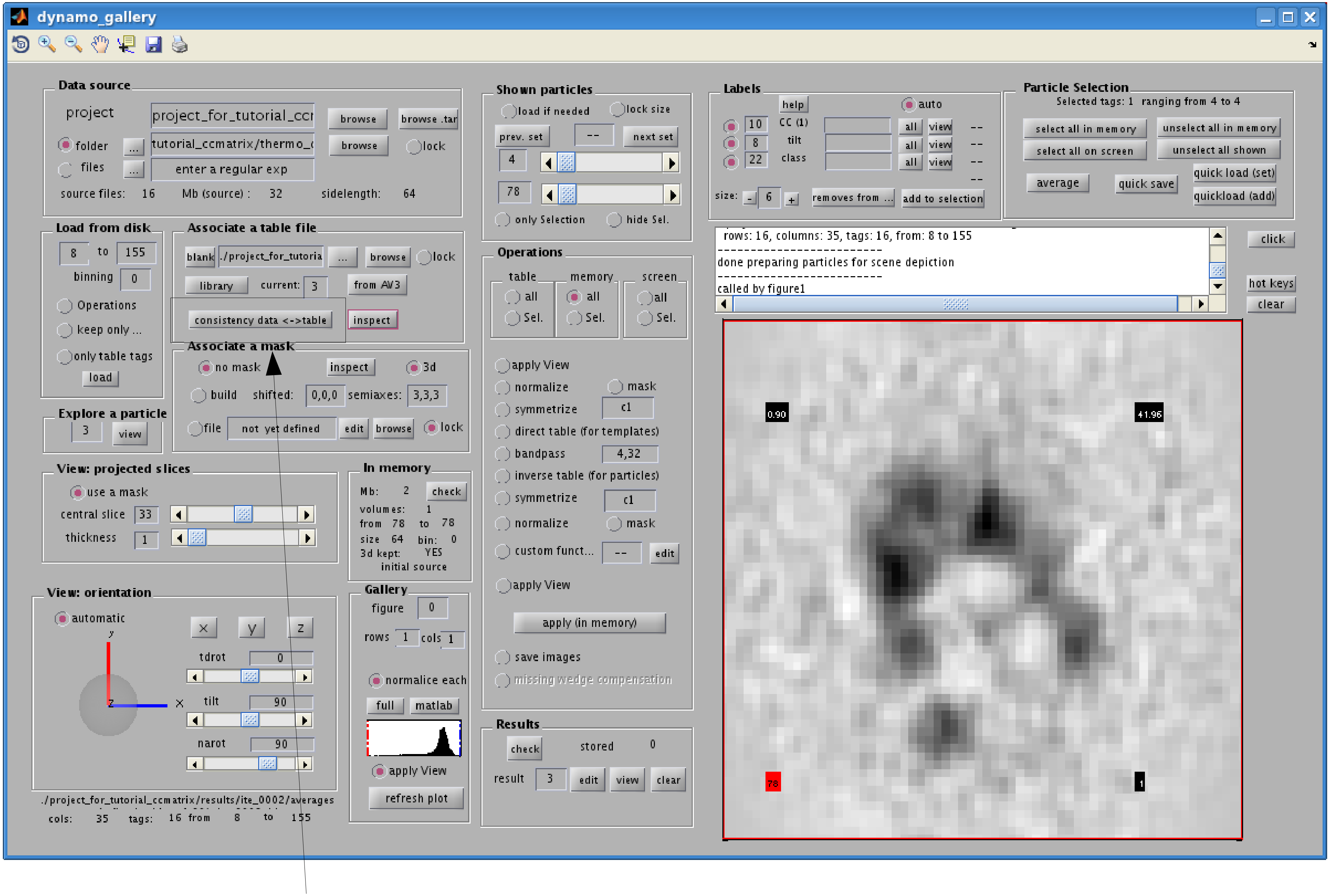
Gallery: figure: 0; rows: 1; cols: 1; ● normalice each; ● apply View

Results: stored: 0; result: 3

Labels: ● 10 CC (1); ● 8 tilt; ● 22 class; ● auto; size: 6

Particle Selection: Selected tags: 1 ranging from 4 to 4

The data location is pointing in the right direction, and table files are already loaded in the internal workspace



```
rows: 16, columns: 35, tags: 16, from: 8 to 155
--------------------------
done preparing particles for scene depiction
--------------------------
called by figure1
```

./project_for_tutorial_ccmatrix/results/ite_0002/averages

cols: 35 tags: 16 from 8 to 155

A good start is to press here: it shows the tags of particles in disk, in memory and in the table, and warns on possible mismatchings

The popup window should
look like this:

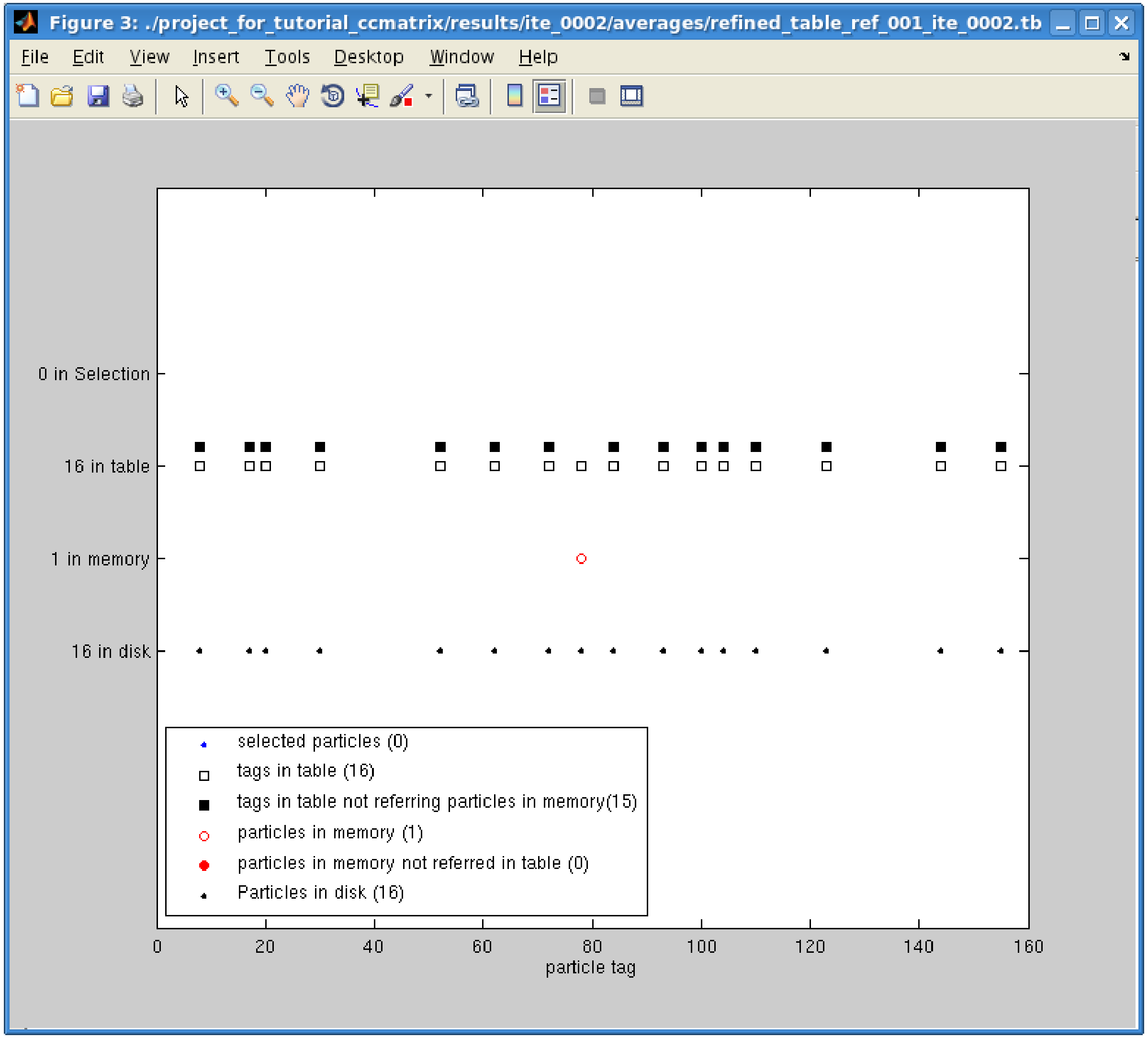


Figure 3: ./project_for_tutorial_ccmatrix/results/ite_0002/averages/refined_table_ref_001_ite_0002.tb

In this project we don't have
a massive number of particles,
We can thus load all of them
simultaneously into memory.

Loading particles:

1) *Switch on (Operations) radiobutton*

This will apply on the each particle whatever series of Operation is defined in the [Operations] Panel (in the right).

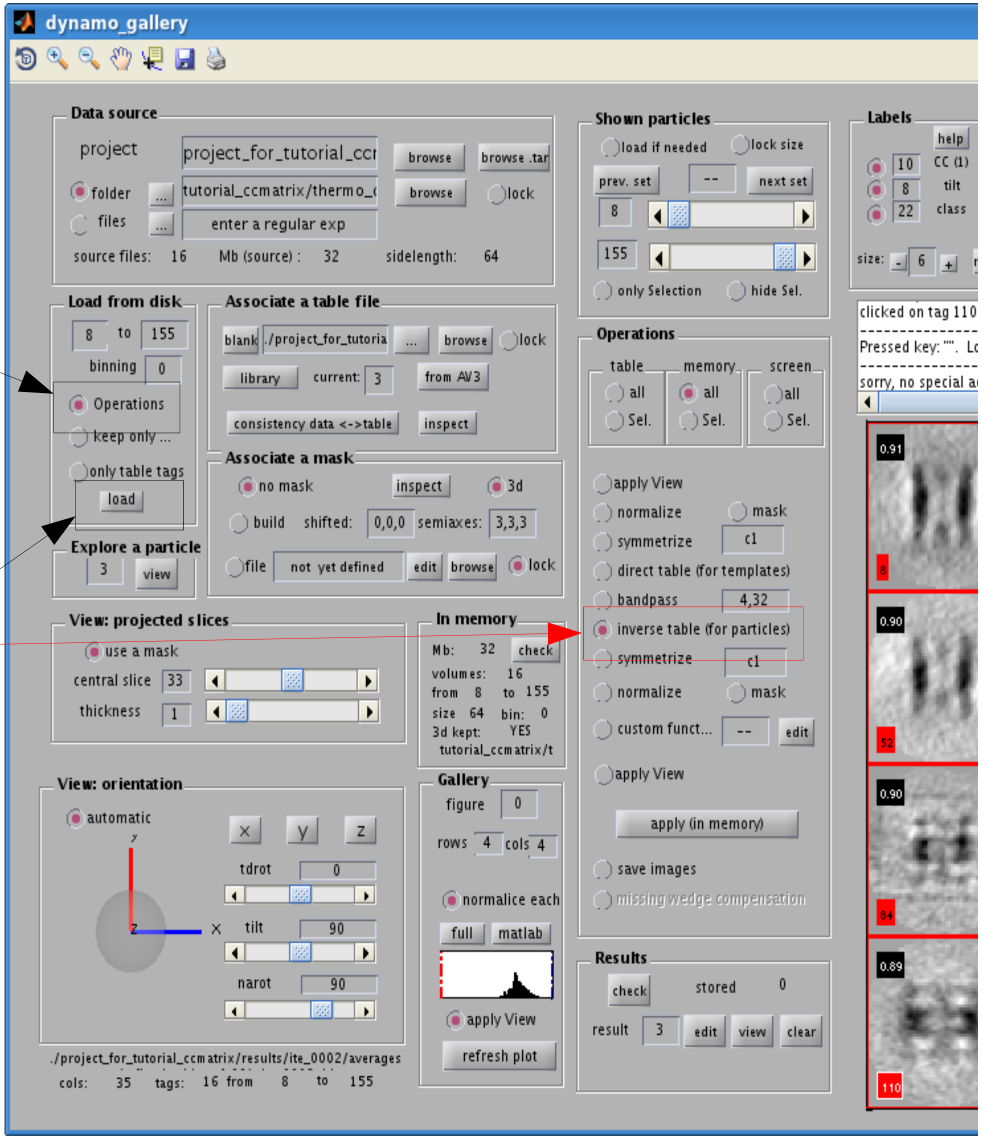
Each raw particle will loaded, transformed and deposited in memory

2) *Switch on (Inverse Table for particles)*

This indicate that you want to align the particles (using the active table)

3) [*Load*] *into memory*

In this case, it will take seconds.
For large data sets it can be quite time consuming.
Also, you might want to use the "bin" option before you load the particles to save memory



dynamo_gallery

Data source
project: project_for_tutorial_ccr
● folder: tutorial_ccmatrix/thermo_(
○ files: enter a regular exp
source files: 16 Mb (source): 32 sidelength: 64

Load from disk
8 to 155
binning: 0
● Operations
○ keep only ...
○ only table tags

Explore a particle: 3

Associate a table file
blank: ./project_for_tutoria
current: 3

Associate a mask
● no mask ● 3d
○ build shifted: 0,0,0 semiaxes: 3,3,3
○ file: not yet defined ● lock

View: projected slices
● use a mask
central slice: 33
thickness: 1

In memory
Mb: 32
volumes: 16
from 8 to 155
size 64 bin: 0
3d kept: YES
tutorial_ccmatrix/t

View: orientation
● automatic
tdrot: 0
tilt: 90
narot: 90

./project_for_tutorial_ccmatrix/results/ite_0002/averages
cols: 35 tags: 16 from 8 to 155

Gallery
figure: 0
rows: 4 cols: 4
● normalice each
● apply View

Shown particles
○ load if needed ○ lock size
--
8
155
○ only Selection ○ hide Sel.

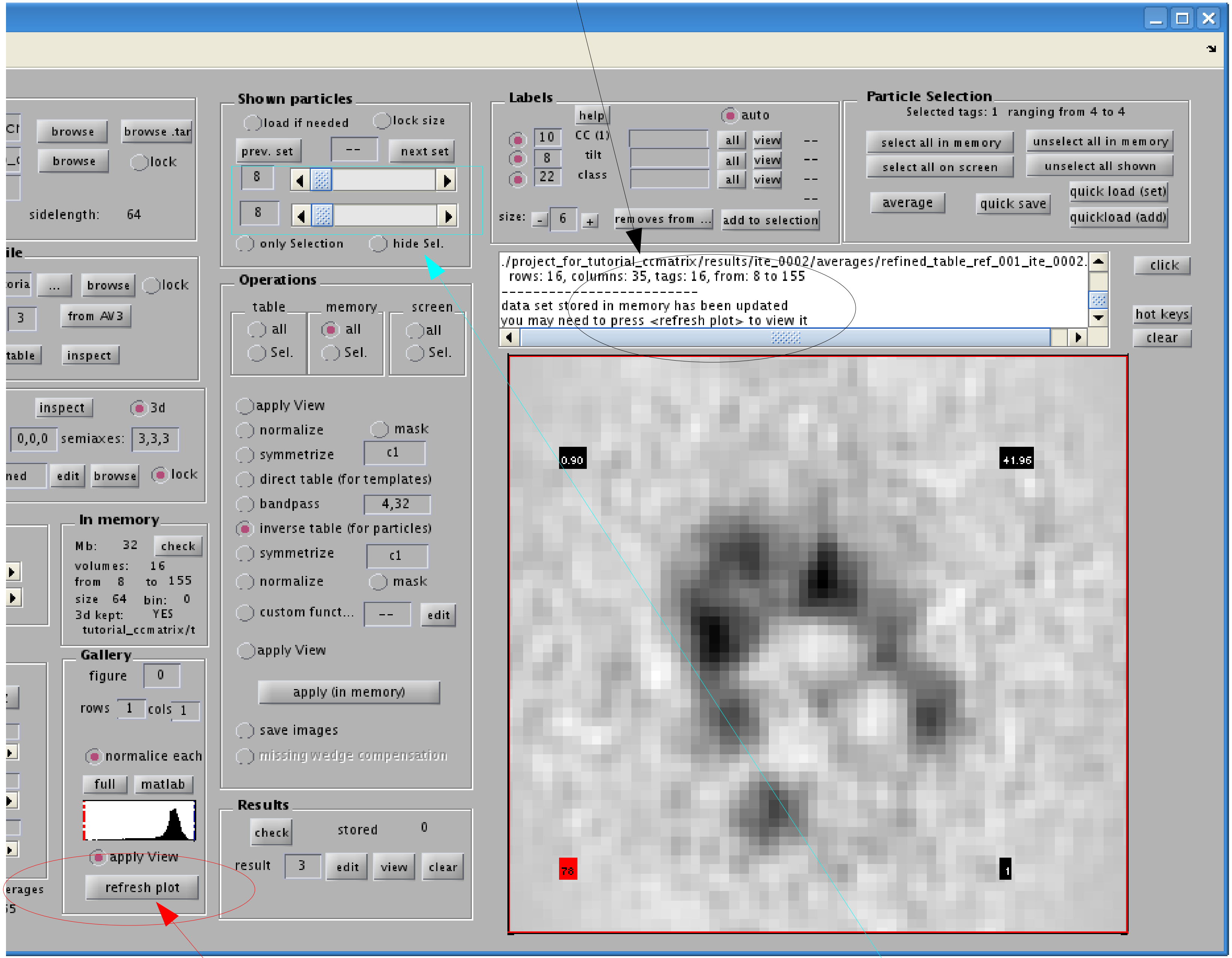
Operations
table: ○ all ○ Sel.
memory: ● all ○ Sel.
screen: ○ all ○ Sel.
○ apply View
○ normalize ○ mask
○ symmetrize: c1
○ direct table (for templates)
○ bandpass: 4,32
● inverse table (for particles)
○ symmetrize: c1
○ normalize ○ mask
○ custom funct...: --
○ apply View
○ save images
○ missing wedge compensation

Results
stored: 0
result: 3

Labels
● 10 CC (1)
● 8 tilt
● 22 class
size: 6

clicked on tag 110
Pressed key: "". Lo
sorry, no special a

... wait until the Information Area anounces that the particles are already there...

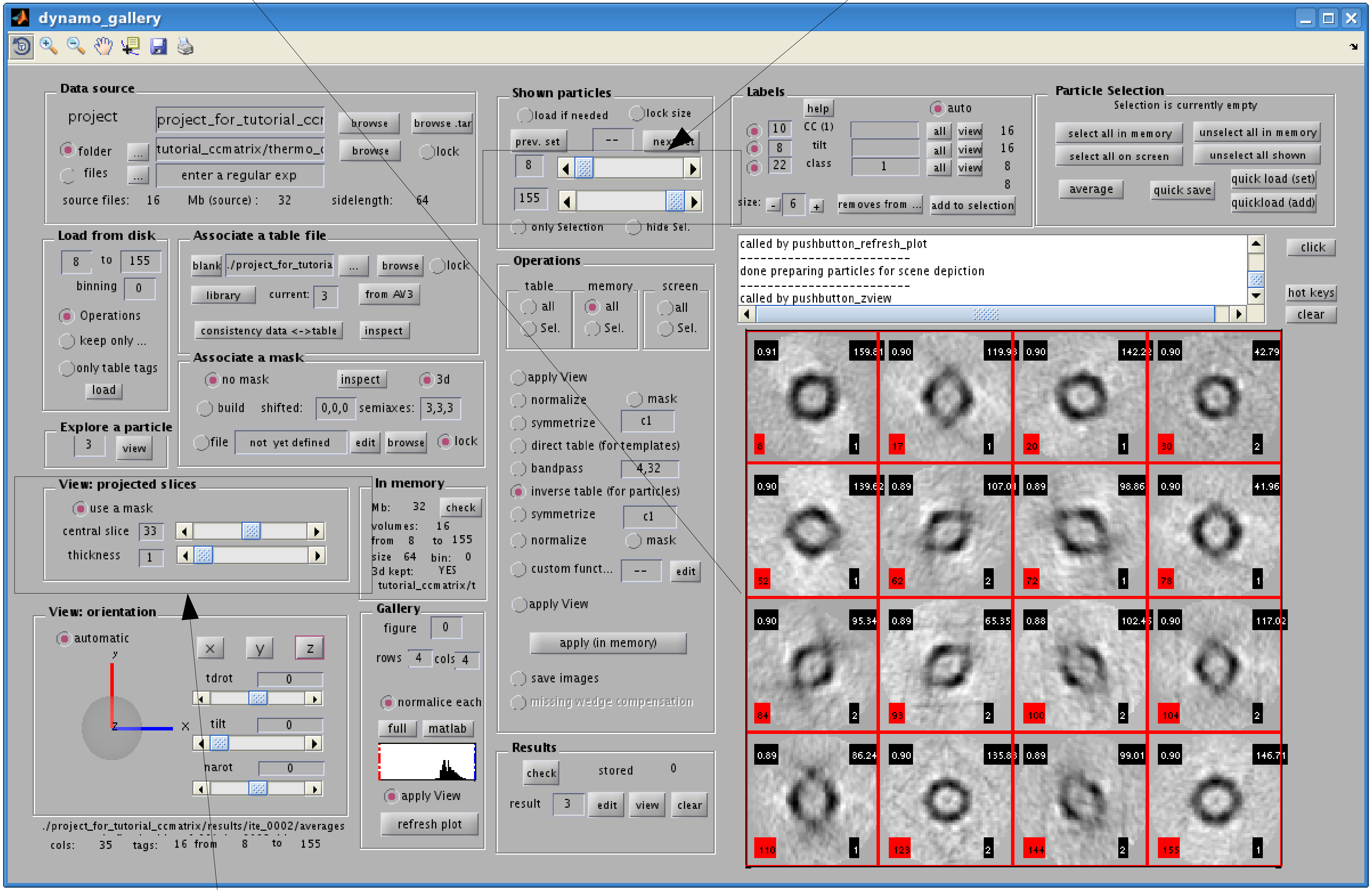


```
./project_for_tutorial_ccmatrix/results/ite_0002/averages/refined_table_ref_001_ite_0002.
 rows: 16, columns: 35, tags: 16, from: 8 to 155
 --------------------------
data set stored in memory has been updated
you may need to press <refresh plot> to view it
```

... and then refresh the plot...

... and select a range of particles to view...

