Dynamo

icosahedral subboxing and multireference

Main Idea

We will produce a data set of icosahedral viruses.
Each capsid will have artificially generated insertions in a random number of its vertices, I.e, we create a population of heterogeneous vertices (with and without insertion).

Thus we are not interested in aligning and classifying capsids themselves (as they can be of many different structural patterns), but in extracting the vertices and classifying them separately.

We will visit the following concepts:

- Icosahedral symmetry
- Subboxing
- PCA + Kmeans classification
- Effect of masks on PCA
- Multireference alignment
- Creation of MRA projects from command lie
- “Dynamo Wizard” interface

- Create a tutorial data set
  Illustrates the main features of the synthetic data
  - A set of of icosahedral viruses:
  - An artificial “ insertion” is added in some vertices of some copies: Heterogeneity.

- Create a noisy tutorial data set
    - Vertices are “subboxed”: a new data set is created, so that each particle is one of the vertices of the original viruses.

- Create a project for multireference alignment
  - The project will align the vertices to several randomly created references.

- Analyze results from a multireference alignment
  - Retrieve results from a multireference project (averages, tables).
  - Scan which particles are in which reference at which iteration.

- Modify a project
    - Experiment with different numerical parameters

We start creating a tutorial folder with all the synthetic data that we need.

folder that will contain all created files

Number of created capsidae

each particle can have a random number of inseritons,
each one located in a random position in the virus

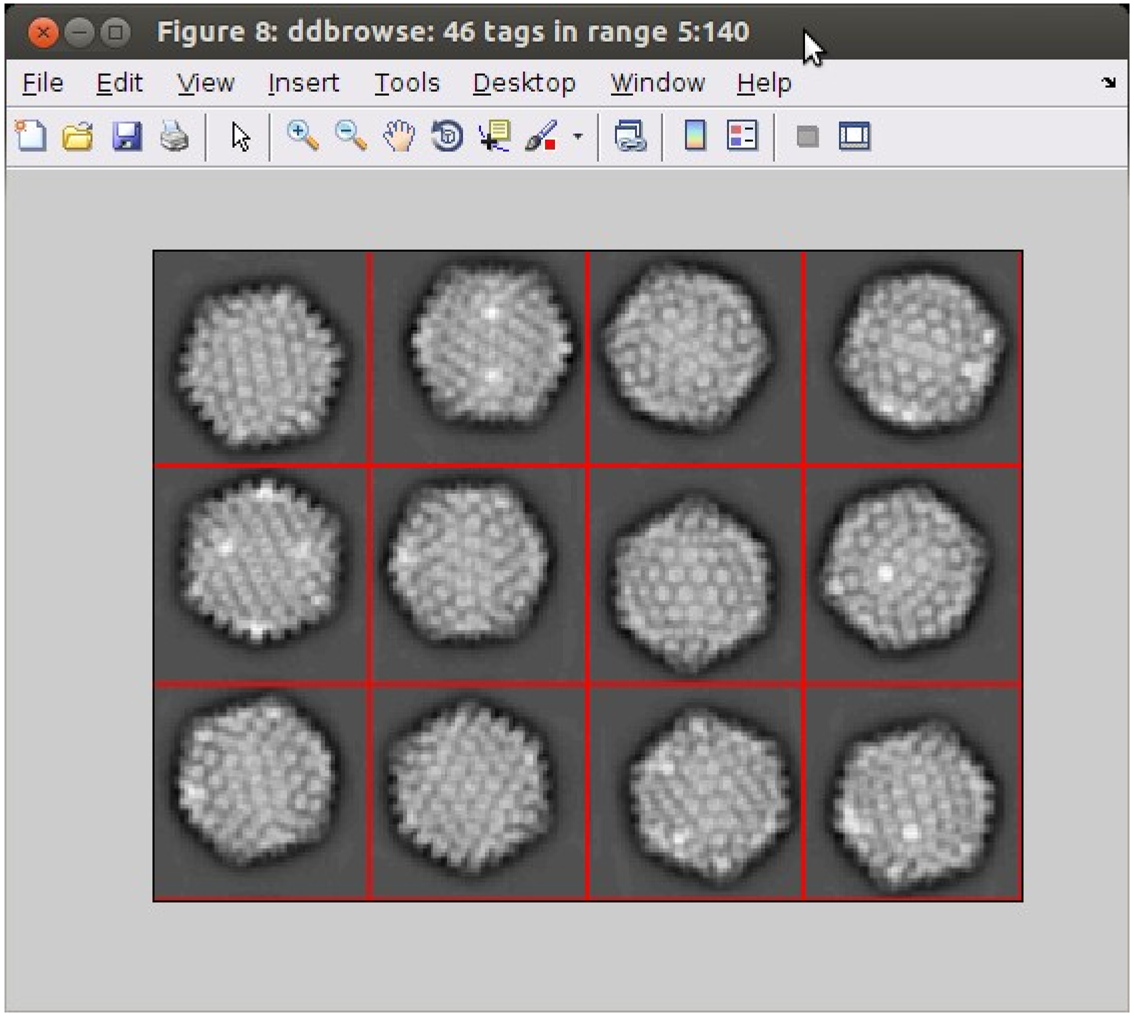
```
dpktut.herpes.tutorial('nonoise','M',12,'maximumInsertions',6,'noise',0,....
'strength',5, 'subbox',false)
```