The scene should now show all particles -in memory- inside this range



project: project_for_tutorial_ccr
folder: tutorial_ccmatrix/thermo_c
files: enter a regular exp
source files: 16 Mb (source): 32 sidelength: 64

Load from disk: 8 to 155, binning 0

Associate a table file: ./project_for_tutoria current: 3

Associate a mask: shifted: 0,0,0 semiaxes: 3,3,3 file: not yet defined

Explore a particle: 3

View: projected slices: central slice 33, thickness 1

View: orientation: tdrot 0, tilt 0, narot 0

./project_for_tutorial_ccmatrix/results/ite_0002/averages
cols: 35 tags: 16 from 8 to 155

In memory: Mb: 32, volumes: 16, from 8 to 155, size 64, bin: 0, 3d kept: YES, tutorial_ccmatrix/t

Gallery: figure 0, rows 4, cols 4

Shown particles: 8, 155

Operations: symmetrize c1, bandpass 4,32, symmetrize c1, custom funct... --

Results: stored 0, result 3

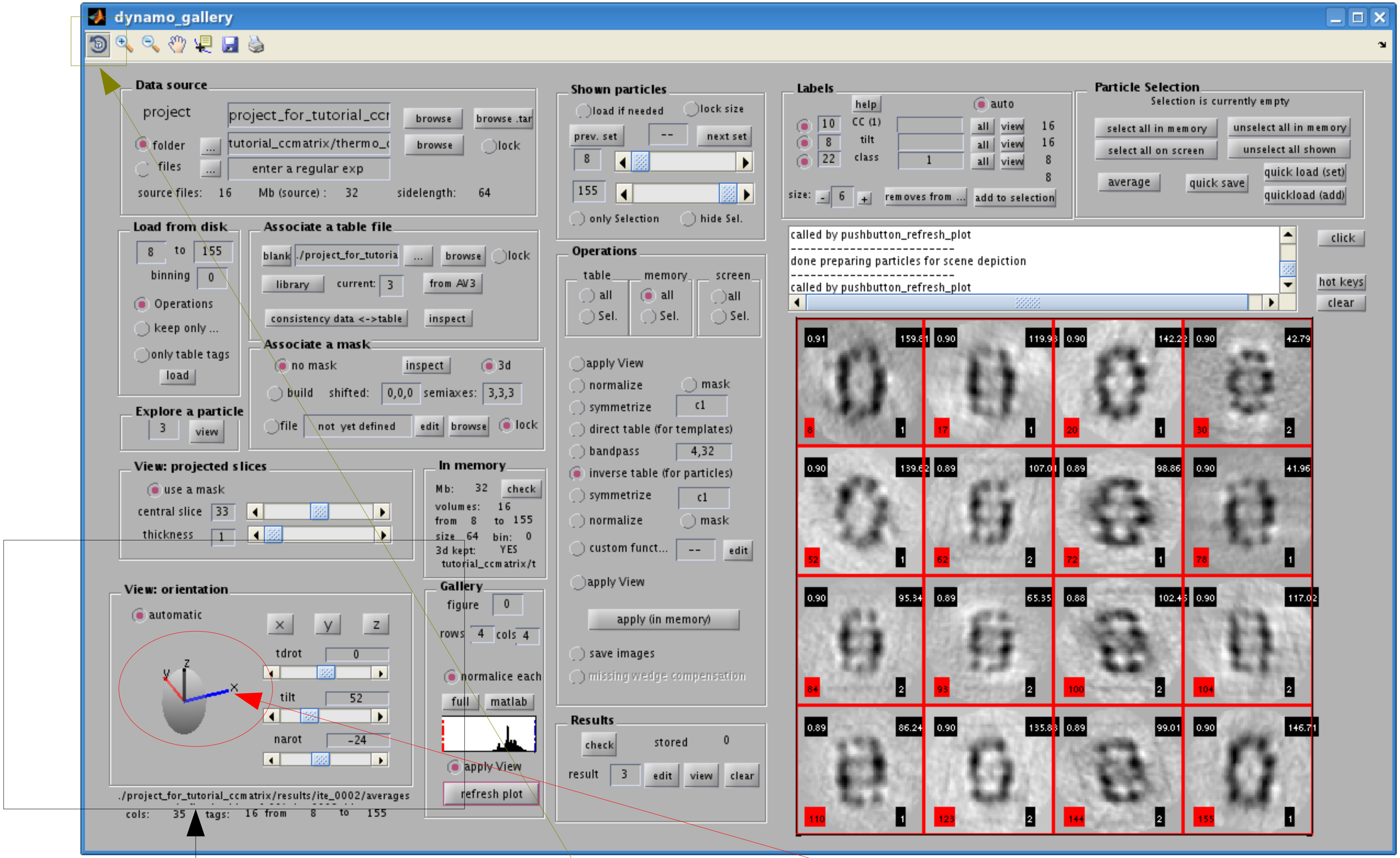
Labels: 10 CC (1) 16; 8 tilt 16; 22 class 1 8; 8; size: 6

```
called by pushbutton_refresh_plot
--------------------------
done preparing particles for scene depiction
--------------------------
called by pushbutton_zview
```

0.91	159.81	0.90	119.93	0.90	142.22	0.90	42.79
8	1	17	1	20	1	30	2
0.90	139.62	0.89	107.01	0.89	98.86	0.90	41.96
52	1	62	2	72	1	78	1
0.90	95.34	0.89	65.35	0.88	102.45	0.90	117.02
84	2	93	2	100	2	104	2
0.89	86.24	0.90	135.83	0.89	99.01	0.90	146.71
110	1	125	2	144	2	155	1

Here you indicate how many slices (around a central one) are projected to represent each particle.
In this case we represent one single slice (no projections) in the center of each each particle (they are of size 64)

Rotating the particles in the scene



Data source: project: project_for_tutorial_ccr; ● folder: tutorial_ccmatrix/thermo_c; ○ files: enter a regular exp

source files: 16 Mb (source) : 32 sidelength: 64

Load from disk: 8 to 155; binning 0; ● Operations; ○ keep only ...; ○ only table tags; load

Associate a table file: blank ./project_for_tutoria ... browse ○ lock; current: 3

Associate a mask: ● no mask; inspect; ● 3d; ○ build shifted: 0,0,0 semiaxes: 3,3,3; ○ file not yet defined edit browse ● lock

Explore a particle: 3 view

View: projected slices: ● use a mask; central slice 33; thickness 1

View: orientation: ● automatic; tdrot 0; tilt 52; narot -24

./project_for_tutorial_ccmatrix/results/ite_0002/averages

cols: 35 tags: 16 from 8 to 155

In memory: Mb: 32; volumes: 16; from 8 to 155; size 64 bin: 0; 3d kept: YES; tutorial_ccmatrix/t

Gallery: figure 0; rows 4 cols 4; ● normalice each; ● apply View

Shown particles: ○ load if needed ○ lock size; --; 8; 155; ○ only Selection ○ hide Sel.

Operations: table ○ all ○ Sel.; memory ● all ○ Sel.; screen ○ all ○ Sel.

○ apply View
○ normalize ○ mask
○ symmetrize c1
○ direct table (for templates)
○ bandpass 4,32
● inverse table (for particles)
○ symmetrize c1
○ normalize ○ mask
○ custom funct... -- edit
○ apply View
○ save images
○ missing wedge compensation

Results: stored 0; result 3

Labels: ● auto; ● 10 CC (1) 16; ● 8 tilt 16; ● 22 class 1 8; 8; size: - 6 +

Particle Selection: Selection is currently empty

```
called by pushbutton_refresh_plot
-----------------------
done preparing particles for scene depiction
-----------------------
called by pushbutton_refresh_plot
```

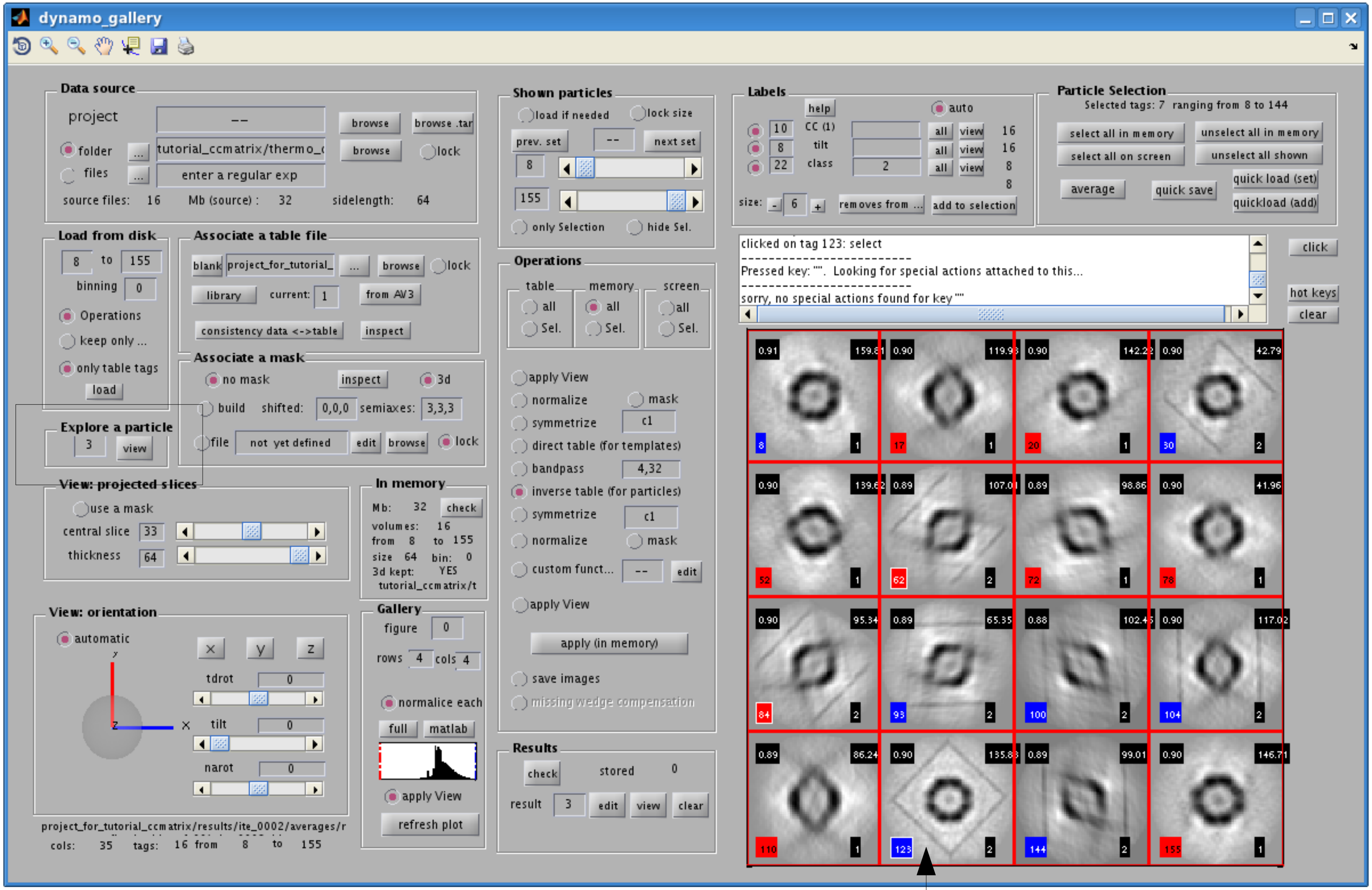
0.91 159.81 8 1	0.90 119.93 17 1	0.90 142.22 20 1	0.90 42.79 30 2
0.90 139.62 52 1	0.89 107.01 62 2	0.89 98.86 72 1	0.90 41.96 78 1
0.90 95.34 84 2	0.89 65.35 93 2	0.88 102.45 100 2	0.90 117.02 104 2
0.89 86.24 110 1	0.90 135.83 123 2	0.89 99.01 144 2	0.90 146.71 155 1

You can use this Panel to select viewing orientations:
* Press [x], [y] or [z]
* Type or slide for the angles, then click [Refresh plot]

The depicted scene will change accordingly

You can also drag the “reference phantom”
To select the viewing orientation
(the rotation icon in the corner must be active)

Choosing a single particle to inspect in detail



clicked on tag 123: select

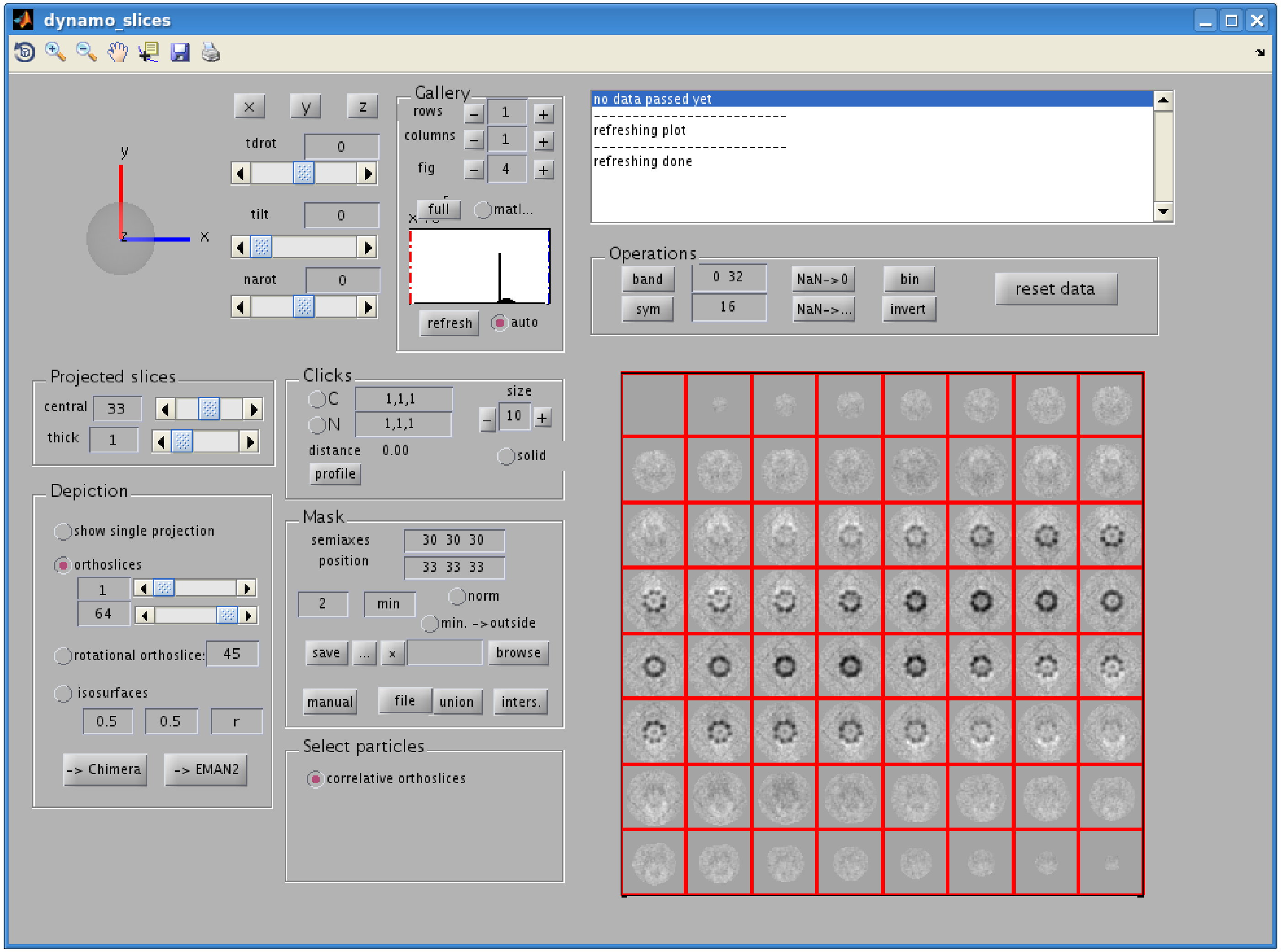
Pressed key: "". Looking for special actions attached to this...
----
sorry, no special actions found for key ""

You can just pass the tag number of the particle (for particles not in the current scene)

... or simply press the key “V” on the particle, if it is visible

For instance here:

You open the familiar `dynamo_mapview` on the particle as seen in `dynamo_gallery` (in this case, aligned)



```
no data passed yet
---------------------------
refreshing plot
---------------------------
refreshing done
```

tdrot 0

tilt 0

narot 0

rows 1

columns 1

fig 4

● auto

band 0 32

sym 16

central 33

thick 1

C 1,1,1

N 1,1,1

size 10

distance 0.00

● orthoslices 1 64

○ rotational orthoslice: 45

isosurfaces 0.5 0.5 r

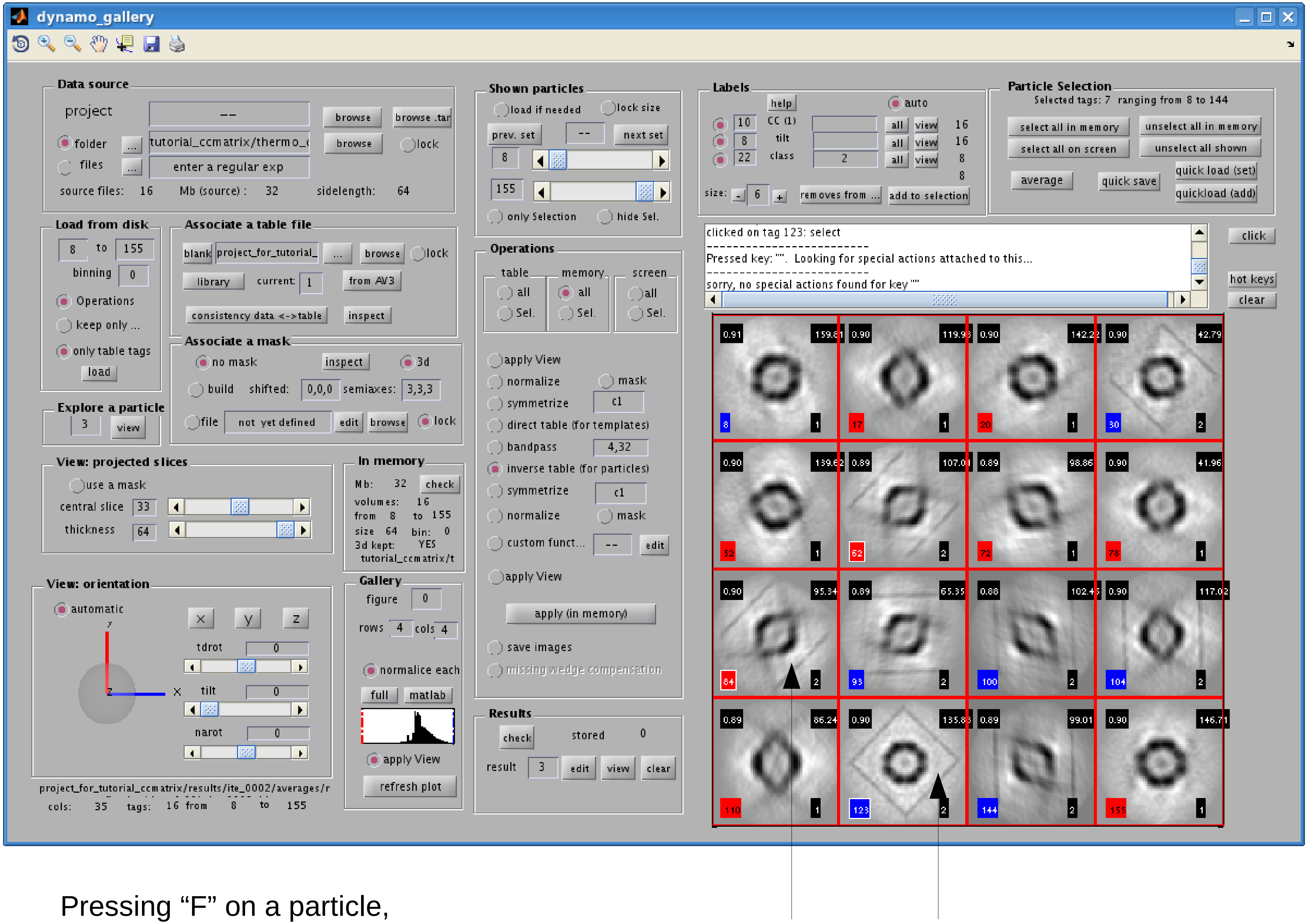
semiaxes 30 30 30

position 33 33 33

2 min

● correlative orthoslices

The particle is in the memory space of mapview, where it can be edited and displayed with the provided tools, (masking, bandpassing, symmetrization, range selection...) etc,
or delegated to other viewers (EMAN2,Chimera...) than can render efficiently the isosurfaces of the processed volume.



Data source: project --
● folder tutorial_ccmatrix/thermo_(
○ files enter a regular exp
○ lock
source files: 16 Mb (source): 32 sidelength: 64

Load from disk: 8 to 155, binning 0
● Operations
○ keep only ...
● only table tags
load

Associate a table file: blank project_for_tutorial_ ... ○ lock
current: 1 from AV3

Associate a mask: ● no mask ● 3d
○ build shifted: 0,0,0 semiaxes: 3,3,3
○ file not yet defined ● lock

Explore a particle: 3

View: projected slices
○ use a mask
central slice 33
thickness 64

In memory
Mb: 32
volumes: 16
from 8 to 155
size 64 bin: 0
3d kept: YES
tutorial_ccmatrix/t

View: orientation
● automatic
tdrot 0
tilt 0
narot 0

project_for_tutorial_ccmatrix/results/ite_0002/averages/r
cols: 35 tags: 16 from 8 to 155

Gallery
figure 0
rows 4 cols 4
● normalice each
● apply View

Shown particles
○ load if needed ○ lock size
--
8
155
○ only Selection ○ hide Sel.

Operations
table: ○ all ○ Sel.
memory: ● all ○ Sel.
screen: ○ all ○ Sel.
○ apply View
○ normalize ○ mask
○ symmetrize c1
○ direct table (for templates)
○ bandpass 4,32
● inverse table (for particles)
○ symmetrize c1
○ normalize ○ mask
○ custom funct... --
○ apply View
○ save images
○ missing wedge compensation

Results
stored 0
result 3

Labels
● auto
● 10 CC (1) all 16
● 8 tilt all 16
● 22 class 2 all 8
8
size: - 6 +

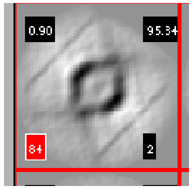
Particle Selection
Selected tags: 7 ranging from 8 to 144

```
clicked on tag 123: select
---------------------------
Pressed key: "".  Looking for special actions attached to this...
---------------------------
sorry, no special actions found for key ""
```

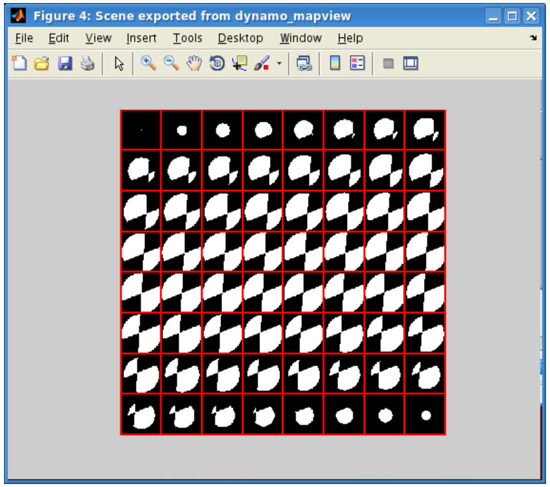
Pressing “F” on a particle,
you see the **missing wedge** on that particle
(with the corresponding geometrical transformation)

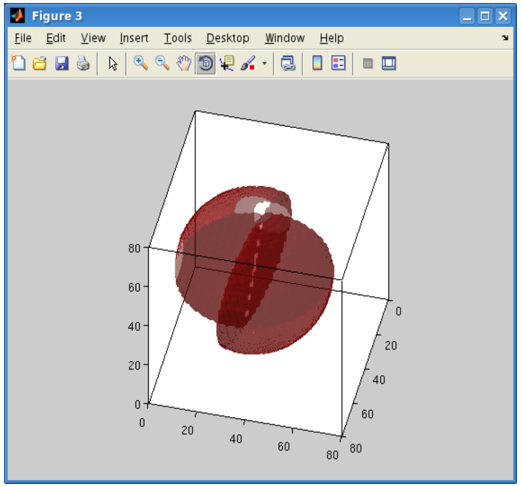
Compare these two: (next slide)

Viewing the missing wedge of particles in `dynamo_mapview`



Press [[F]]

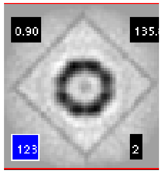




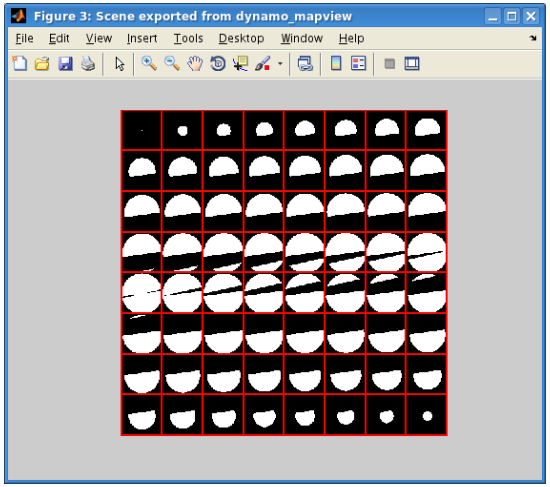
Projection along z
(as shown in the gallery)

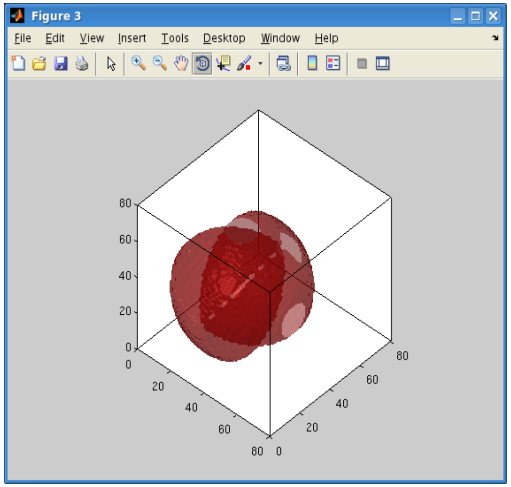
`dynamo_mapview` shows the
present fourier components
as series of slices (in this case from z)...

...or isosurfaces

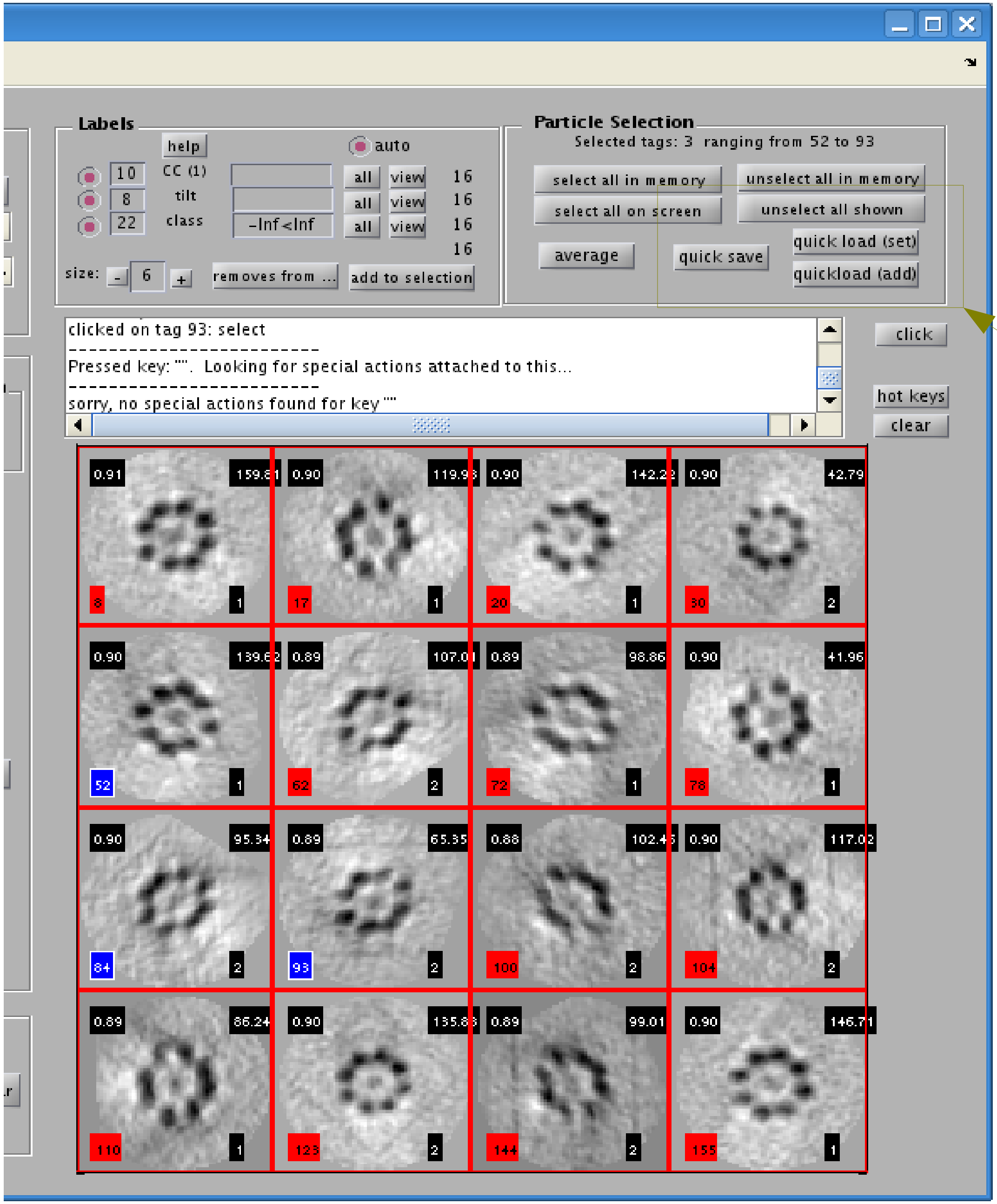


Press [[F]]





[Particle Selection] Panel



Selected tags: 3 ranging from 52 to 93

```
clicked on tag 93: select
------------------------
Pressed key: "". Looking for special actions attached to this...
------------------------
sorry, no special actions found for key ""
```

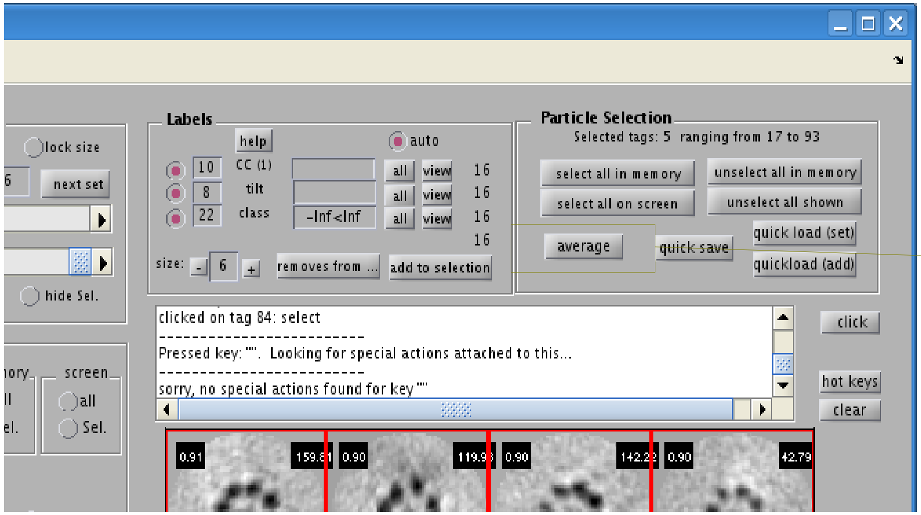
You can select particles left-clicking on them (and right-clicking to deselect).

Groups of particles can be selected drawing a selection box
(start with middle-click, quit with “Q”).

You can import/export a Selection using the file `quickbuffer.tags`
This text file contains just a column of integers (the selected tag numbers)

It is customarily used to pass Selection sets across other applications in Dynamo
(or to and from the command line)
in a quick way.

[Particle Selection] Panel
Averaging sets of particles



Selected tags: 5 ranging from 17 to 93

```
clicked on tag 84: select
-------------------------
Pressed key: "".  Looking for special actions attached to this...
-------------------------
sorry, no special actions found for key ""
```

This pushbutton opens a dialog box
Where you can detail numerical settings
To compute and average of the selected
particles.

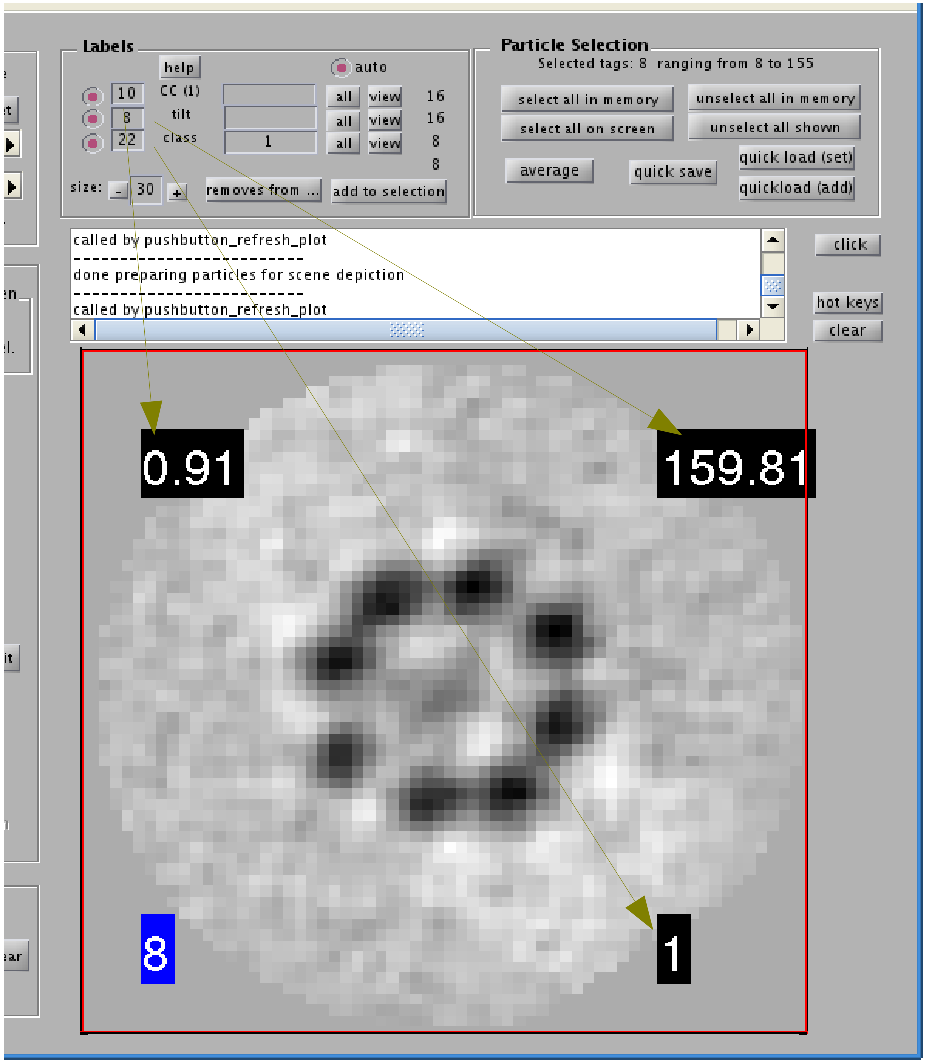
Particle annotation: [Labels]

At each corner of the particle you can show a table entry for the corresponding particle.

The image shows the initial settings, showing columns 10, 8 and 22 of the table
(in clockwise order from the up left corner).

You can choose any other column in the table
(type it in the input field),
or switch the label off.

This corner shows always the particle “tag” number
(shown in blue if the particle has been selected
 in red otherwise)



Labels

● 10 CC (1)
● 8 tilt
● 22 class 1

size: 30

Particle Selection
Selected tags: 8 ranging from 8 to 155

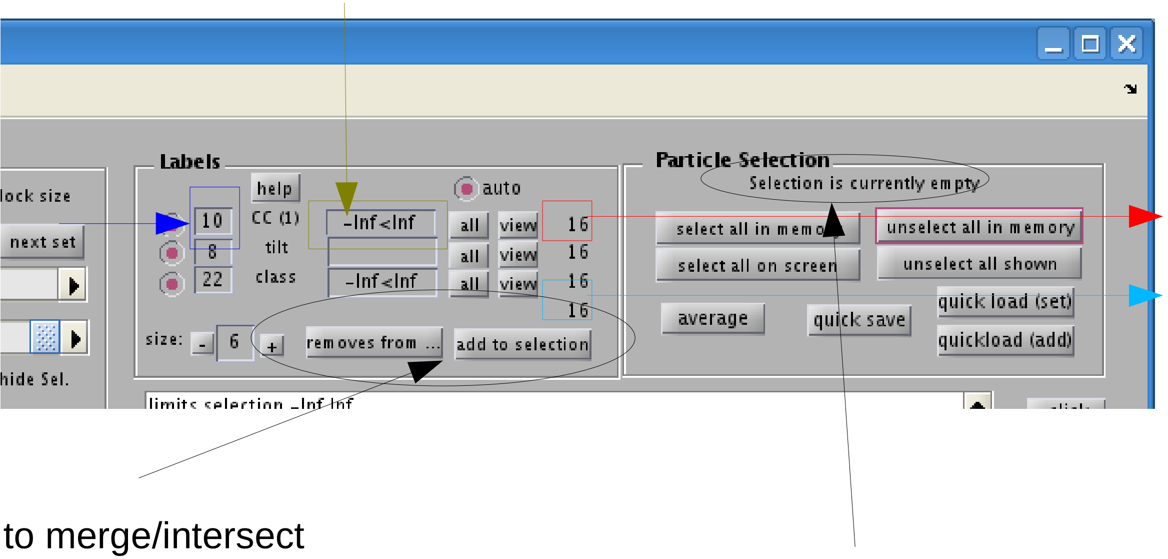
```
called by pushbutton_refresh_plot
--------------------------
done preparing particles for scene depiction
--------------------------
called by pushbutton_refresh_plot
```

Selection of particles according to table properties

It can be useful to group particles according to their statistical properties.
When analyzing your data, it is not uncommon to get into thoughts as the following:
 “Hm... I'd like to examine the 30% particles with best cross correlation coefficient,
   but only those whose orientation is close to the beam direction, and
   also those that belong to a certain label that I've put previously in the table”.

Command line tools as `dynamo_table_restrict` or the graphical table manager `dynamo_tableview` provide extensive support for such functionalities, but `dynamo_gallery` can deliver simultaneous visual support on the Selection.

Use this field to select an interval of values (on the selected table column)



Selection is currently empty

limits selection -Inf Inf

Select here a table column
to use as restriction criterion

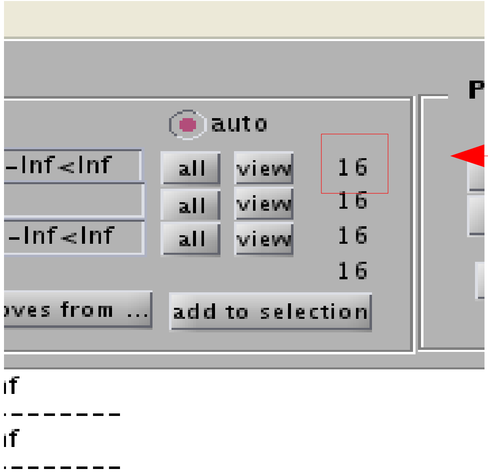
Number of particles
that meet this restriction

Number of particles
that meet *all* restrictions

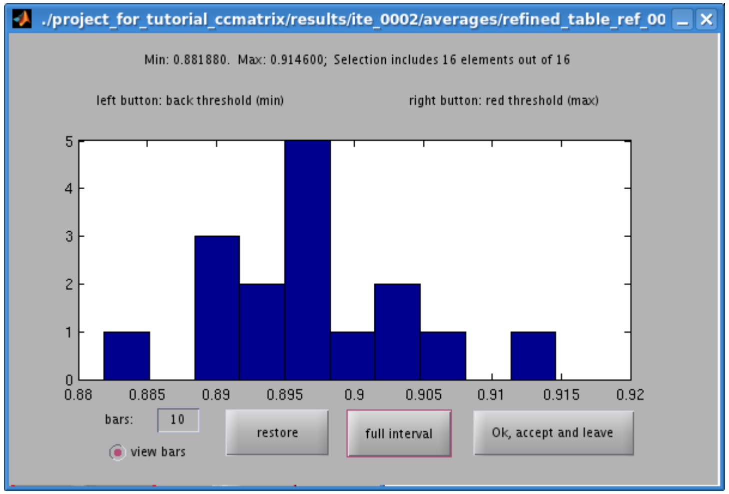
Press these buttons to merge/intersect
the “selection by property” to the Selection set.

You can inspect here the current contents of the Selection
(and, obviously, you can check the blue/red labels in the scene)

Selection of particles according to table properties



Pushbutton [view] opens a histogram for the values of the selected property in all the tags in the table

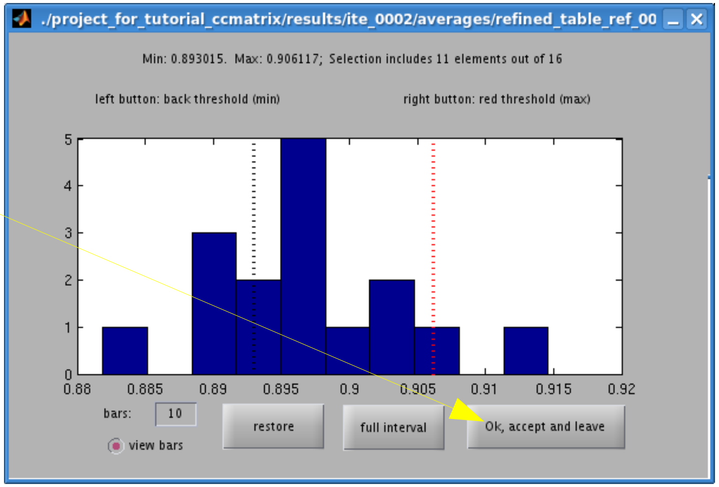

./project_for_tutorial_ccmatrix/results/ite_0002/averages/refined_table_ref_00

Min: 0.881880. Max: 0.914600; Selection includes 16 elements out of 16

left button: back threshold (min)

right button: red threshold (max)

You can mark an interval of values (left and right clicks)

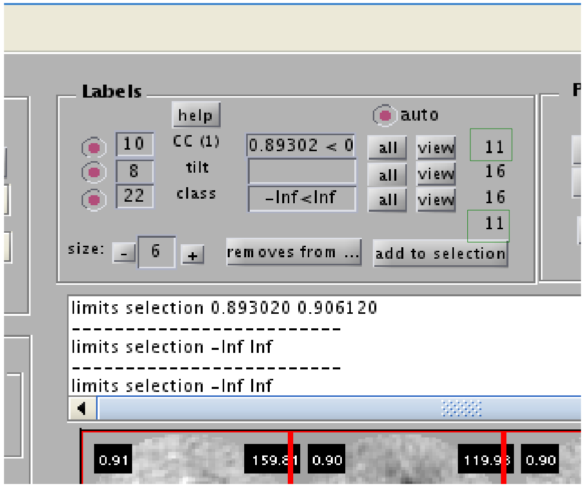

./project_for_tutorial_ccmatrix/results/ite_0002/averages/refined_table_ref_00

Min: 0.893015. Max: 0.906117; Selection includes 11 elements out of 16

left button: back threshold (min)

right button: red threshold (max)

And when you leave the histogram, it updates the set “selected by property” in `dynamo_gallery`



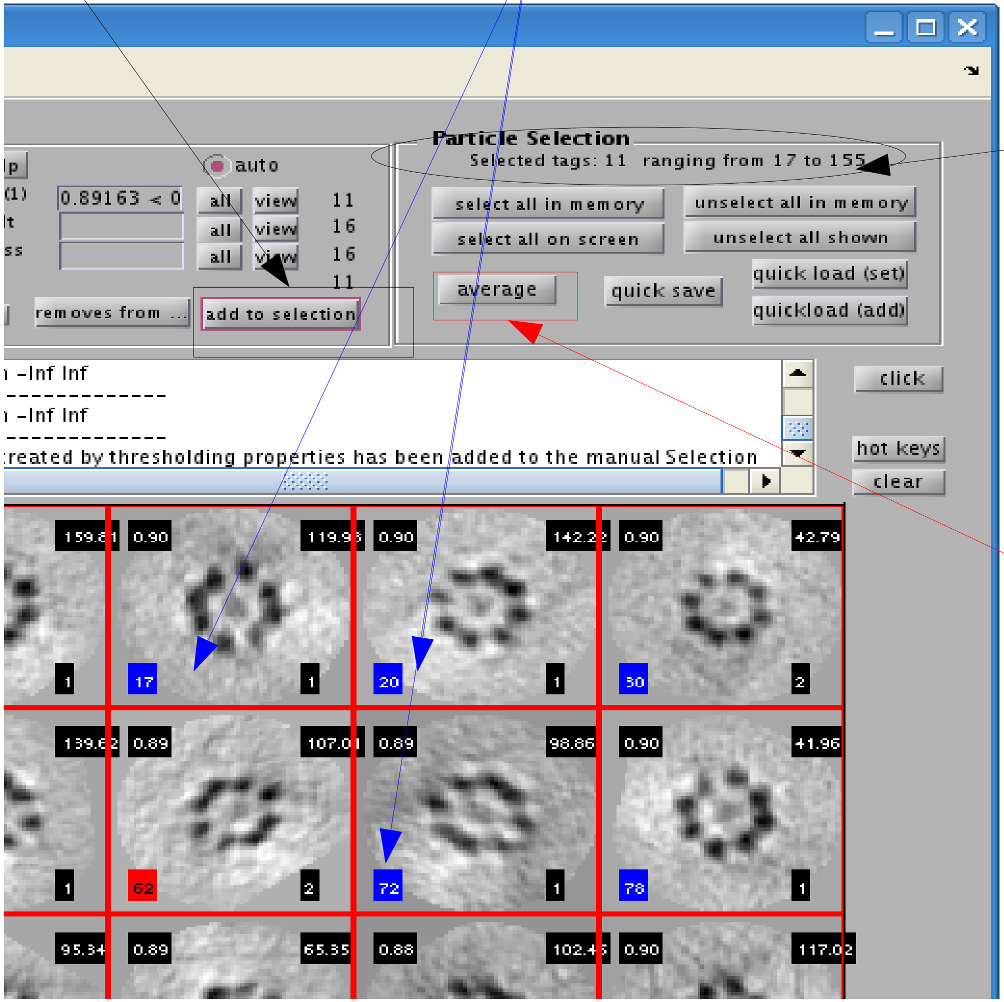
```
limits selection 0.893020 0.906120
-------------------------
limits selection -Inf Inf
-------------------------
limits selection -Inf Inf
```

... we can thus make an average of the particles “selected by property”....

Add these particles to the general selection....

... the Selection labels get updated....

... and also the Selection report

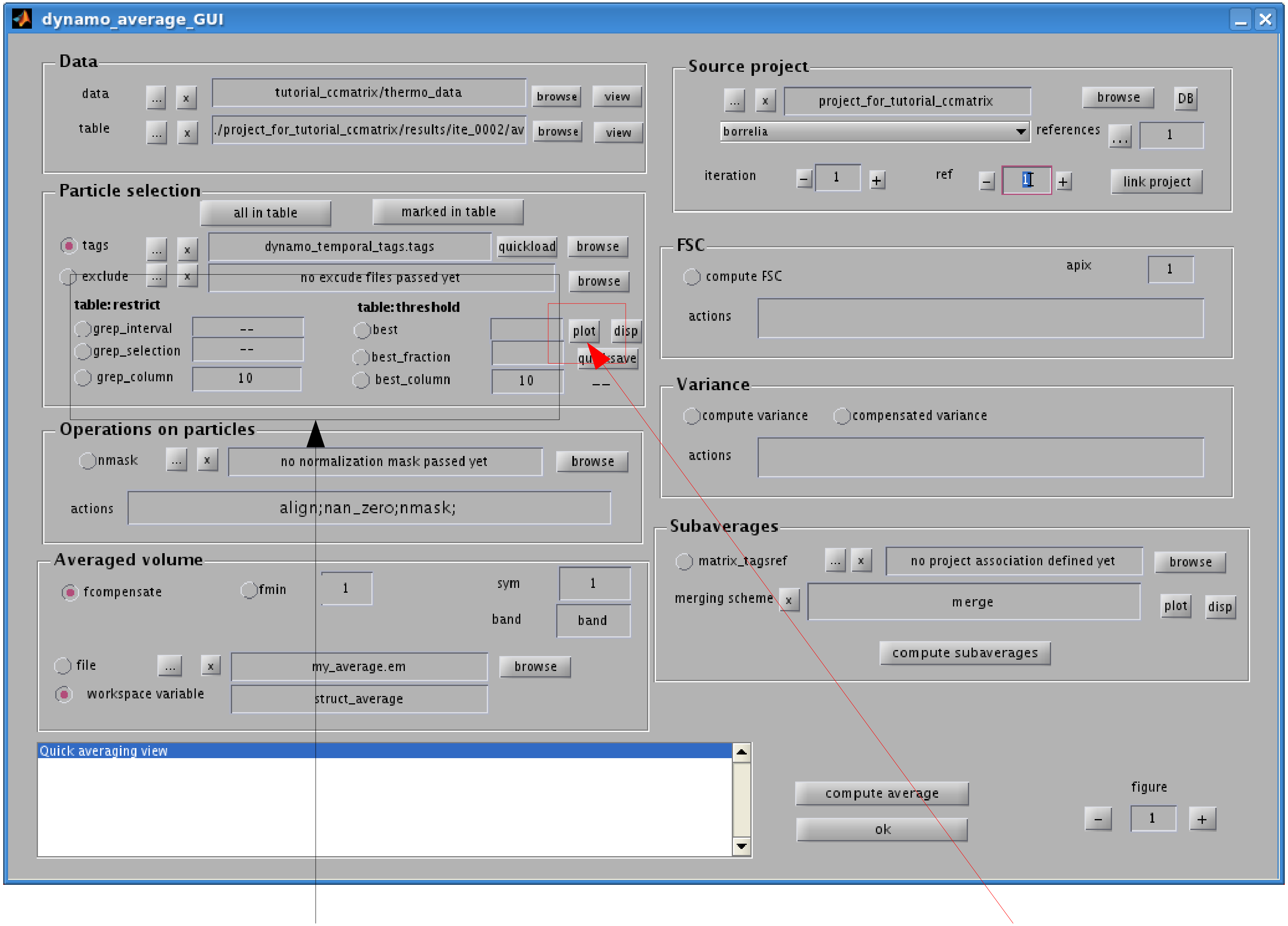


Selected tags: 11 ranging from 17 to 155

```
 -Inf Inf
 -------------
 -Inf Inf
 -------------
created by thresholding properties has been added to the manual Selection
```

We can now open the average tool

In the GUI for averaging, all the input fields contain pipelined values from the gallery



dynamo_average_GUI

Data

data: tutorial_ccmatrix/thermo_data

table: ./project_for_tutorial_ccmatrix/results/ite_0002/av

Particle selection

● tags: dynamo_temporal_tags.tags

○ exclude: no excude files passed yet

table:restrict

○ grep_interval: --

○ grep_selection: --

○ grep_column: 10

table:threshold

○ best:

○ best_fraction:

○ best_column: 10

--

Operations on particles

○ nmask: no normalization mask passed yet

actions: align;nan_zero;nmask;

Averaged volume

● fcompensate

○ fmin: 1

sym: 1

band: band

○ file: my_average.em

● workspace variable: struct_average

Quick averaging view

Source project

project_for_tutorial_ccmatrix

borrelia

references: 1

iteration: 1

ref: 1

FSC

○ compute FSC

apix: 1

actions:

Variance

○ compute variance

○ compensated variance

actions:

Subaverages

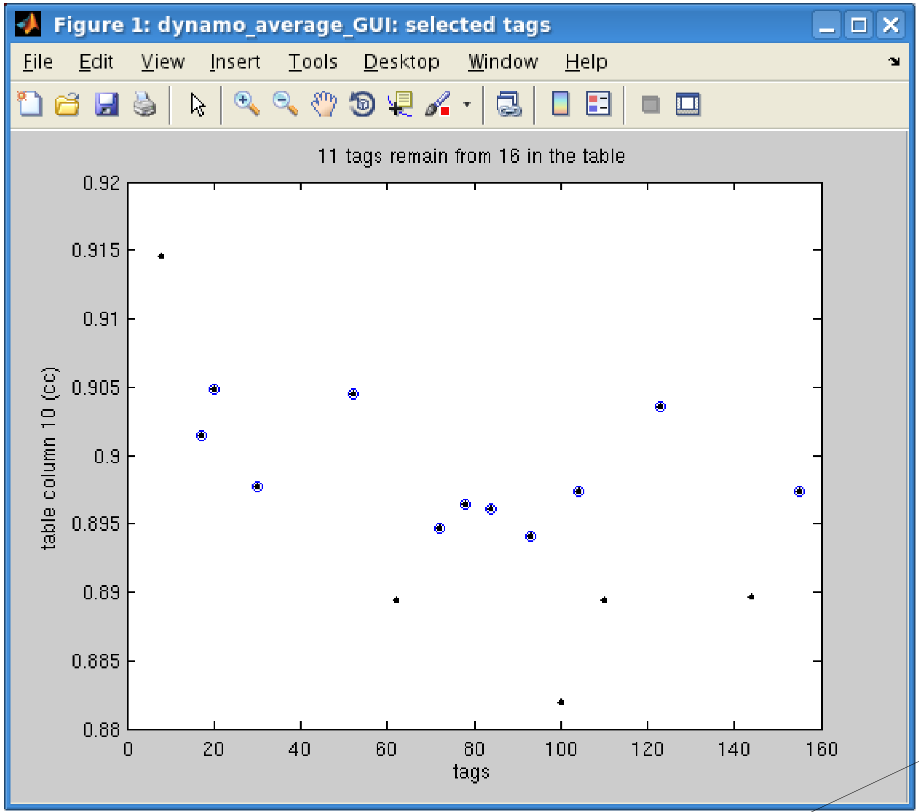
○ matrix_tagsref: no project association defined yet

merging scheme: merge

figure: 1

As you can operate further to restrict the set...