we want to do the subboxing (cropping of the vertices) manually

intensity of the synthetic insertions

Let us check what we have

```
>> dslices nonoise/data -jz 0
```



Each data particle is a copy of a virus with a random number of vertices occupied by an “ insertion”.
There is no “unique” repeating structure at the scale of the virus.
The structure that is repeating are the two version of the vertex (with and without the insertion).

This is exactly the task of *subboxing*:

If we know:

- The shifts and rotations of the particles in relation to an average or an ideal template.
  (I. e.: we know the table of the virus data set)

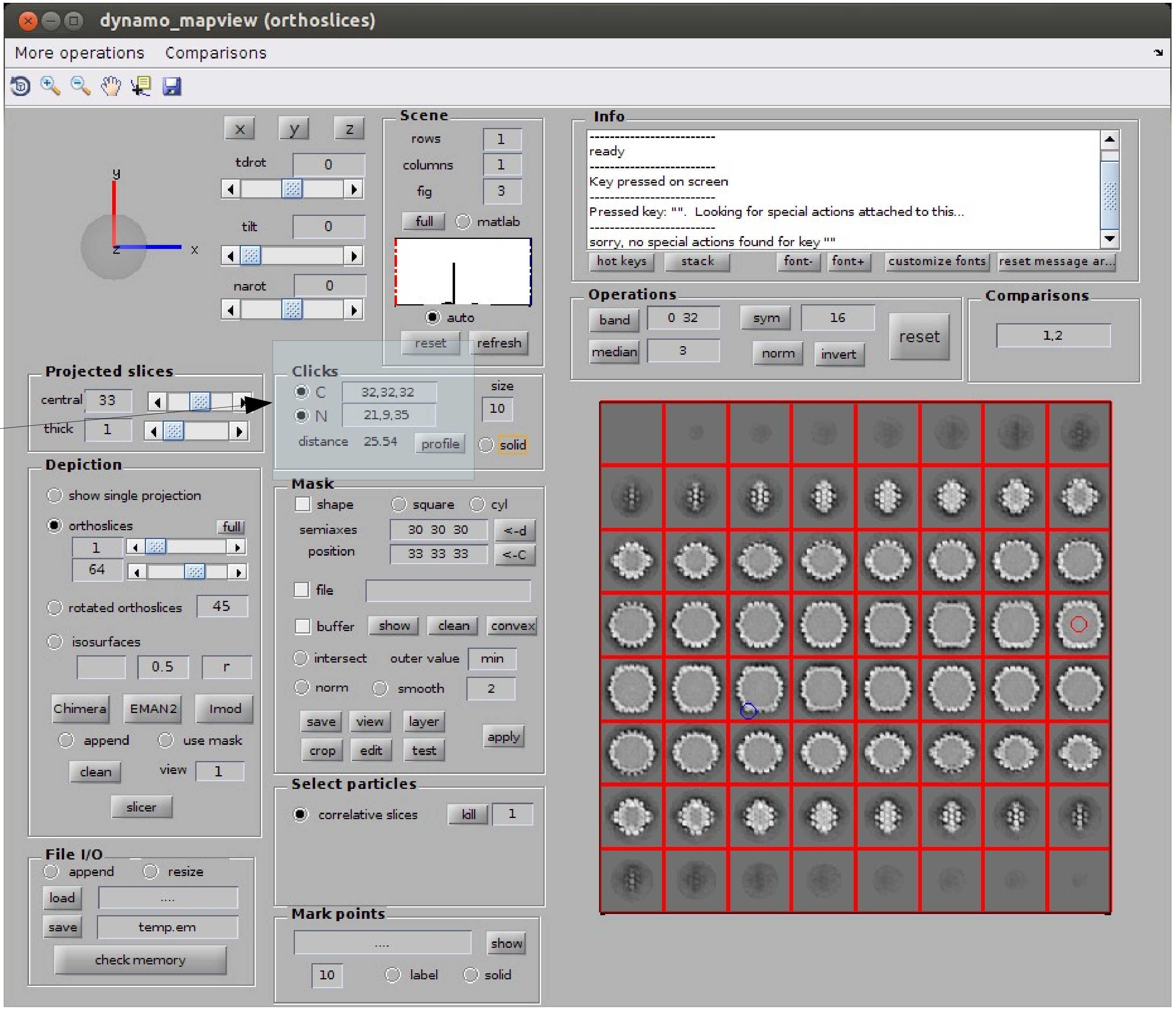
- The position of an asymmetric unit in the ideal template.

- The symmetry operations that link the different asymmetric units that we want to analyze separately.

We just need to measure the distance from a vertex to the center of the virus:

As suggested by the tutorial, we depict the average of the virus particles:

```
dmapview nonoise/realAverage.em
```



Clicks: C 32,32,32; N 21,9,35; distance 25.54; size 10

Projected slices: central 33; thick 1

Scene: rows 1; columns 1; fig 3

Operations: band 0 32; sym 16; median 3

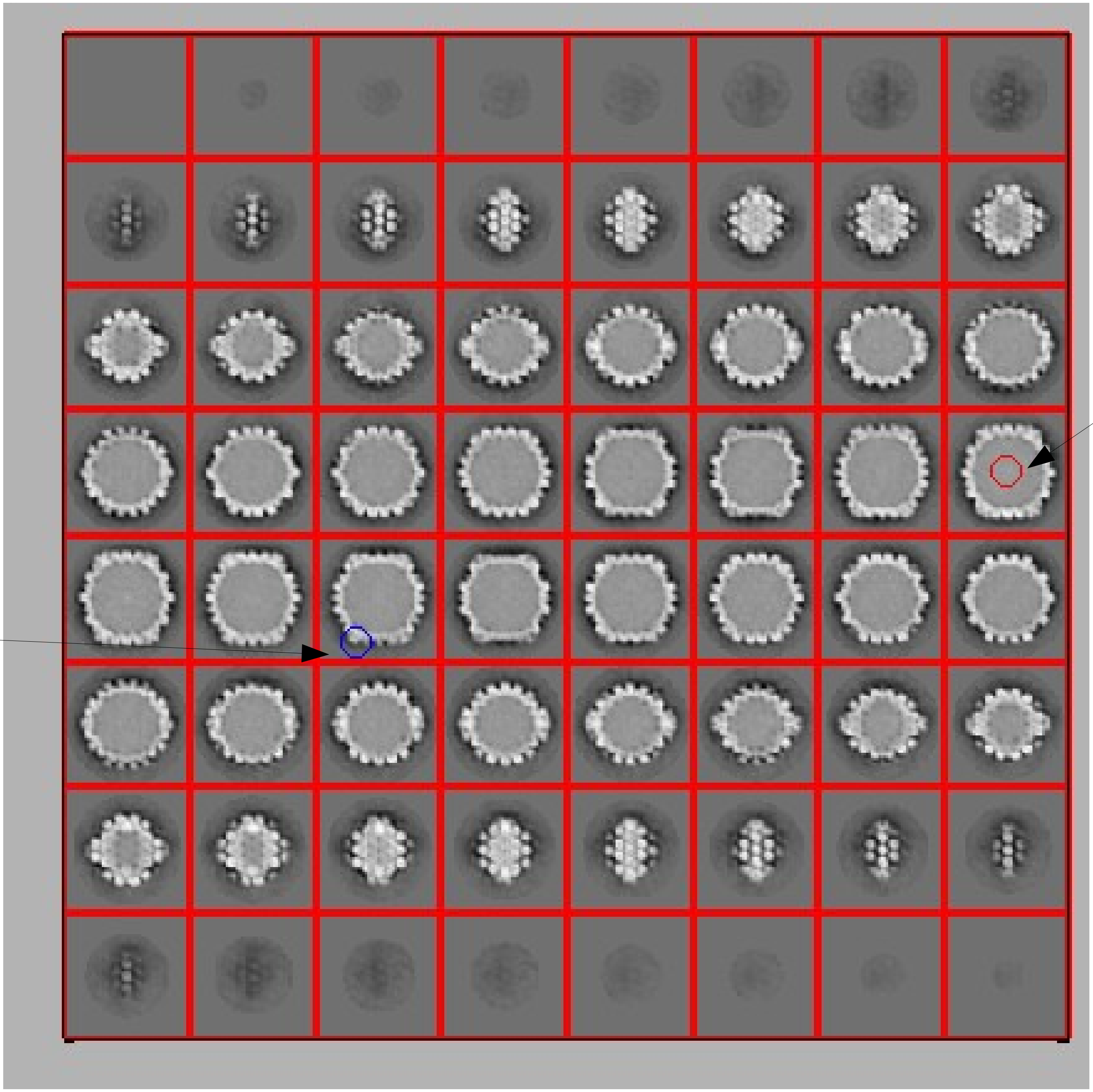
Comparisons: 1,2

Mask: semiaxes 30 30 30; position 33 33 33