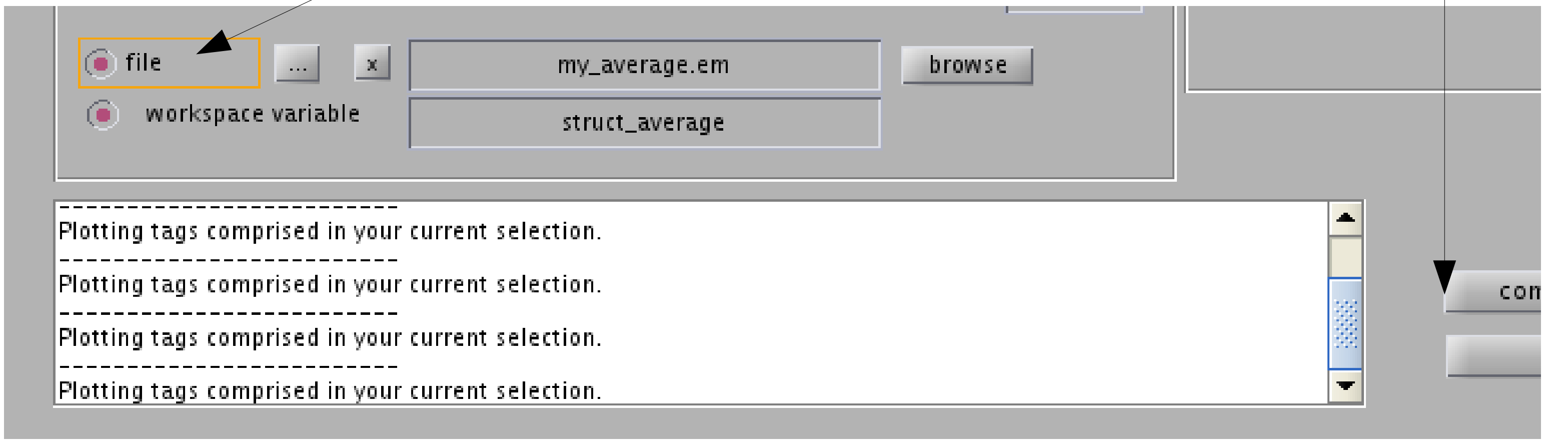
it is always a good idea to plot the tags
of the particles that will enter into the average



In the resulting plot, the tags that will enter the average are marked in blue. They are in the CC range that we indicated inside `dynamo_gallery`

We can thus determine where we want our output (as structured Matlab workspace variable and/or as file)

... and then compute the average...



● file: my_average.em

● workspace variable: struct_average

```
--------------------------
Plotting tags comprised in your current selection.
--------------------------
Plotting tags comprised in your current selection.
--------------------------
Plotting tags comprised in your current selection.
--------------------------
Plotting tags comprised in your current selection.
```

Navegating large data sets

(A final remark on `dynamo_gallery`)

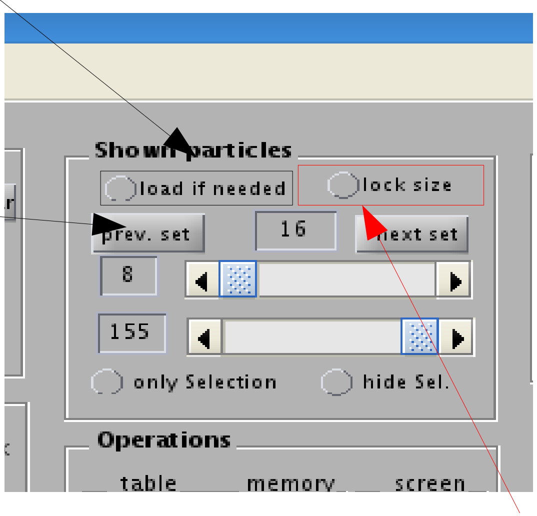
The tools in the [Shown particles] Panel will be useful (i.e. completely necessary) in real data sets.

They allow you to load into (or release from) memory subsets of particles from the hard disk in a controlled way.

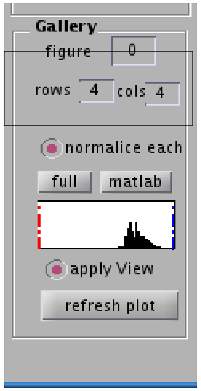
`[load if needed]`radiobutton:

When switched ON, previous/next batch of particles
defined in the *table* will be automatically loaded
into the shown scene if required
(by pushbuttons `[prev. set]` or `[next set.]`)
even if their particles are not already in *memory.*

`[prev. set]`
Brings into the shown scene
the previous batch of particles
already available in *memory*.



(batch size defined
implicitely here)



Switch ON if particles out of the scene
scope must be released immediately
from memory to allow loading new particles:
i.e. it locks the total size of the particle set allowable in memory

Back to work!: classification

Ok, we checked the alignment provided by the “table” files by using `dynamo_gallery` to inspect how data particles transform under these tables.

Now we start the next goal of the tutorial, the actual classification procedure.

We will base on the analysis of the ccmatrix objects created during our project**.**

How do we get the ccmatrix in general?

There are different ways to compute a ccmatrix to setup a classification.
In this tutorial we just computed them as “collateral product” in an alignment experiment.

In many cases it will be more appropriate to computed them independently.
This can be done using “`dynamo_ccmatrix_project_manager`”
(an analogous of “`dynamo_project_manager`”)
... or combinations of lower lever commands of the *Dynamo* toolbox.

Starting the analysis of the ccmatrix

```
>> dynamo_ccmatrix_analysis('project','project_for_tutorial_ccmatrix');
```

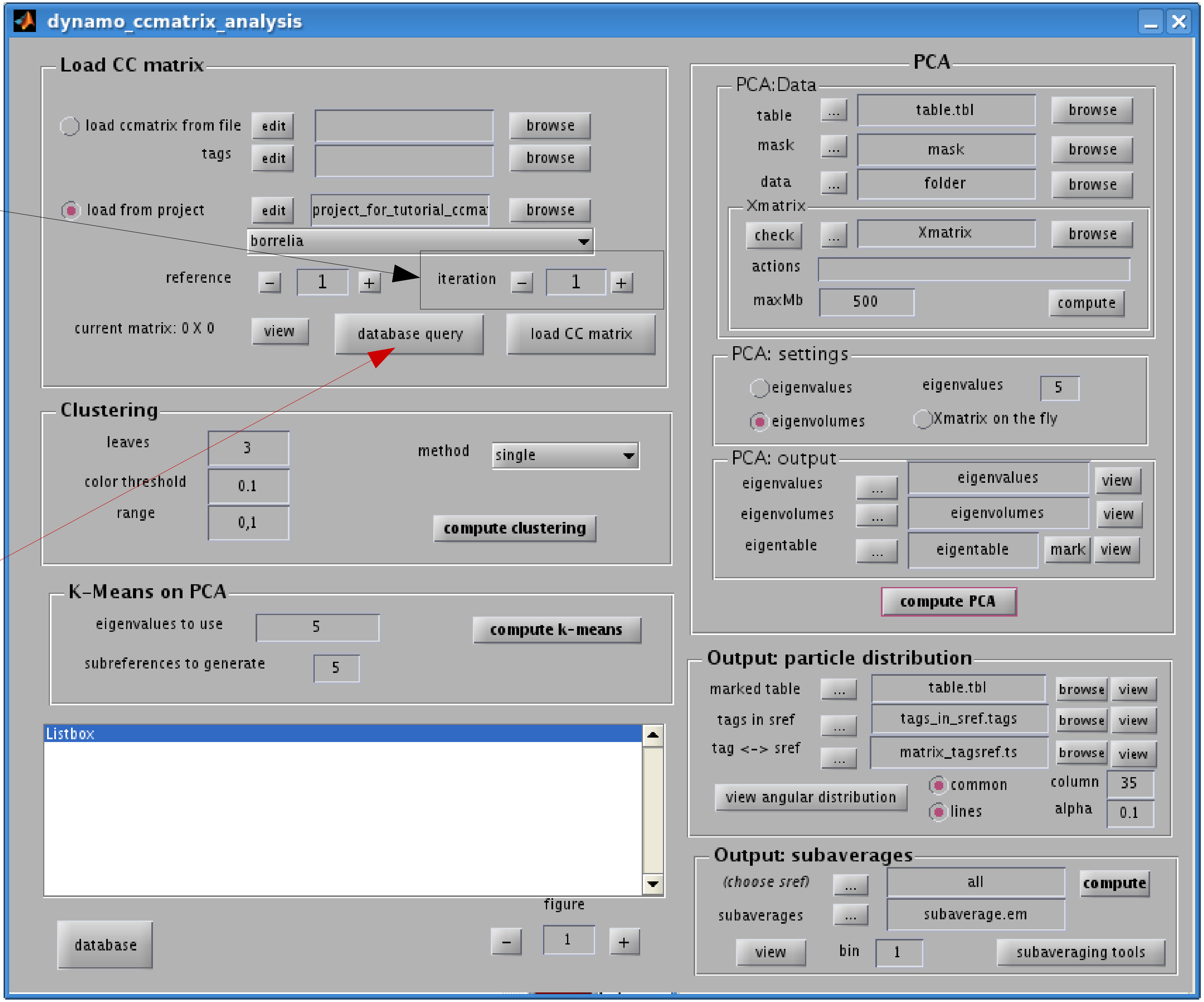
`dynamo_ccmatrix_analysis` will open on this project

By default, it opens
on the first iteration.

This is correct, we want
to analyze first this one.

click here to fill up
all fields of the GUI
according to the
current values of
* project
* iteration (1)
* reference (1)

This will pipeline
all the next steps
to these settings



dynamo_ccmatrix_analysis

Load CC matrix
○ load ccmatrix from file
tags
● load from project: project_for_tutorial_ccma
borrelia
reference: 1
iteration: 1
current matrix: 0 X 0

Clustering
leaves: 3
color threshold: 0.1
range: 0,1
method: single

K-Means on PCA
eigenvalues to use: 5
subreferences to generate: 5

Listbox

figure: 1

PCA

PCA:Data
table: table.tbl
mask: mask
data: folder
Xmatrix: Xmatrix
actions:
maxMb: 500

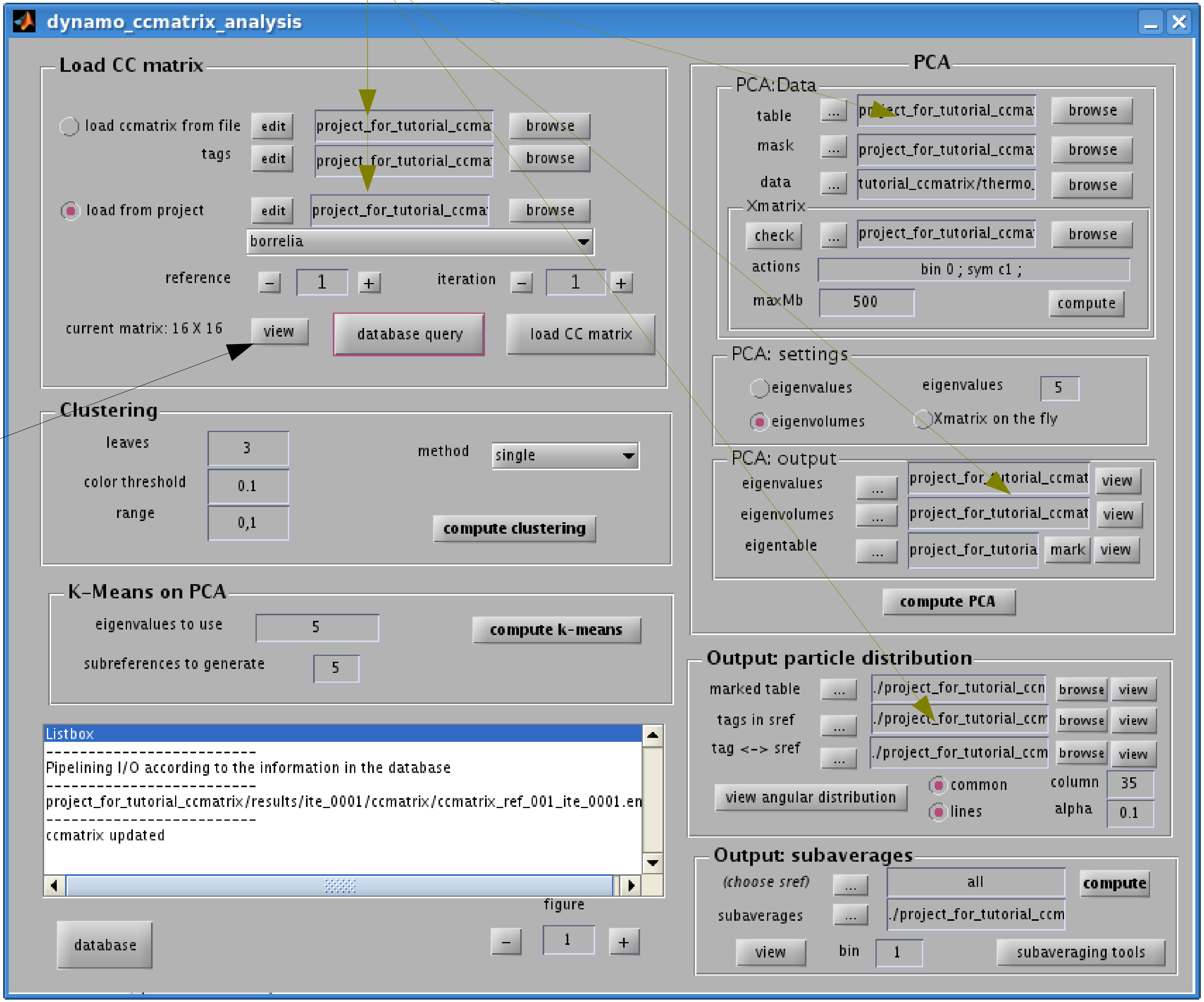
PCA: settings
○ eigenvalues
● eigenvolumes
eigenvalues: 5
○ Xmatrix on the fly

PCA: output
eigenvalues: eigenvalues
eigenvolumes: eigenvolumes
eigentable: eigentable

Output: particle distribution
marked table: table.tbl
tags in sref: tags_in_sref.tags
tag <-> sref: matrix_tagsref.ts
● common
● lines
column: 35
alpha: 0.1

Output: subaverages
(choose sref): all
subaverages: subaverage.em
bin: 1

Now, all the fields are coherently filled with valid database locations and you can proceed:



dynamo_ccmatrix_analysis

Load CC matrix

○ load ccmatrix from file: project_for_tutorial_ccma
tags: project_for_tutorial_ccma
● load from project: project_for_tutorial_ccma
borrelia
reference: 1 — iteration: 1
current matrix: 16 X 16

Clustering

leaves: 3
color threshold: 0.1
range: 0,1
method: single

K-Means on PCA

eigenvalues to use: 5
subreferences to generate: 5

```
Listbox
-------------------------
Pipelining I/O according to the information in the database
-------------------------
project_for_tutorial_ccmatrix/results/ite_0001/ccmatrix/ccmatrix_ref_001_ite_0001.em
-------------------------
ccmatrix updated
```

figure: 1

PCA

PCA:Data
table: project_for_tutorial_ccma
mask: project_for_tutorial_ccma
data: tutorial_ccmatrix/thermo_
Xmatrix: project_for_tutorial_ccma
actions: bin 0 ; sym c1 ;
maxMb: 500

PCA: settings
○ eigenvalues
● eigenvolumes
eigenvalues: 5
○ Xmatrix on the fly

PCA: output
eigenvalues: project_for_tutorial_ccmat
eigenvolumes: project_for_tutorial_ccmat
eigentable: project_for_tutoria

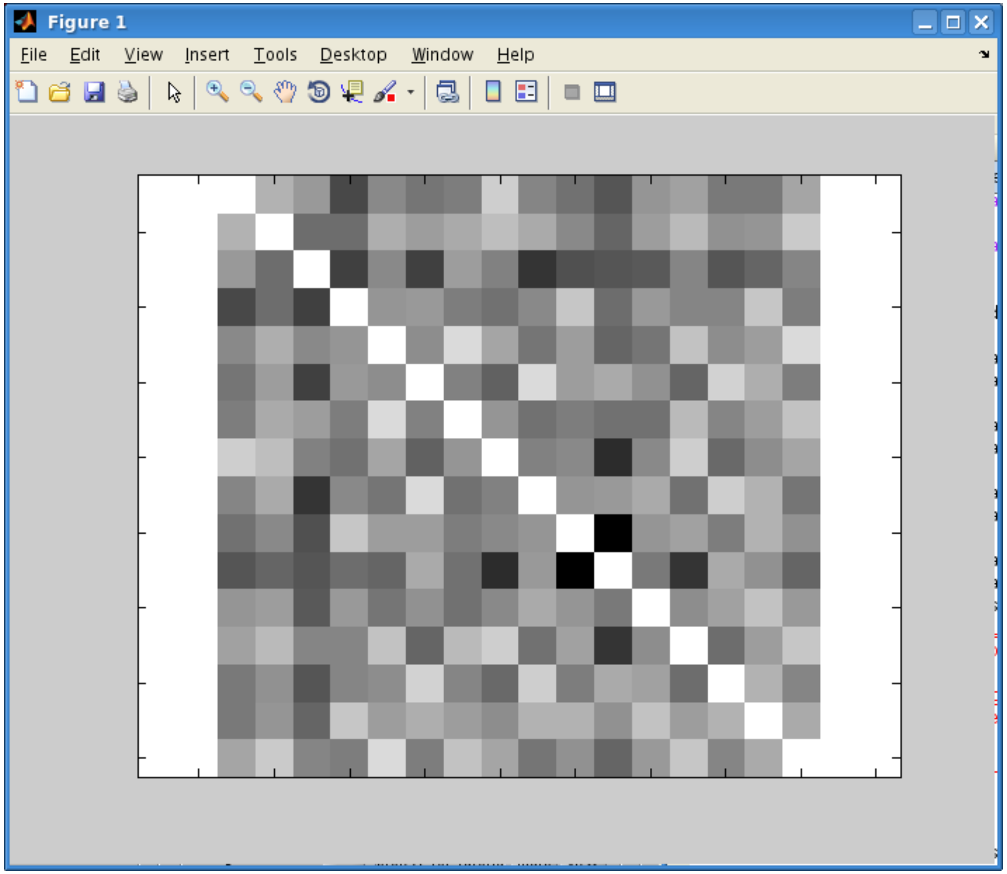
Output: particle distribution

marked table: ./project_for_tutorial_ccn
tags in sref: ./project_for_tutorial_ccm
tag <-> sref: ./project_for_tutorial_ccm
● common
● lines
column: 35
alpha: 0.1

Output: subaverages

(choose sref): all
subaverages: ./project_for_tutorial_ccm
bin: 1

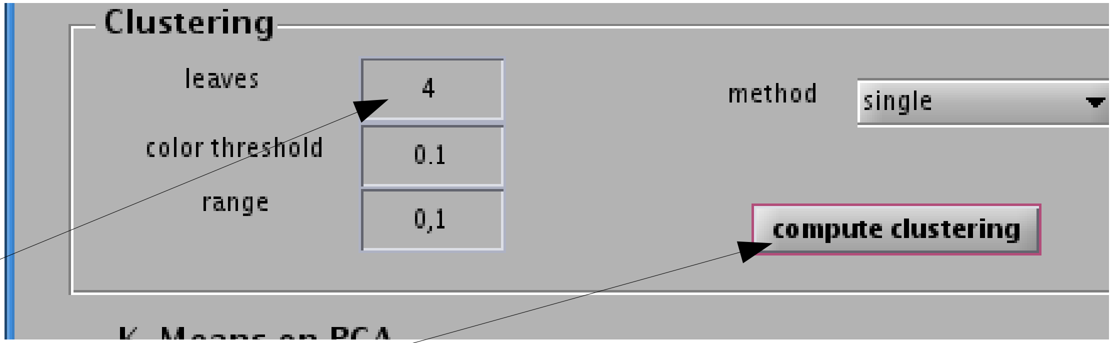
Click here to see the cc-matrix



This matrix represents the similarity of each pair of particles after the first round of alignment.

Missing wedge compensation has been taken into account by filtering both aligned particles to the common fourier component.

We first try to create a basic classification, using the Matlab commands for classification based only on this distance matrix



Clustering

leaves: 4

color threshold: 0.1

range: 0,1

method: single

In the GUI dynamo_ccmatrix_analysis

* choose 4 leaves

* press here to compute a classification

The graphical output should look similar to this:

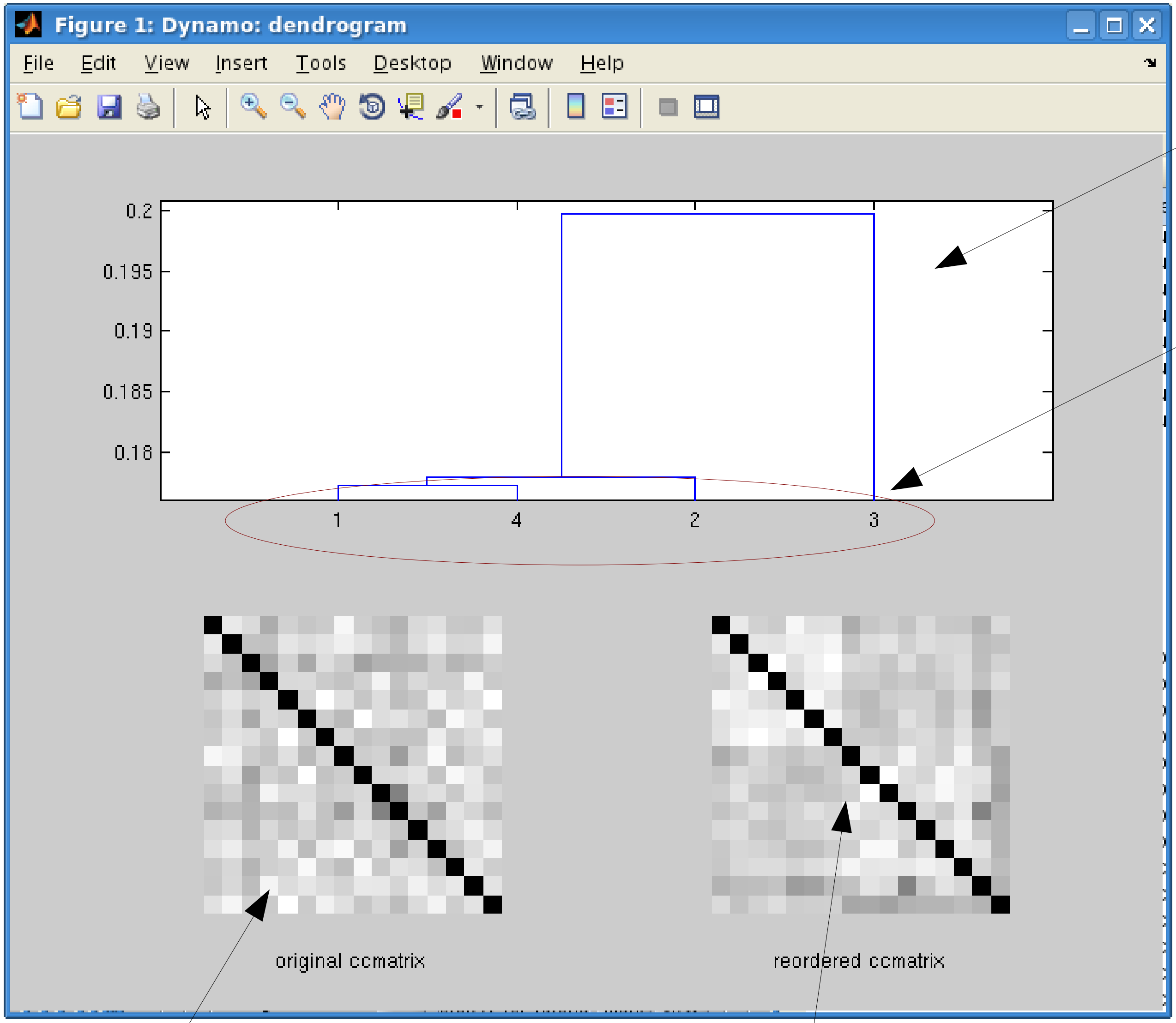


Figure 1: Dynamo: dendrogram
File Edit View Insert Tools Desktop Window Help
0.2
0.195
0.19
0.185
0.18
1
4
2
3
original ccmatrix
reordered ccmatrix

Hierarchical structure of the classification

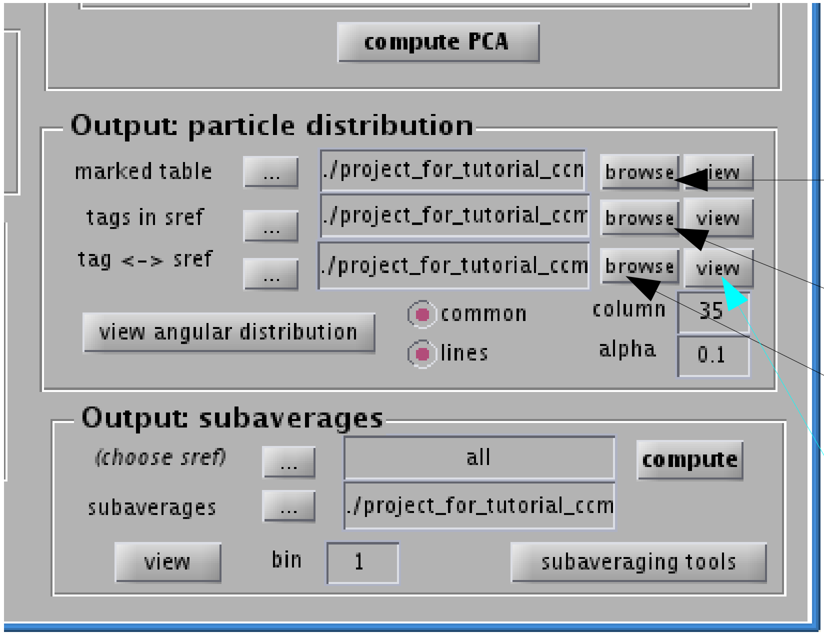
Labels of sets created in particles
Indexes are “subreferences” in Dynamo

The original matrix

The matrix reordered according
To the computed classification

In this case, one starts to see the presence of two populations

Finding the particles



compute PCA

Output: particle distribution

- marked table ... ./project_for_tutorial_ccn browse view
- tags in sref ... ./project_for_tutorial_ccm browse view
- tag <-> sref ... ./project_for_tutorial_ccm browse view

view angular distribution ● common ○ lines column 35 alpha 0.1

Output: subaverages

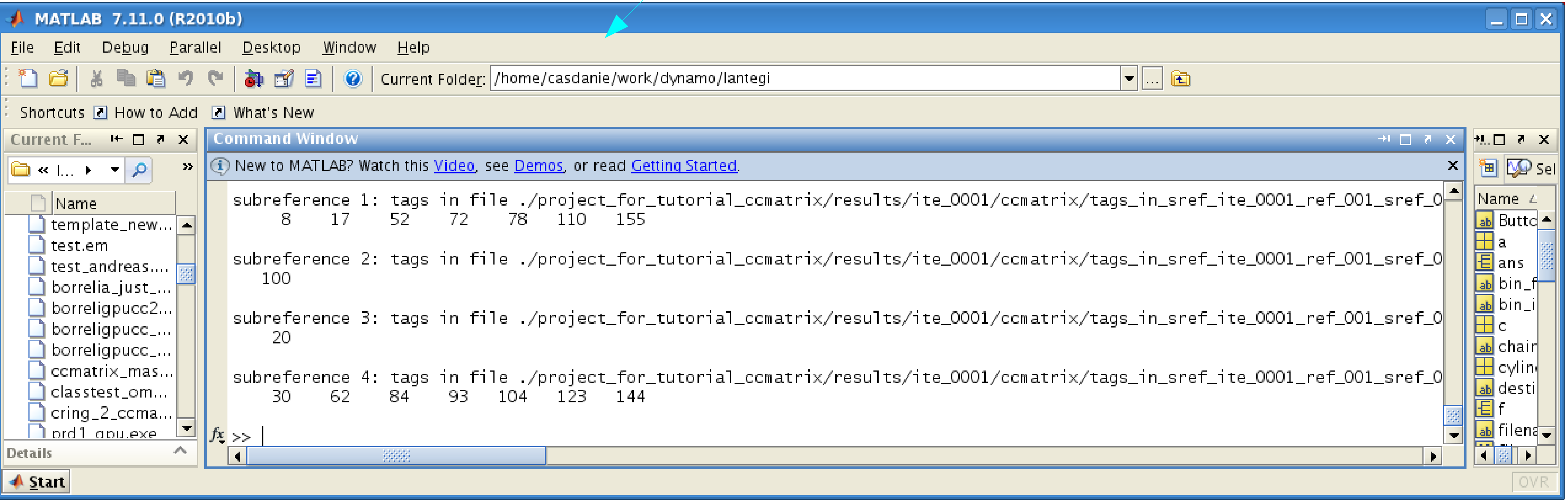
- (choose sref) ... all compute
- subaverages ... ./project_for_tutorial_ccm
- view bin 1 subaveraging tools

The identity of the particles assigned to each subreference
Is stored in these files, in different formats.

- A Dynamo-type table, assigning the subreference number of each particle in the column 35.
- A file (extension .tags) for each subrefrence
- A single two column file with (tag/sreference) pairs at each row.

They are all text files and can be opened with an editor.
or displayed into the screen using type (Matlab) or cat (Linux)

You can also click the [view] option suggested in the GUI
Or right click on a file name to get more suggestions.



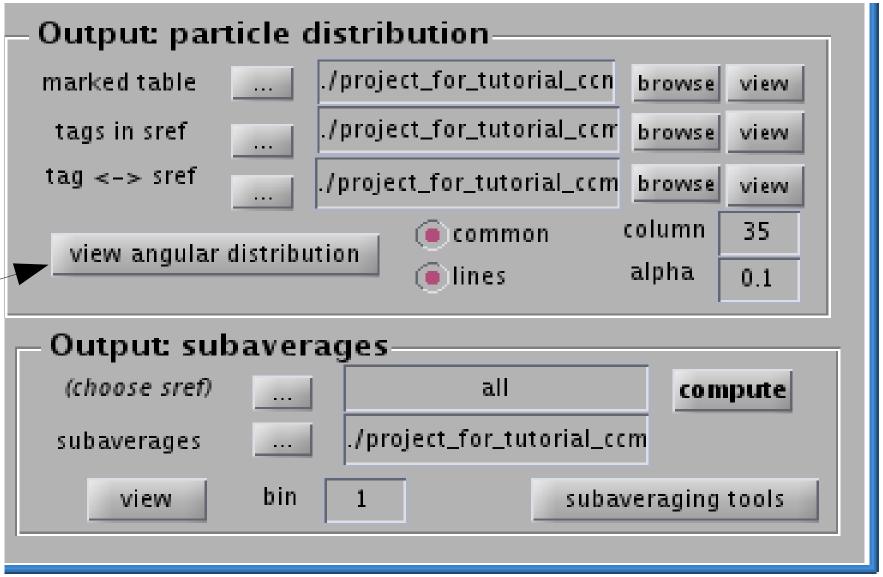
Current Folder: /home/casdanie/work/dynamo/lantegi

New to MATLAB? Watch this Video, see Demos, or read Getting Started.

```
subreference 1: tags in file ./project_for_tutorial_ccmatrix/results/ite_0001/ccmatrix/tags_in_sref_ite_0001_ref_001_sref_0
     8     17     52     72     78    110    155

subreference 2: tags in file ./project_for_tutorial_ccmatrix/results/ite_0001/ccmatrix/tags_in_sref_ite_0001_ref_001_sref_0
   100

subreference 3: tags in file ./project_for_tutorial_ccmatrix/results/ite_0001/ccmatrix/tags_in_sref_ite_0001_ref_001_sref_0
    20

subreference 4: tags in file ./project_for_tutorial_ccmatrix/results/ite_0001/ccmatrix/tags_in_sref_ite_0001_ref_001_sref_0
    30     62     84     93    104    123    144

fx >>
```

Distribution of particle orientations

This is a good moment to check if the created classes are an orientation artifact. A possible way is to depict the orientation of each particle as a point in the unit sphere, indicating with the same color those particles on the same Subreference number.



Output: particle distribution

marked table ... ./project_for_tutorial_ccn browse view

tags in sref ... ./project_for_tutorial_ccm browse view

tag <-> sref ... ./project_for_tutorial_ccm browse view

view angular distribution ● common ● lines column 35 alpha 0.1

Output: subaverages

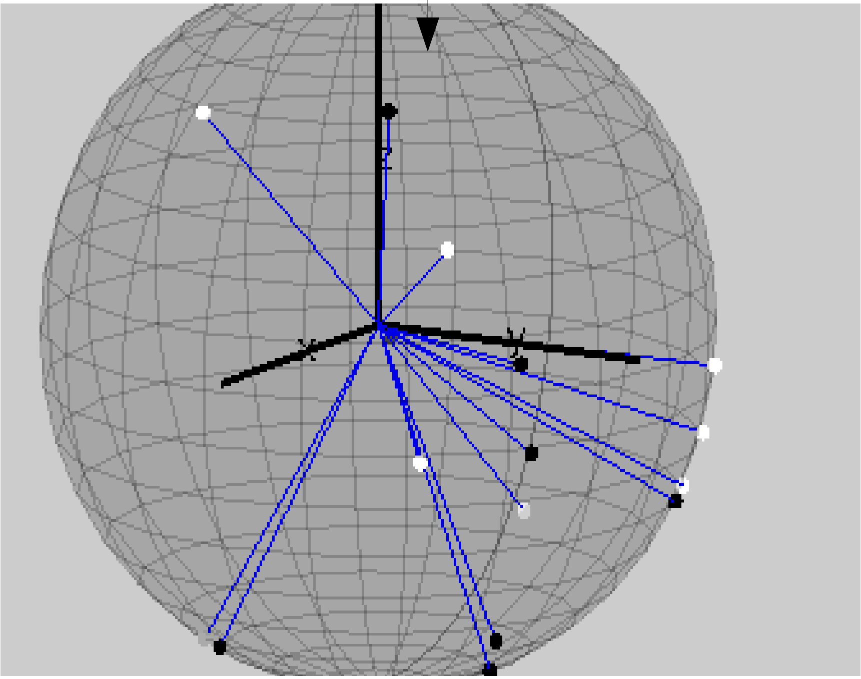
(choose sref) ... all compute

subaverages ... ./project_for_tutorial_ccm

view bin 1 subaveraging tools

Click here

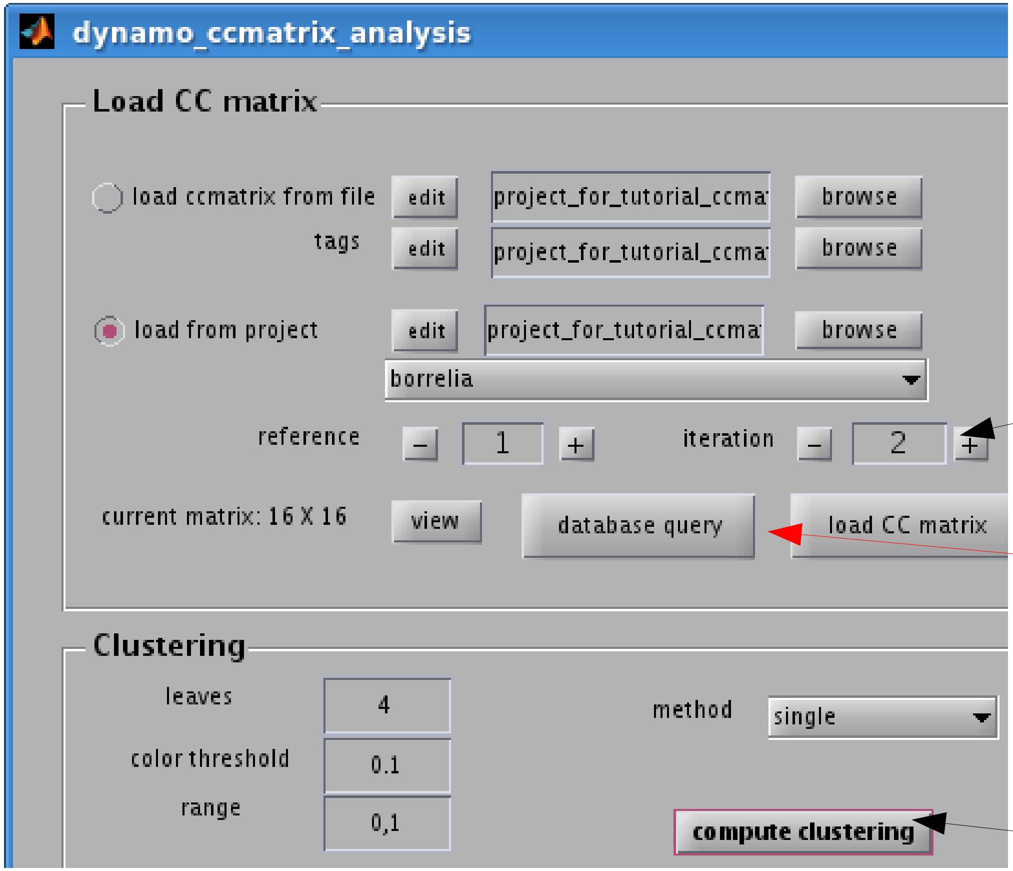
... to produce this depiction



Note:
dynamo_tableview on the “marked table” will give you a more flexible insight into the information coded in a table.

Better alignment -> better classification

What happens if we classify the particles according to the results of the SECOND alignment round?



dynamo_ccmatrix_analysis

Load CC matrix

○ load ccmatrix from file: project_for_tutorial_ccma

tags: project_for_tutorial_ccma

● load from project: project_for_tutorial_ccma

borrelia

reference: 1

iteration: 2

current matrix: 16 X 16

Clustering

leaves: 4

color threshold: 0.1

range: 0,1

method: single

Input: [Iteration]: 2
and press [[Return]]

Update the database links
(just pressing here)

And compute again
a classification

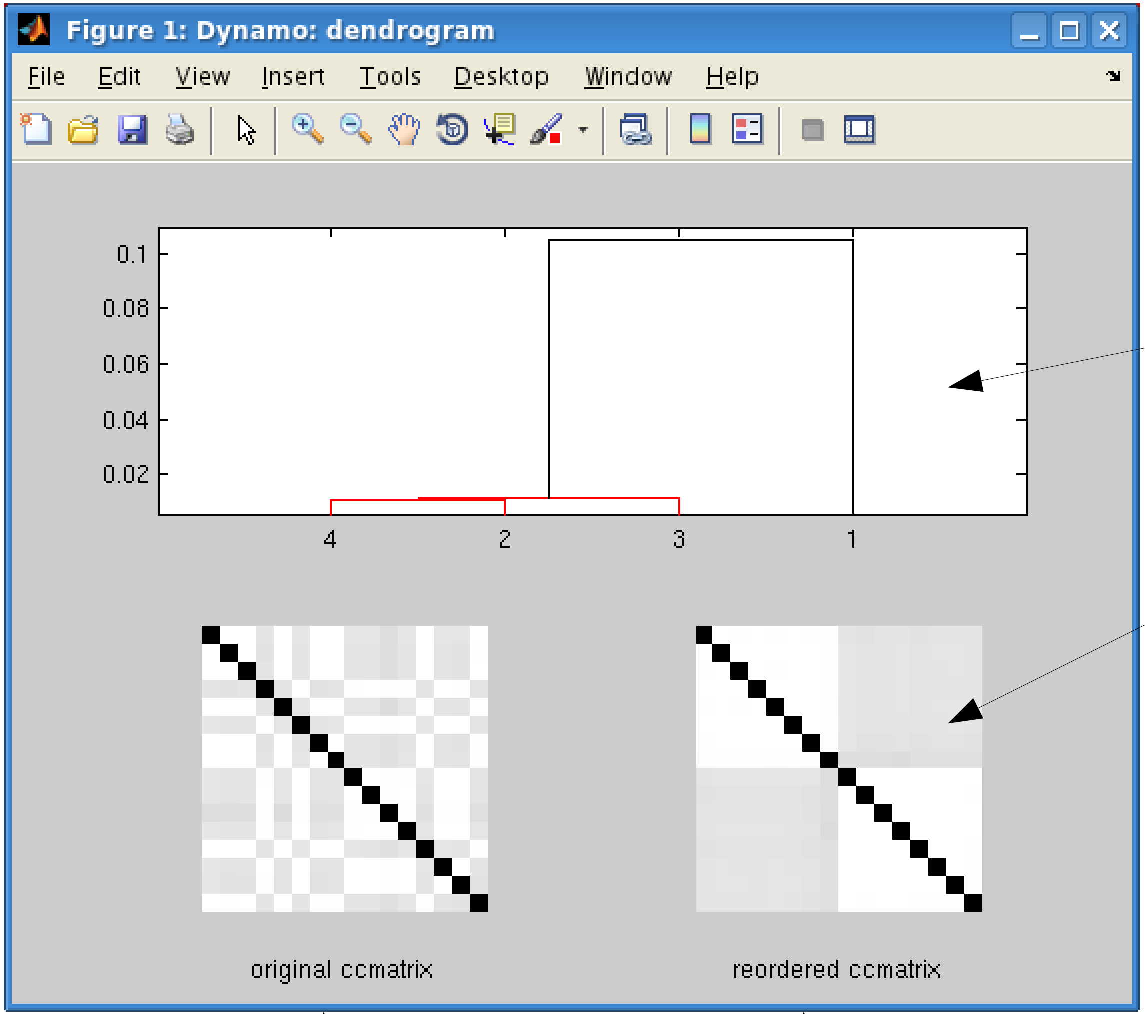


Figure 1: Dynamo: dendrogram
File Edit View Insert Tools Desktop Window Help
0.1
0.08
0.06
0.04
0.02
4
2
3
1
original ccmatrix
reordered ccmatrix

Now the two populations are evident, both in the hierarchical tree (where 3 of the 4 generated subreferences are clearly related) and in the reordered ccmatrix.

The difference of similarity between particles of both groups is more apparent...

... making the classification work better

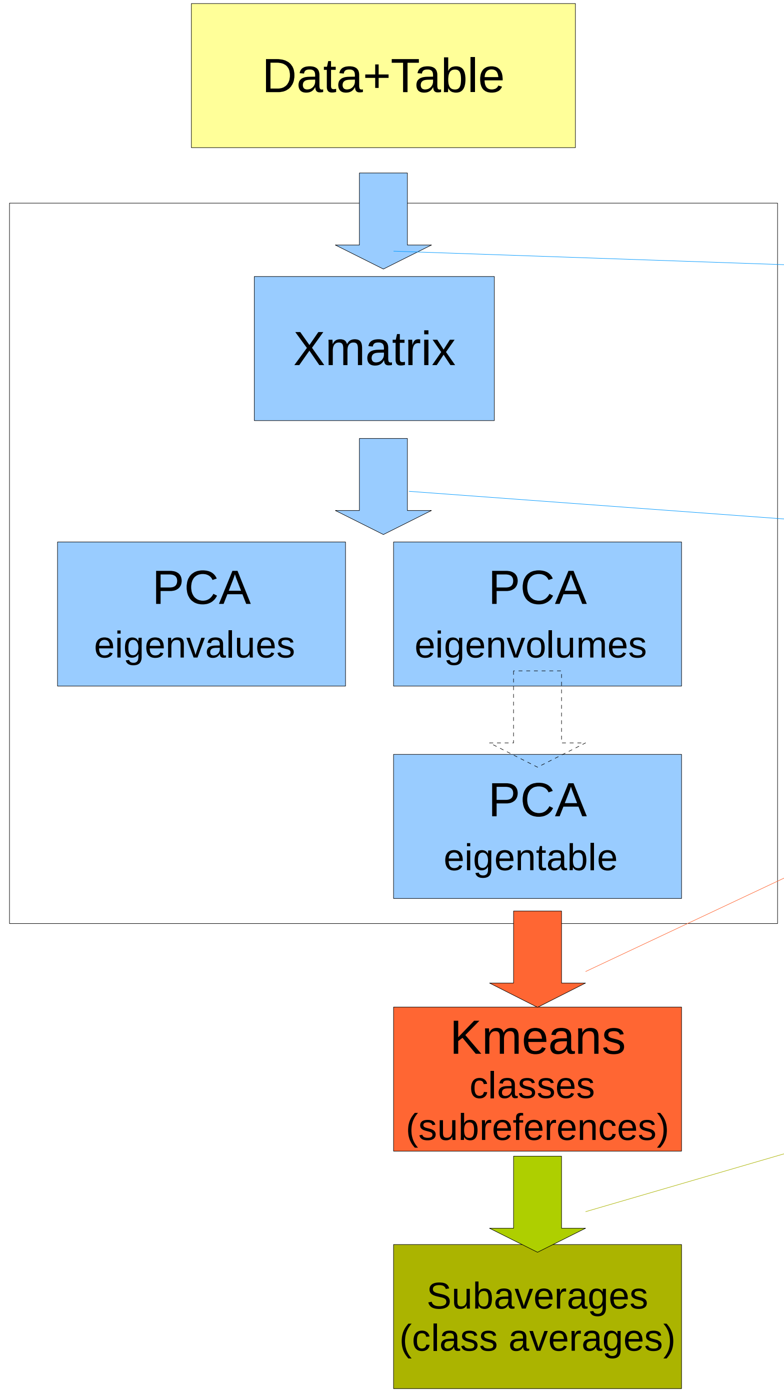
Computing Principal Component Analysis

Why PCA?

The distance-based classification did already provide a good result in this case, and we could already produce our averages using the particle sets assigned to each produced subreference.

However (and in our experience) this kind of classification will not perform well in many real cases.

We will now describe how the same GUI for ccmatrix analysis can be used to drive a PCA computation.



Data+Table

Xmatrix

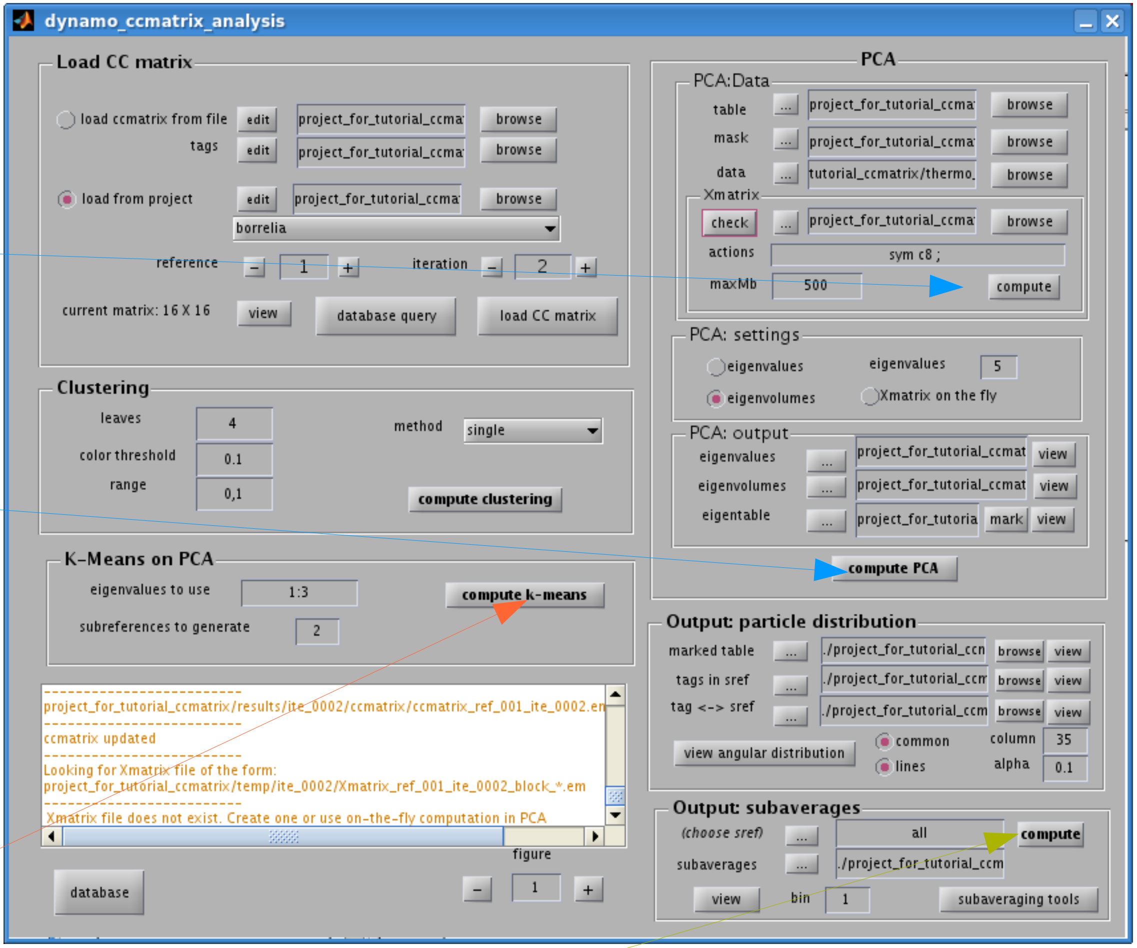
PCA eigenvalues

PCA eigenvolumes

PCA eigentable

Kmeans classes (subreferences)

Subaverages (class averages)



dynamo_ccmatrix_analysis

Load CC matrix

○ load ccmatrix from file: project_for_tutorial_ccma

tags: project_for_tutorial_ccma

● load from project: project_for_tutorial_ccma

borrelia

reference: 1 iteration: 2

current matrix: 16 X 16

Clustering

leaves: 4

color threshold: 0.1

range: 0,1

method: single

K-Means on PCA

eigenvalues to use: 1:3

subreferences to generate: 2

```
--------------------------
project_for_tutorial_ccmatrix/results/ite_0002/ccmatrix/ccmatrix_ref_001_ite_0002.em
--------------------------
ccmatrix updated
--------------------------
Looking for Xmatrix file of the form:
project_for_tutorial_ccmatrix/temp/ite_0002/Xmatrix_ref_001_ite_0002_block_*.em
--------------------------
 Xmatrix file does not exist. Create one or use on-the-fly computation in PCA
```

figure: 1

PCA

PCA:Data

table: project_for_tutorial_ccma

mask: project_for_tutorial_ccma

data: tutorial_ccmatrix/thermo_

Xmatrix: project_for_tutorial_ccma

actions: sym c8 ;

maxMb: 500

PCA: settings

○ eigenvalues

● eigenvolumes

eigenvalues: 5

○ Xmatrix on the fly

PCA: output

eigenvalues: project_for_tutorial_ccmat

eigenvolumes: project_for_tutorial_ccmat

eigentable: project_for_tutoria

Output: particle distribution

marked table: ./project_for_tutorial_ccn

tags in sref: ./project_for_tutorial_ccm

tag <-> sref: ./project_for_tutorial_ccm

● common

● lines

column: 35

alpha: 0.1

Output: subaverages

(choose sref): all

subaverages: ./project_for_tutorial_ccm

bin: 1

Overview on PCA analysis

trough `dynamo_ccmatrix_analysis`

More information in the Classification Roadmap document in our website

What is the Xmatrix?

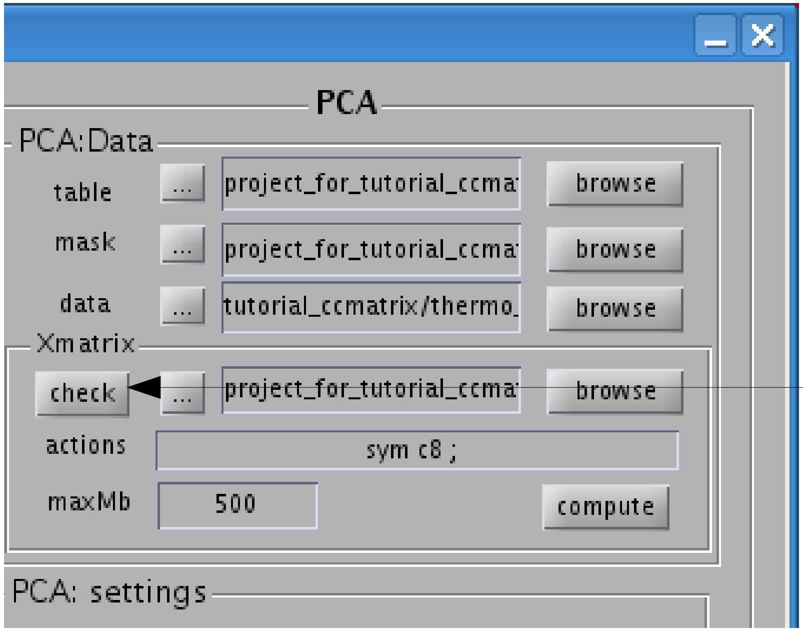


table: project_for_tutorial_ccma

mask: project_for_tutorial_ccma

data: tutorial_ccmatrix/thermo_

Xmatrix: project_for_tutorial_ccma

actions: sym c8 ;

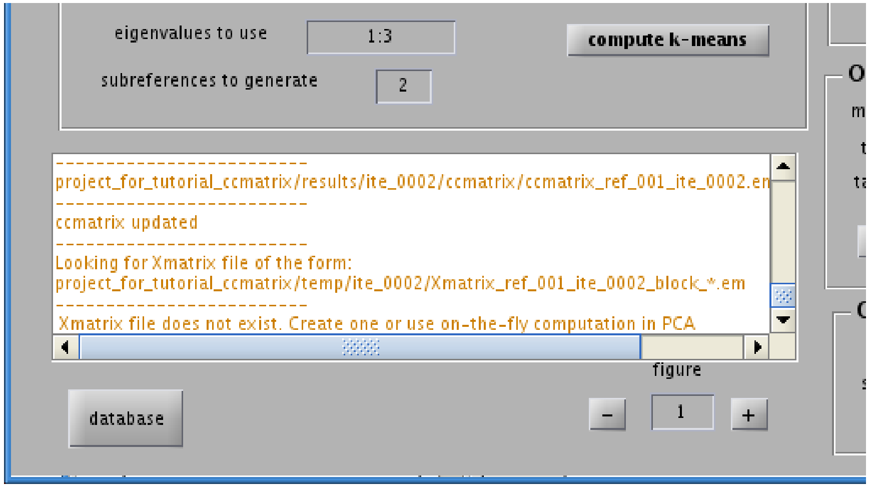
maxMb: 500

The Xmatrix is just a computing help.
Is a matrix that stores:
* in each row a particle
* in each column a pixel value

Check if the Xmatrix is already available

In this tutorial the Xmatrix not be available at this point:
 the parameters of our original project
`project_for_tutorial_ccmatrix`
did not include a command to create the Xmatrix.

The information area will thus warn us:



eigenvalues to use: 1:3

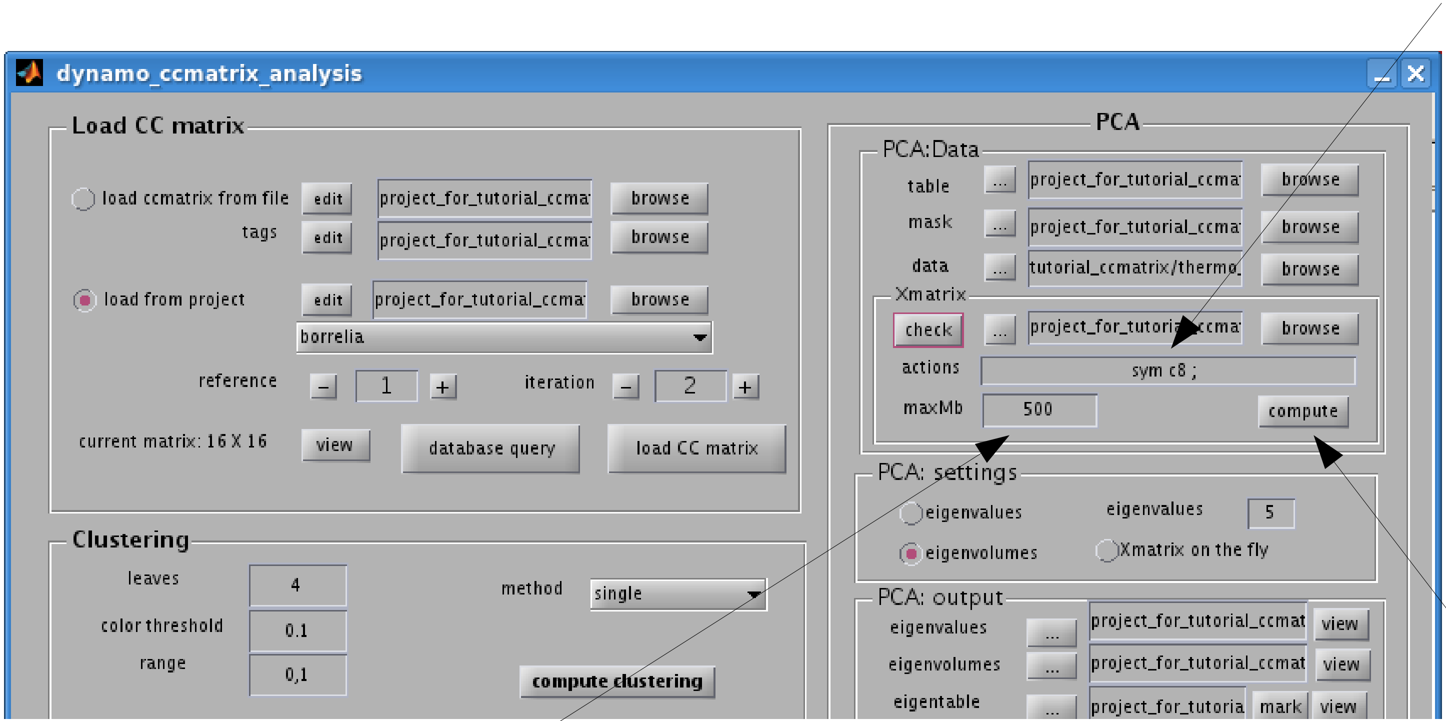
subreferences to generate: 2

figure: 1

```
-------------------------
project_for_tutorial_ccmatrix/results/ite_0002/ccmatrix/ccmatrix_ref_001_ite_0002.em
-------------------------
ccmatrix updated
-------------------------
Looking for Xmatrix file of the form:
project_for_tutorial_ccmatrix/temp/ite_0002/Xmatrix_ref_001_ite_0002_block_*.em
-------------------------
 Xmatrix file does not exist. Create one or use on-the-fly computation in PCA
```

Computing a Xmatrix

We can select an series of “actions” (i.e. bandpass, resizing, symmetrization)
to be operated on each particle



dynamo_ccmatrix_analysis

Load CC matrix

○ load ccmatrix from file: project_for_tutorial_ccma

tags: project_for_tutorial_ccma

● load from project: project_for_tutorial_ccma

borrelia

reference: 1

iteration: 2

current matrix: 16 X 16

Clustering

leaves: 4

color threshold: 0.1

range: 0,1

method: single

PCA

PCA:Data

table: project_for_tutorial_ccma

mask: project_for_tutorial_ccma

data: tutorial_ccmatrix/thermo_

Xmatrix: project_for_tutorial_ccma

actions: sym c8 ;

maxMb: 500

PCA: settings

○ eigenvalues

● eigenvolumes

eigenvalues: 5

○ Xmatrix on the fly

PCA: output

eigenvalues: project_for_tutorial_ccmat

eigenvolumes: project_for_tutorial_ccmat

eigentable: project_for_tutoria

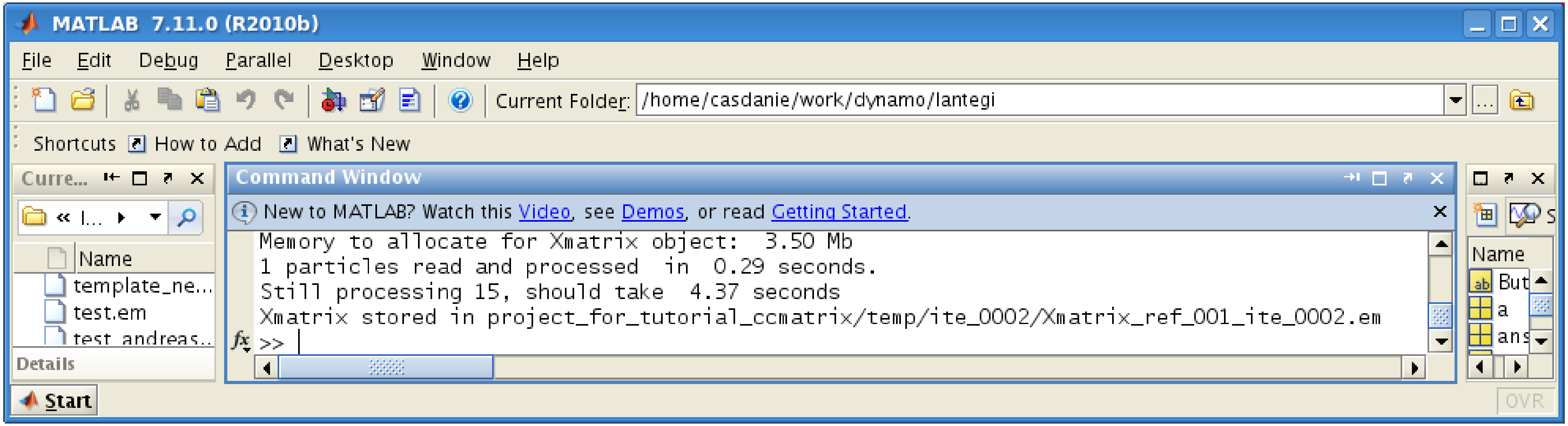
An Xmatrix can be huge if you have many particlesr very large particles.

Keeping large matrices in (RAM) memory can block your system.
You can use this parameters to tell Dynamo what is the largest matrix size
That you allow in your memory. If the Xmatrix of your problem turns to be
Bigger, Dynamo will sepate it in pieces and produce a separate file for
each matrix fragment.

This will slow down performance, but ensure the stability of your system.

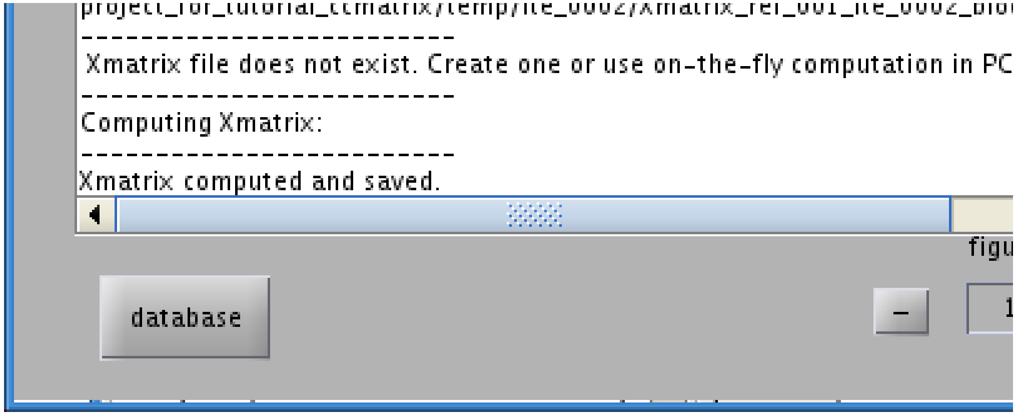
Once these parametes are
set we click here
to start computing
the Xmatrix

The Matlab (of Linux/MacOS shell) will be busy for a while...

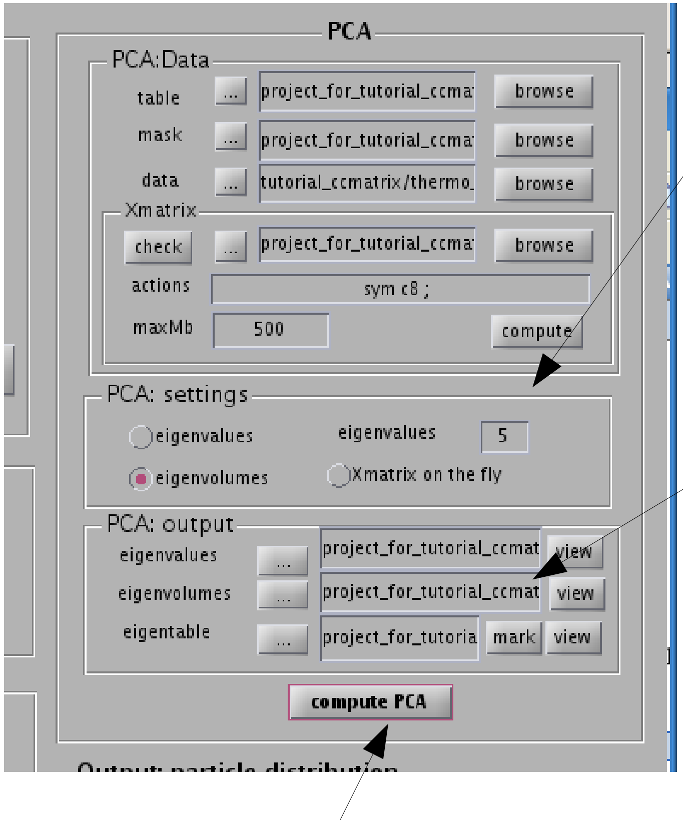

MATLAB 7.11.0 (R2010b)
File Edit Debug Parallel Desktop Window Help
Current Folder: /home/casdanie/work/dynamo/lantegi
Shortcuts How to Add What's New
Curre...
Name
template_ne...
test.em
test_andreas...
Details
Start
Command Window
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
Name
But
a
ans
OVR

```
Memory to allocate for Xmatrix object:  3.50 Mb
1 particles read and processed  in  0.29 seconds.
Still processing 15, should take  4.37 seconds
Xmatrix stored in project_for_tutorial_ccmatrix/temp/ite_0002/Xmatrix_ref_001_ite_0002.em
>> 
```

... until dynamo_ccmatrix_analysis anounces that the Xmatrix is ready



```
project_for_tutorial_ccmatrix/temp/ite_0002/Xmatrix_ref_001_ite_0002_bio
-------------------------
 Xmatrix file does not exist. Create one or use on-the-fly computation in PC
-------------------------
Computing Xmatrix:
-------------------------
Xmatrix computed and saved.
```



PCA:Data

table: project_for_tutorial_ccma

mask: project_for_tutorial_ccma

data: tutorial_ccmatrix/thermo_

Xmatrix: project_for_tutorial_ccma

actions: sym c8 ;

maxMb: 500

PCA: settings

○ eigenvalues

● eigenvolumes

eigenvalues: 5

○ Xmatrix on the fly

PCA: output

eigenvalues: project_for_tutorial_ccmat

eigenvolumes: project_for_tutorial_ccmat

eigentable: project_for_tutoria

Output: particle distribution

Choose 5 as total number of eigenvolumes to compute

Output will be stored in these files

In this tutorial, the output file names are standard database locations. They were automatically generated as we are working inside a project, but you can forward the output to other folders.

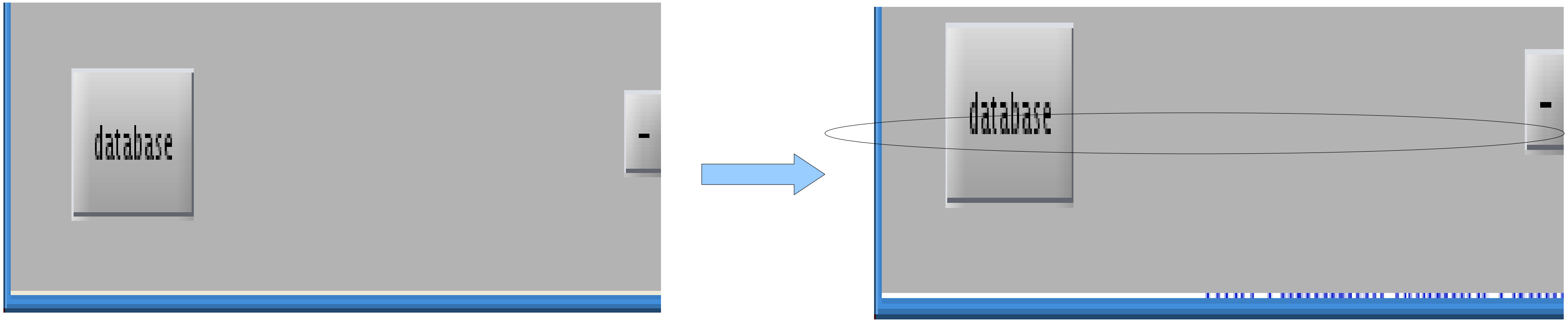
Now you can compute the PCA

This will assign to each particle its coefficient expansion in terms of the computed eigenvectors.

This information is integrated with the rest of the available information on a particle in the “eigentable” file.

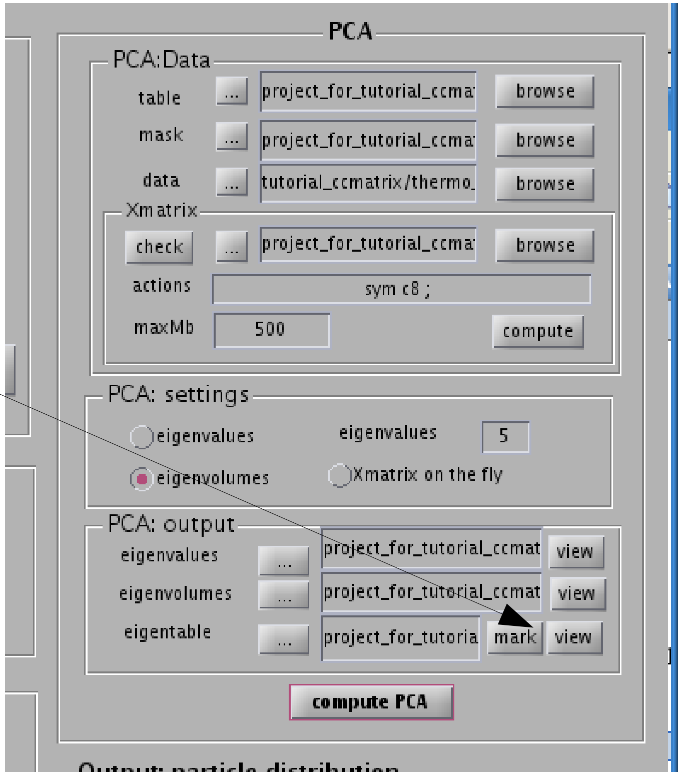
Columns 41, 42, etc.. will store the 1st, 2.nd coefficients, etc.

Let it some time to run....



Now, in the output “eigentable” each row has an associate set of eigenvectors. We can play a little to view them with `dynamo_tableview`:

Press here



PCA:Data
table ... project_for_tutorial_ccma
mask ... project_for_tutorial_ccma
data ... tutorial_ccmatrix/thermo_
Xmatrix ... project_for_tutorial_ccma
actions sym c8 ;
maxMb 500

PCA: settings
○ eigenvalues   eigenvalues 5
● eigenvolumes   ○ Xmatrix on the fly

PCA: output
eigenvalues ... project_for_tutorial_ccmat
eigenvolumes ... project_for_tutorial_ccmat
eigentable ... project_for_tutoria

Note that this is equivalent to open `dynamo_tableview` on the file indicated in the Information Area.

This can be done from the command line or right clicking on the field and selecting `dynamo_tableview` in the menu of possible actions that will popup.

Then, we want to see a scatterplot of the two first eigencomponents

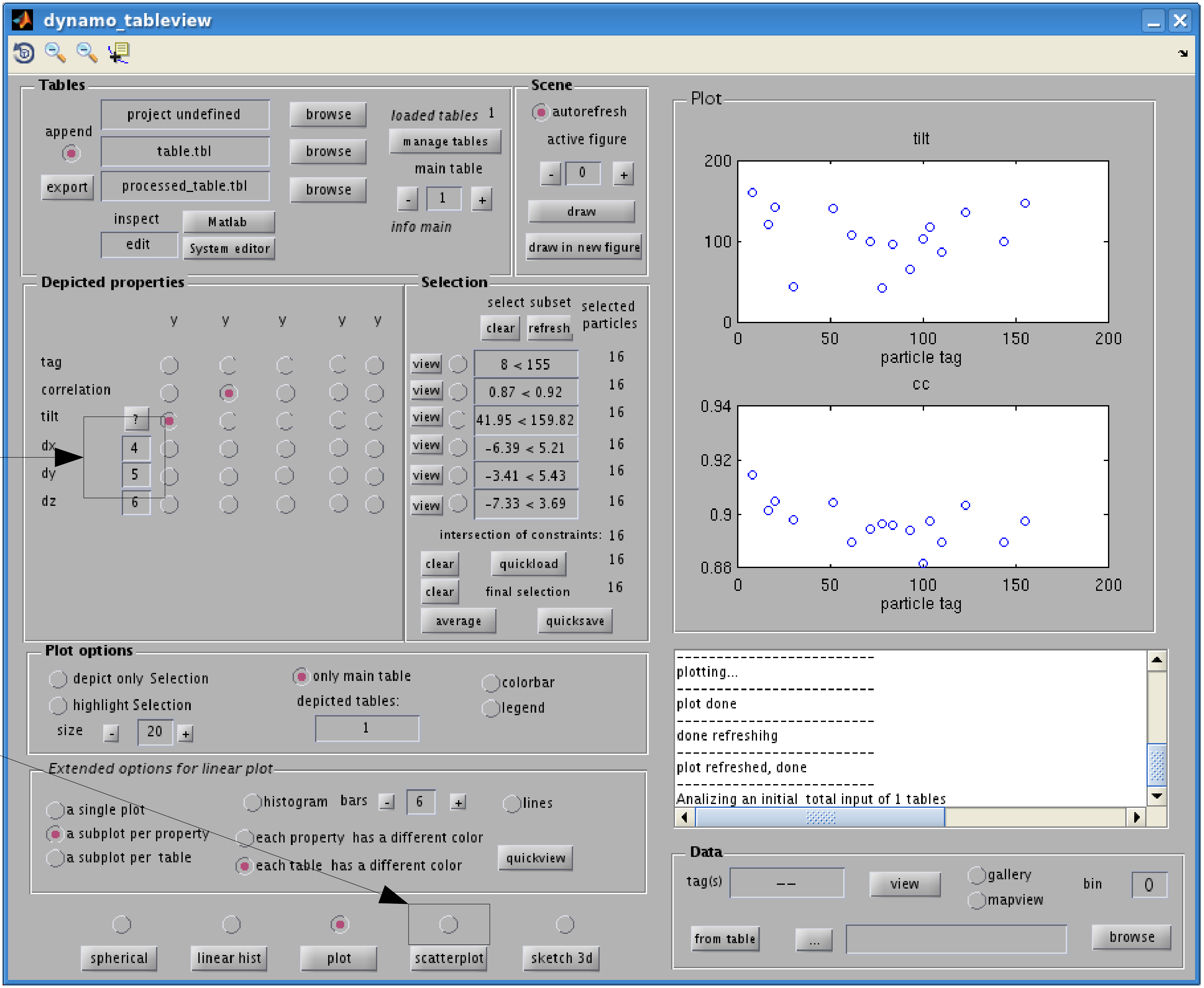
Exploring the eigentable

dynamo_tableview will open with on a scene similar to this one:

We want now to see a scatterplot of the two first eigencomponents of all the particles

So, we need to change the "Depicted prorperties":
We need to pick
columns 41 and 42

And we need to select the "scatterplot" selection modus



Tables: project undefined; table.tbl; processed_table.tbl; loaded tables 1; main table 1

Scene: ● autorefresh; active figure 0

Depicted properties: dx 4; dy 5; dz 6

	selected particles
8 < 155	16
0.87 < 0.92	16
41.95 < 159.82	16
-6.39 < 5.21	16
-3.41 < 5.43	16
-7.33 < 3.69	16
intersection of constraints:	16
quickload	16
final selection	16

Plot options: ○ depict only Selection; ○ highlight Selection; size 20; ● only main table; depicted tables: 1; ○ colorbar; ○ legend

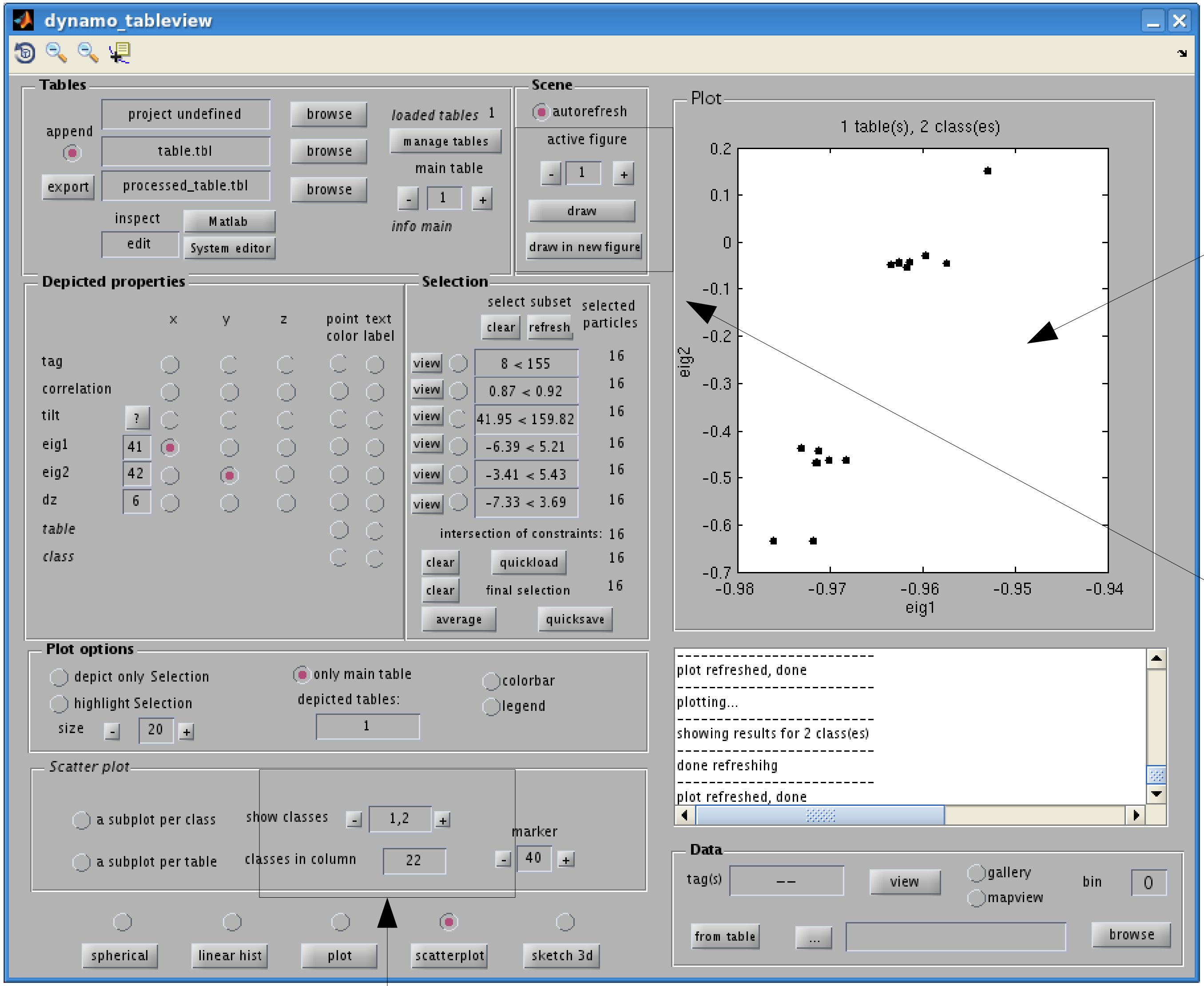
Extended options for linear plot: ○ a single plot; ● a subplot per property; ○ a subplot per table; ○ histogram bars 6; ○ lines; ○ each property has a different color; ● each table has a different color

○ spherical ○ linear hist ● plot ○ scatterplot ○ sketch 3d

```
---------------------------
plotting...
---------------------------
plot done
---------------------------
done refreshihg
---------------------------
plot refreshed, done
---------------------------
Analizing an initial total input of 1 tables
```

Data: tag(s) --; ○ gallery; ○ mapview; bin 0

With these depiction settings:



dynamo_tableview

Tables

append ●

project undefined

table.tbl

processed_table.tbl

loaded tables 1

main table 1

info main

Scene

● autorefresh

active figure 1

Depicted properties

		x	y	z	point text	color label
tag		○	○	○	○	○
correlation		○	○	○	○	○
tilt	?	○	○	○	○	○
eig1	41	●	○	○	○	○
eig2	42	○	●	○	○	○
dz	6	○	○	○	○	○
table					○	○
class					○	○

Selection

select subset	selected particles
8 < 155	16
0.87 < 0.92	16
41.95 < 159.82	16
-6.39 < 5.21	16
-3.41 < 5.43	16
-7.33 < 3.69	16

intersection of constraints: 16

quickload 16

final selection 16

Plot options

○ depict only Selection

○ highlight Selection

size 20

● only main table

depicted tables: 1

○ colorbar

○ legend

Scatter plot

○ a subplot per class

○ a subplot per table

show classes 1,2

classes in column 22

marker 40

○ spherical ○ linear hist ○ plot ● scatterplot ○ sketch 3d

```
---------------------------
plot refreshed, done
---------------------------
plotting...
---------------------------
showing results for 2 class(es)
---------------------------
done refreshihg
---------------------------
plot refreshed, done
```

Data

tag(s) --

○ gallery

○ mapview

bin 0

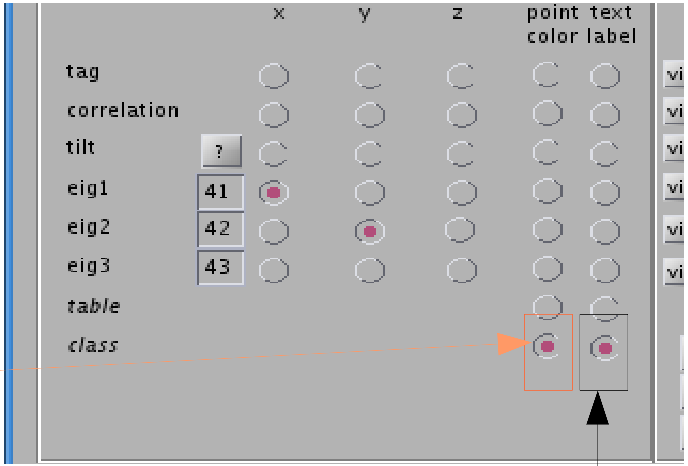
You should see two clusters (more or less)

Note that you can export the scene to edit it with Matlab native tools.

Note that the table has an original “cheat” mark in column 22, labeling which particles were generated in which class. We just select all the available classes for the depiction, to ensure that all the particles in the table are plotted (leave the field [show classes] empty, tableview will fill it with all the values detected in column 22)

Is the clustering produced
by our PCA analysis true?

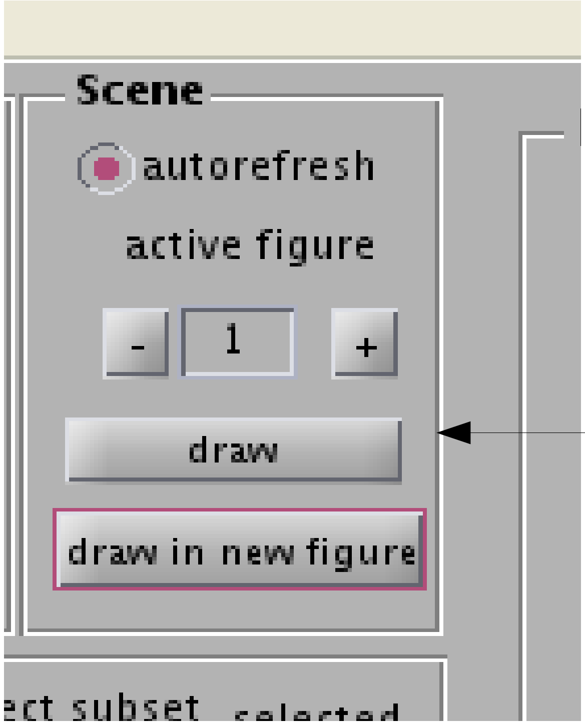
Let us compare it with the ground truth
(contained in column 22 of the table)



		x	y	z	point color	text label
tag		○	○	○	○	○
correlation		○	○	○	○	○
tilt	?	○	○	○	○	○
eig1	41	●	○	○	○	○
eig2	42	○	●	○	○	○
eig3	43	○	○	○	○	○
table					○	○
class					●	●

We color each the points in the
Scatter plot of the eig1/eig2
according to their value in column 22...

... and label label them also according to column 22



Scene

● autorefresh

active figure

- 1 +

draw

draw in new figure

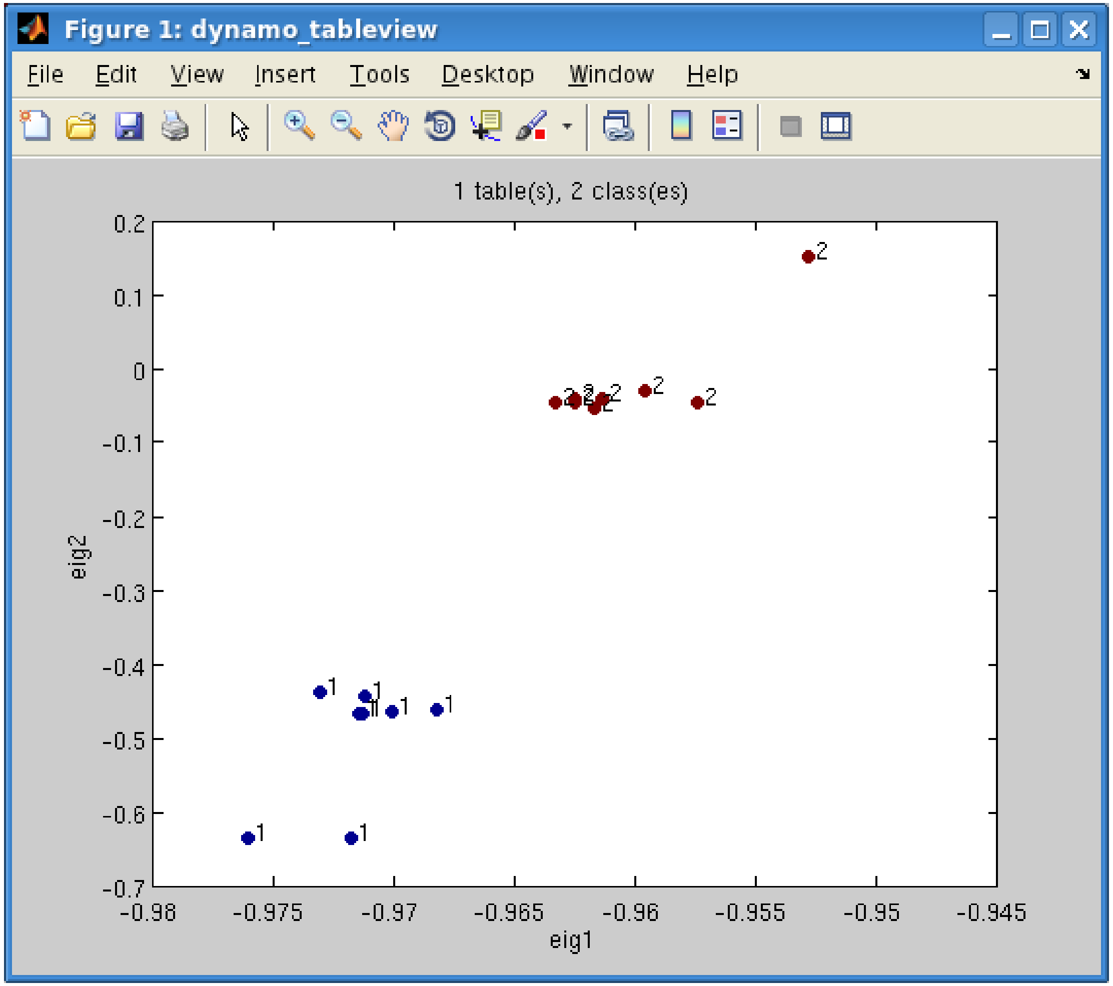
ect subset selected

We draw the result in another figure for clarity...

Looks good...

Apparently, the two first eigenvalues of our analysis catch the -synthetically Generated- differencing features of the data set:

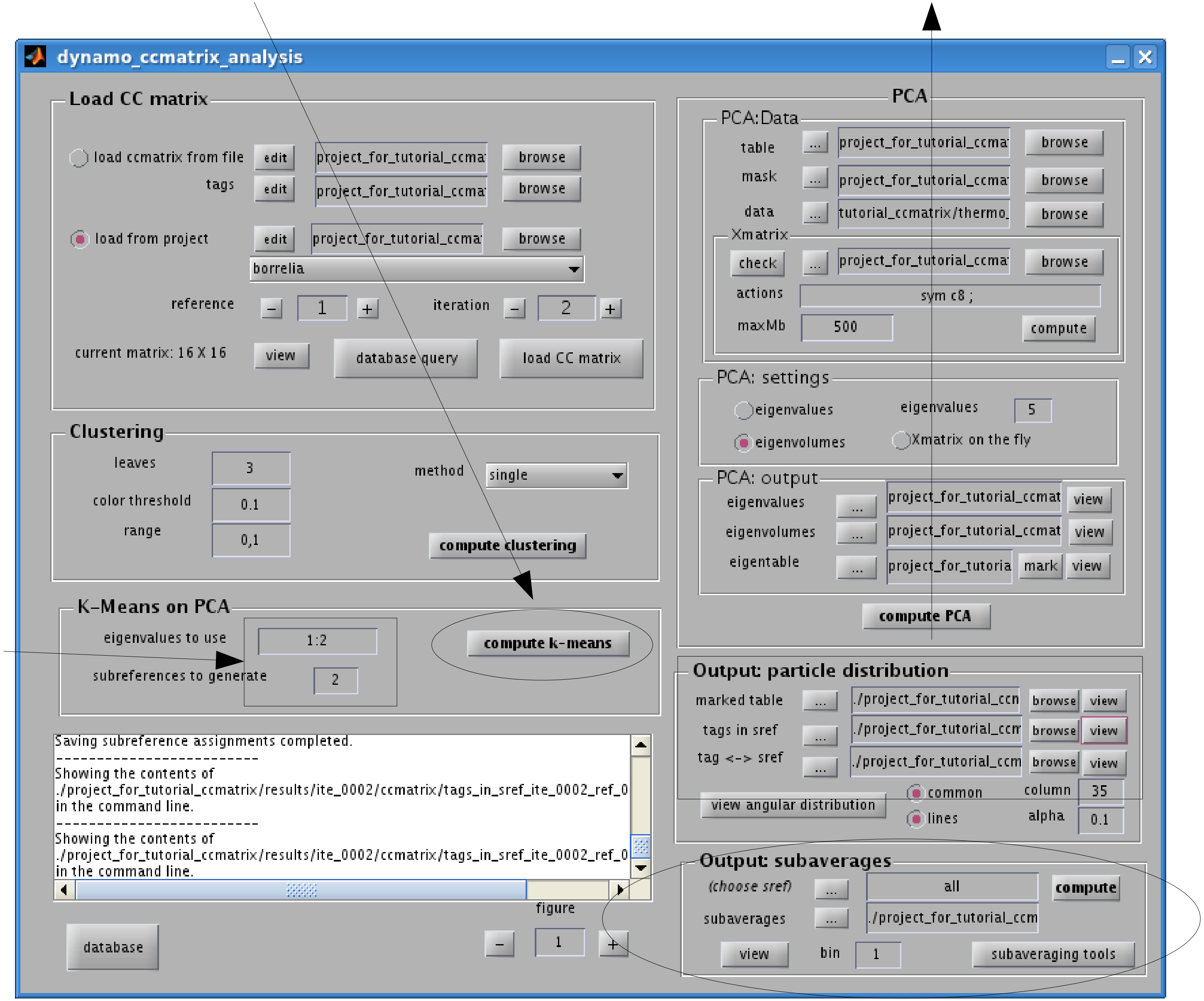
One cluster gets painted in blue (labeled “1” during the genaration of the data set),
The other cluster gets painted in red (and includes all the particles labeled as “2”)



So the “cheat” in column 22 tells us that the analysis will be right...
In real life, the scatterplot already tells us that classification according to the first two eigenvalues will give us a good separation. But we still wouldn't know if the induced separation has a physical meaning.

So, we compute the classification...

... oputput is generated to the same area as with clustering
Here the text files with the classification graph...



dynamo_ccmatrix_analysis

Load CC matrix

○ load ccmatrix from file: project_for_tutorial_ccma
tags: project_for_tutorial_ccma
● load from project: project_for_tutorial_ccma
borrelia
reference: 1
iteration: 2
current matrix: 16 X 16

Clustering

leaves: 3
color threshold: 0.1
range: 0,1
method: single

K-Means on PCA

eigenvalues to use: 1:2
subreferences to generate: 2

```
Saving subreference assignments completed.
-------------------------
Showing the contents of
./project_for_tutorial_ccmatrix/results/ite_0002/ccmatrix/tags_in_sref_ite_0002_ref_0
in the command line.
-------------------------
Showing the contents of
./project_for_tutorial_ccmatrix/results/ite_0002/ccmatrix/tags_in_sref_ite_0002_ref_0
in the command line.
```

figure: 1

PCA

PCA:Data
table: project_for_tutorial_ccma
mask: project_for_tutorial_ccma
data: tutorial_ccmatrix/thermo_
Xmatrix: project_for_tutorial_ccma
actions: sym c8 ;
maxMb: 500

PCA: settings
○ eigenvalues
eigenvalues: 5
● eigenvolumes
○ Xmatrix on the fly

PCA: output
eigenvalues: project_for_tutorial_ccmat
eigenvolumes: project_for_tutorial_ccmat
eigentable: project_for_tutoria

Output particle distribution

marked table: ./project_for_tutorial_ccn
tags in sref: ./project_for_tutorial_ccm
tag <-> sref: ./project_for_tutorial_ccm
● common
● lines
column: 35
alpha: 0.1

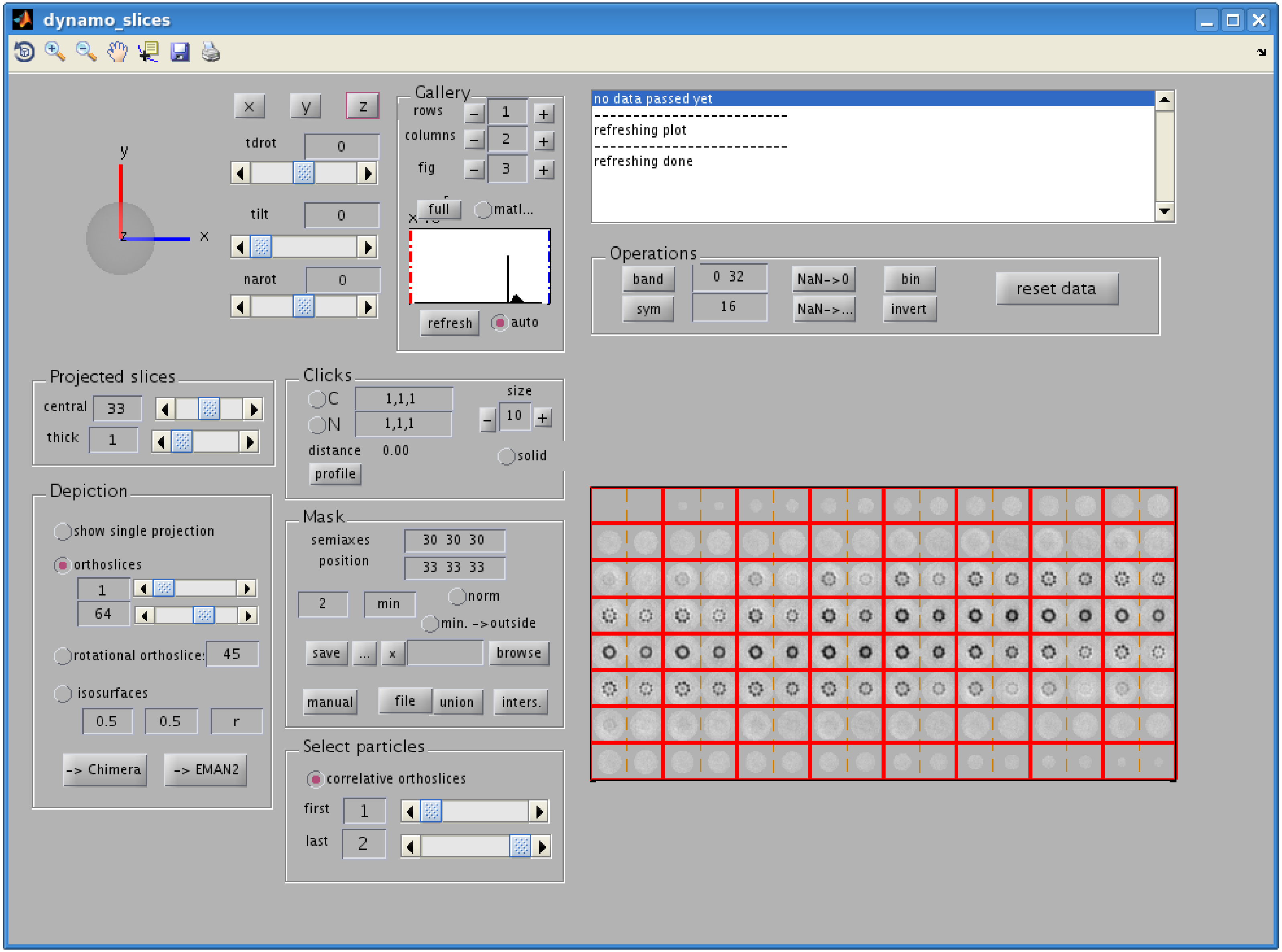
Output subaverages

(choose sref): all
subaverages: ./project_for_tutorial_ccm
bin: 1

with these settings

... and here we command the production and depiction of subaverages...

The [view] pushbutton should give you a window similar to this one:



dynamo_slices

```
no data passed yet
---------------------------
refreshing plot
---------------------------
refreshing done
```

tdrot 0

tilt 0

narot 0

Gallery
rows 1
columns 2
fig 3

○ matl... ● auto

Operations
0 32
16

Projected slices
central 33
thick 1

Clicks
○ C 1,1,1
○ N 1,1,1
distance 0.00
size 10
○ solid

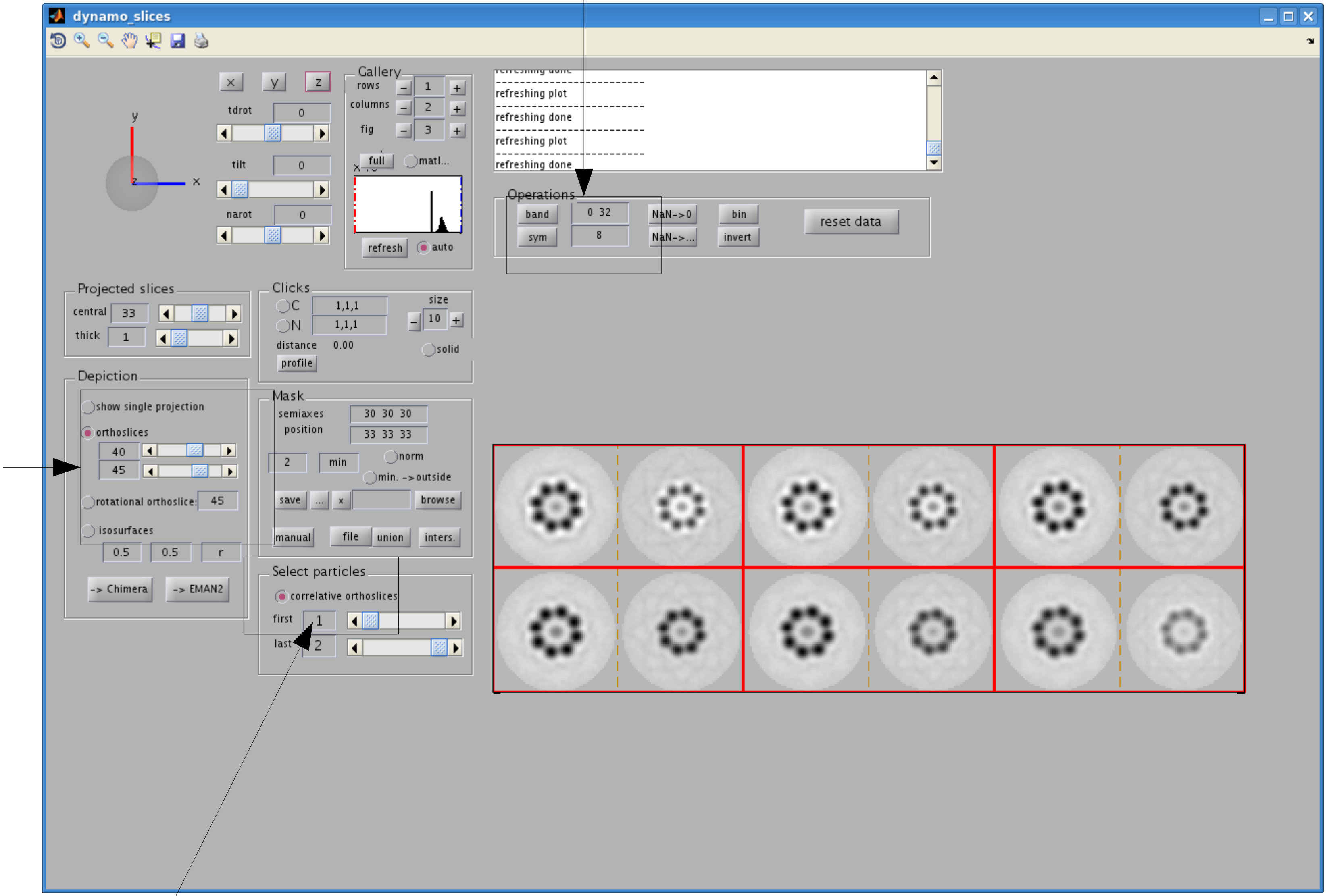
Depiction
○ show single projection
● orthoslices 1 64
○ rotational orthoslice: 45
○ isosurfaces 0.5 0.5 r

Mask
semiaxes 30 30 30
position 33 33 33
2
○ norm
○ min. ->outside

Select particles
● correlative orthoslices
first 1
last 2

Apply a C8 symmetry to increase the signal quality

Perhaps select
a smaller range
of z-slices
(e.g. 40 to 45)



```
refreshing plot
------------------------------
refreshing done
------------------------------
refreshing plot
------------------------------
refreshing done
```

Use the “correlative orthoslices” option
to see corresponding slices of the two averages
side to side

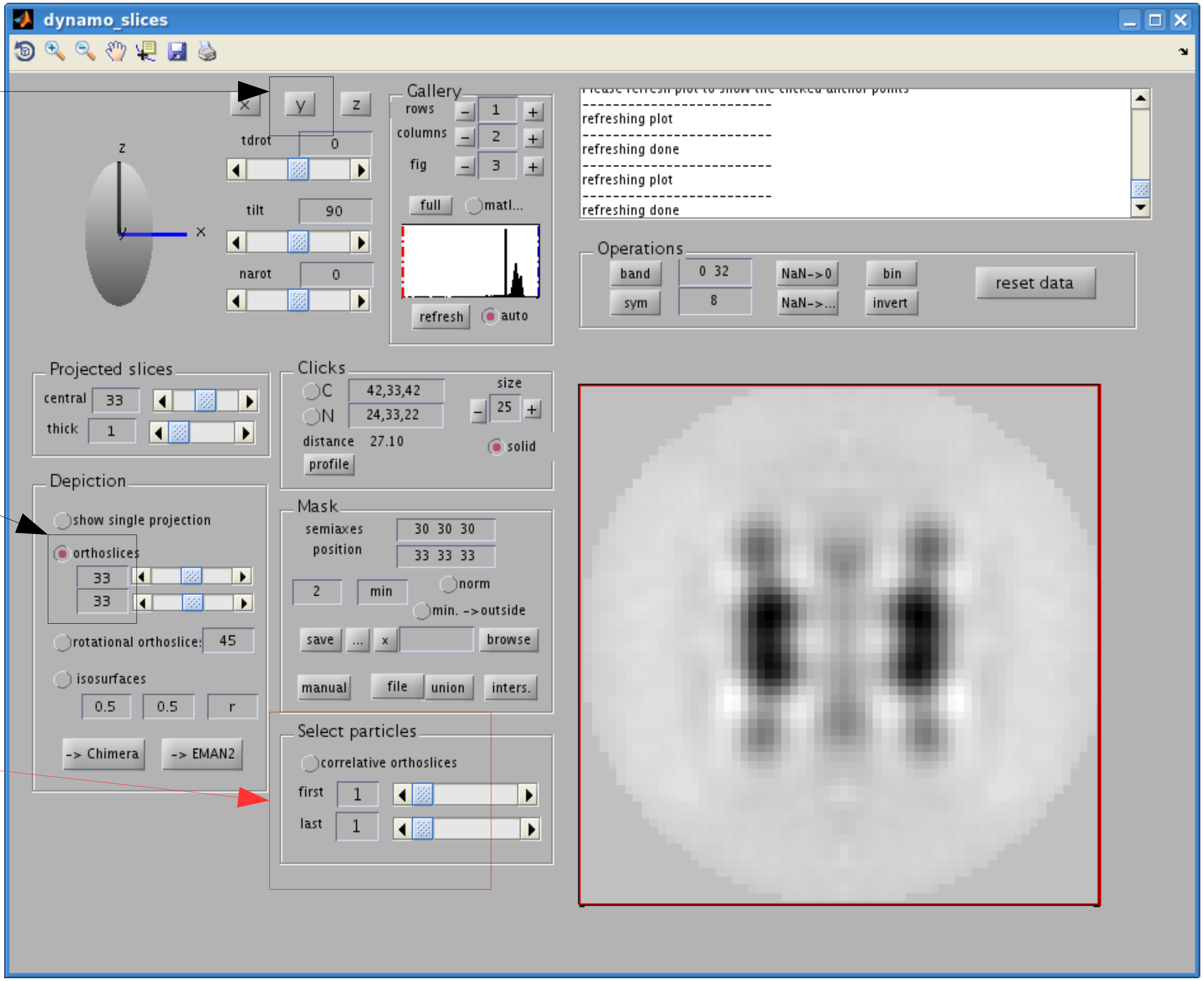
Now it becomes clear how the data set was modeled
the two “classes” arising in the PCA
correspond merely to different magnification

We can confirm our visual impression by measuring distances on screen:

choose the “y” view
(to measure on the
diagonal)

Select a single orthoslice
(across the center: 33 to 33)

Switch off the correlative
orthoslice option and choose
the first average



dynamo_slices

```
Please refresh plot to show the clicked anchor points
-------------------------
refreshing plot
-------------------------
refreshing done
-------------------------
refreshing plot
-------------------------
refreshing done
```

tdrot 0
tilt 90
narot 0

Gallery
rows 1
columns 2
fig 3
○ matl...
● auto

Operations
band 0 32
sym 8

Projected slices
central 33
thick 1

Clicks
○ C 42,33,42
○ N 24,33,22
size 25
distance 27.10
● solid

Depiction
○ show single projection
● orthoslices 33 33
○ rotational orthoslice: 45
○ isosurfaces 0.5 0.5 r

Mask
semiaxes 30 30 30
position 33 33 33
2 min
○ norm
○ min. ->outside

Select particles
○ correlative orthoslices
first 1
last 1

Measure the distance between two corners:

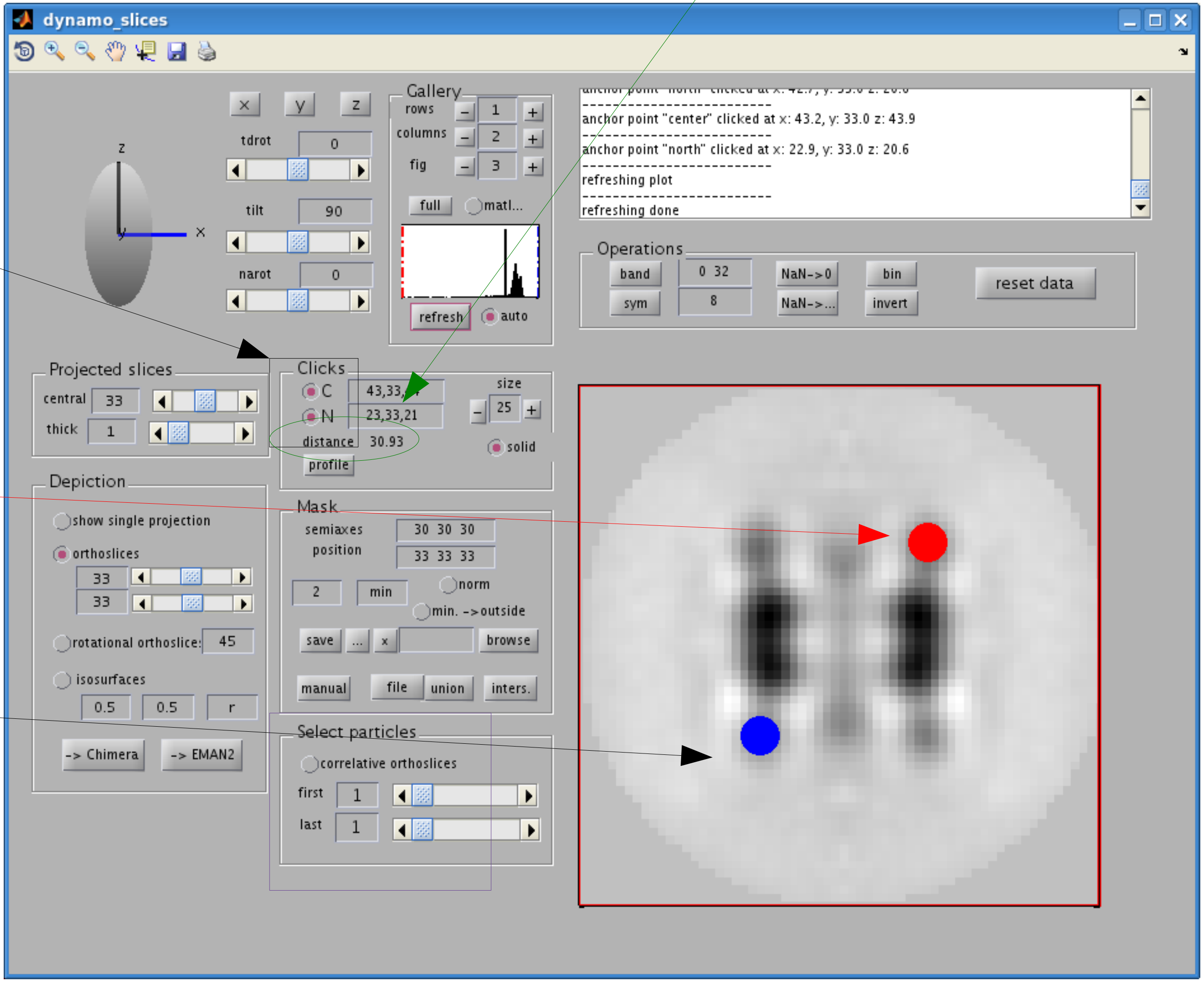
(clicking on the darkest spots)

You should get something around 30'9

Switch on the clicking tools

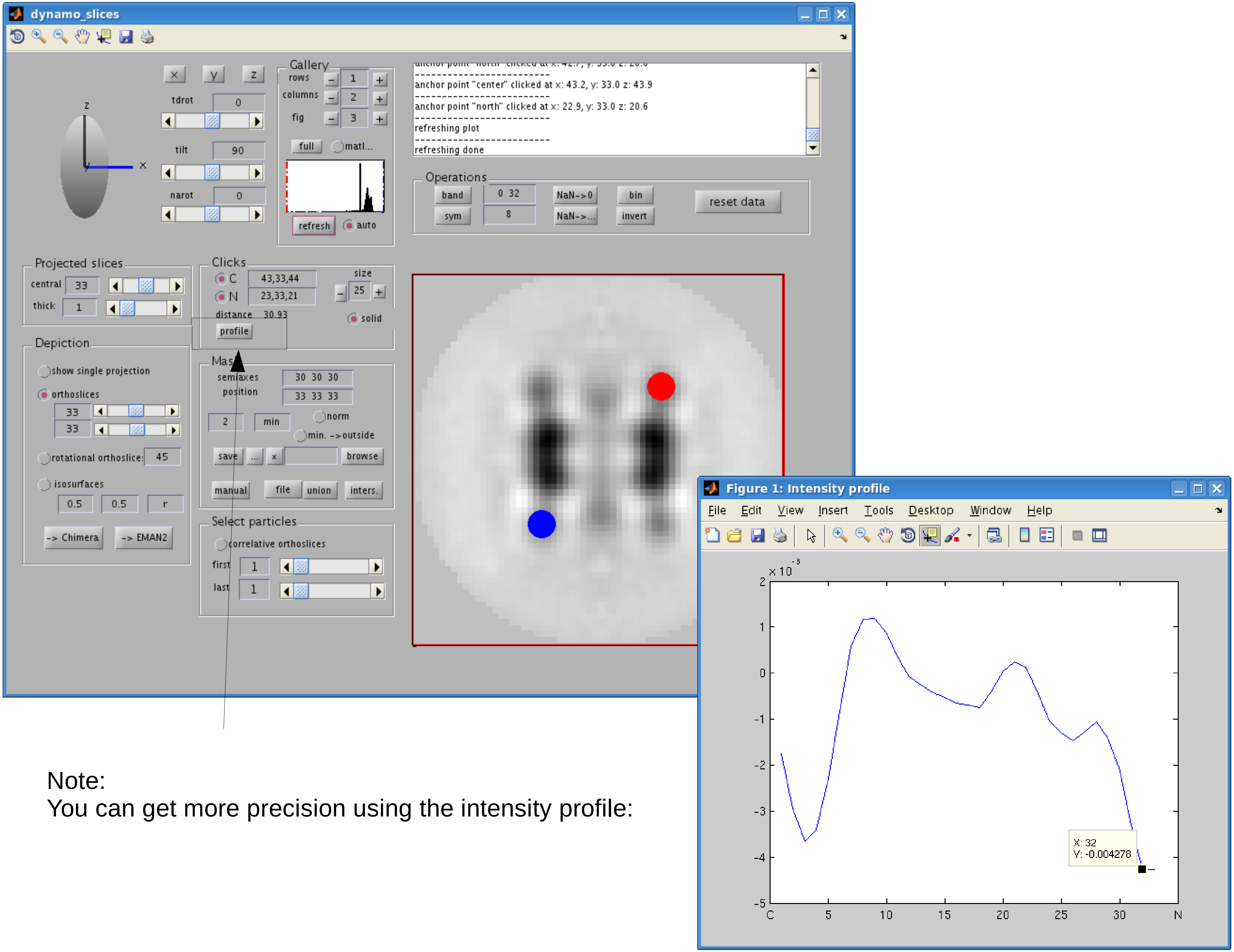
Left click:
Places red point on screen
(labeled “C(enter)” clicker)

Right click:
Places blue point on screen
(labeled “N(orth)” clicker)



```
anchor point "north" clicked at x: 42.7, y: 33.0 z: 20.6
---------------------------
anchor point "center" clicked at x: 43.2, y: 33.0 z: 43.9
---------------------------
anchor point "north" clicked at x: 22.9, y: 33.0 z: 20.6
---------------------------
refreshing plot
---------------------------
refreshing done
```

Clicks
- ● C 43,33,4
- ● N 23,33,21
- distance 30.93



```
---------------------------
anchor point "center" clicked at x: 43.2, y: 33.0 z: 43.9
---------------------------
anchor point "north" clicked at x: 22.9, y: 33.0 z: 20.6
---------------------------
refreshing plot
---------------------------
refreshing done
```

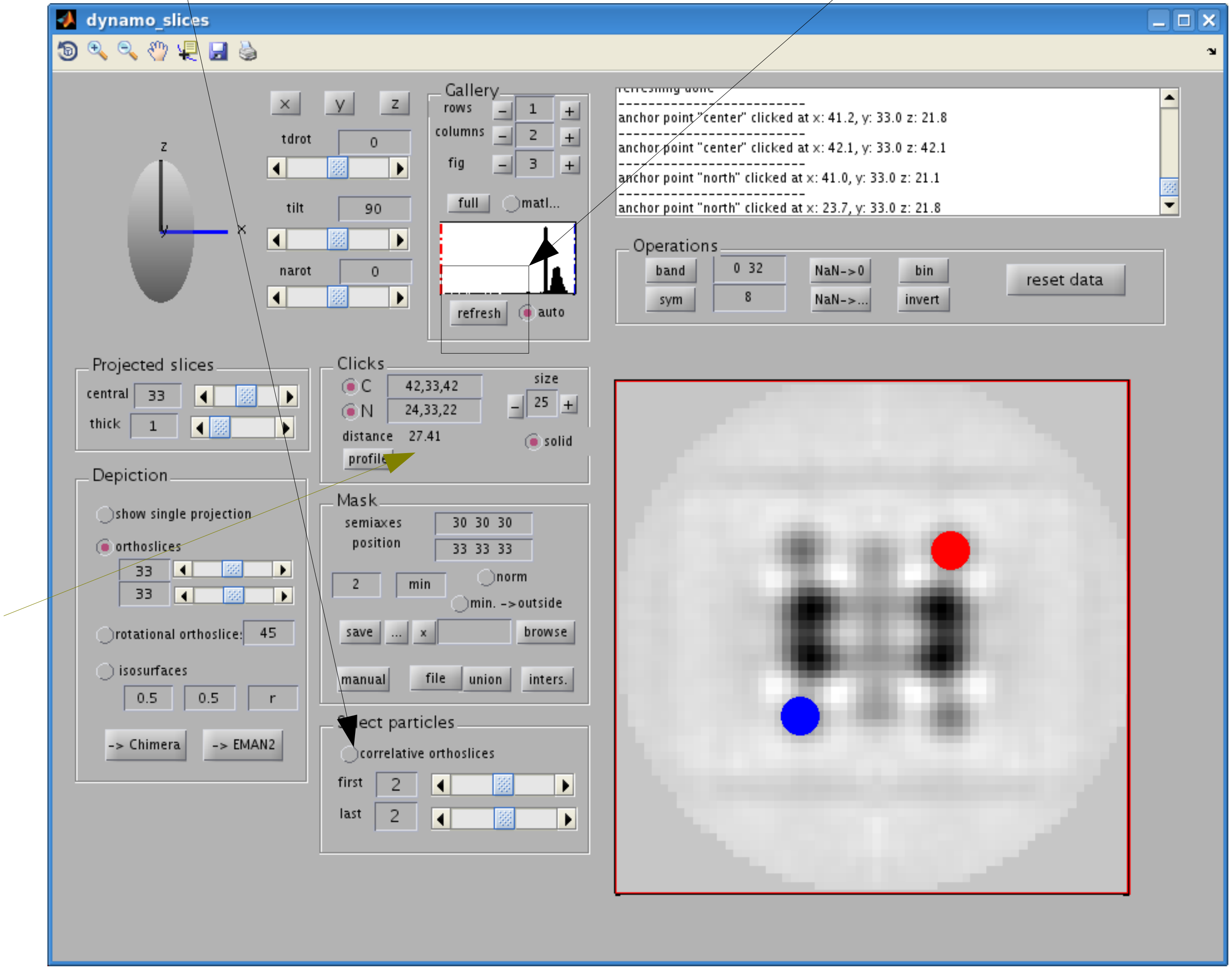
Note:
You can get more precision using the intensity profile:

Now we do the measurement for the second average

We probably need to refresh the screen

We place again the
red and the blue markers...

... and the distance now
should be around 28 pixels



dynamo_slices

```
refreshing done
---------------------------
anchor point "center" clicked at x: 41.2, y: 33.0 z: 21.8
---------------------------
anchor point "center" clicked at x: 42.1, y: 33.0 z: 42.1
---------------------------
anchor point "north" clicked at x: 41.0, y: 33.0 z: 21.1
---------------------------
anchor point "north" clicked at x: 23.7, y: 33.0 z: 21.8
```

tdrot 0

tilt 90

narot 0

Gallery: rows 1, columns 2, fig 3

Projected slices: central 33, thick 1

Depiction: ● orthoslices 33, 33; rotational orthoslice: 45; isosurfaces 0.5 0.5 r

Clicks: ● C 42,33,42; ● N 24,33,22; distance 27.41; size 25; ● solid

Mask: semiaxes 30 30 30; position 33 33 33; 2

Operations: 0 32; 8

Select particles: first 2; last 2

.... and this actually corresponds to how the data set was modeled:
it comprises random rotations of two templates of same molecule, one scaled to the 90%.
This toy classification example by PCA just recovered this scaling factor.