Info:

```
-------------------------
ready
-------------------------
Key pressed on screen
-------------------------
Pressed key: "".  Looking for special actions attached to this...
-------------------------
sorry, no special actions found for key ""
```

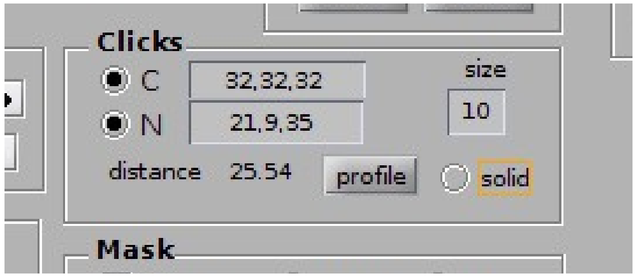
we activate the option of clicking two points (called 'C' for center and 'N' for north);



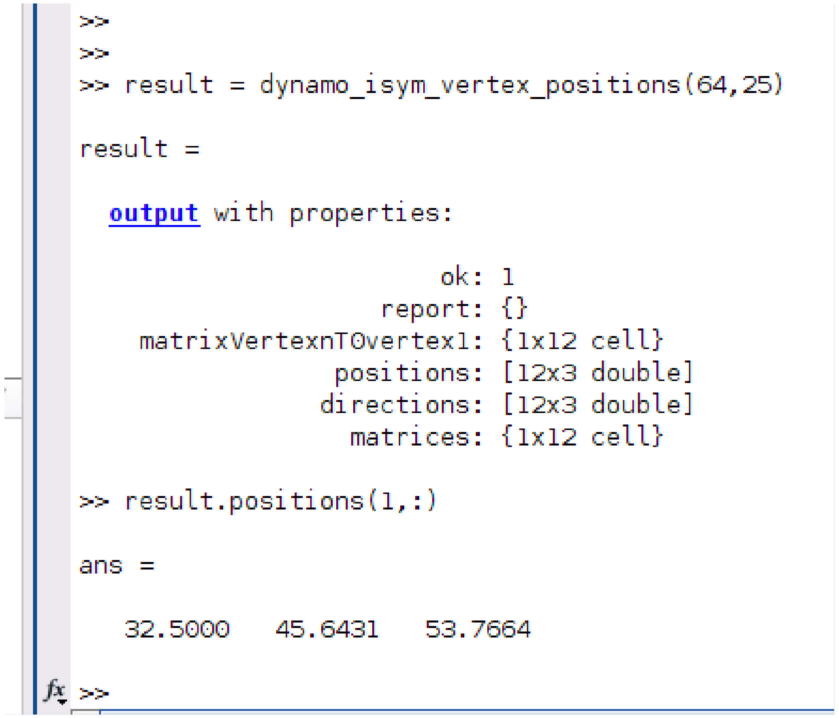
use [C] to choose
 the center

use [N] to click
on one vertex

you can read the distance in the [Clicks] Panel of `mapview`



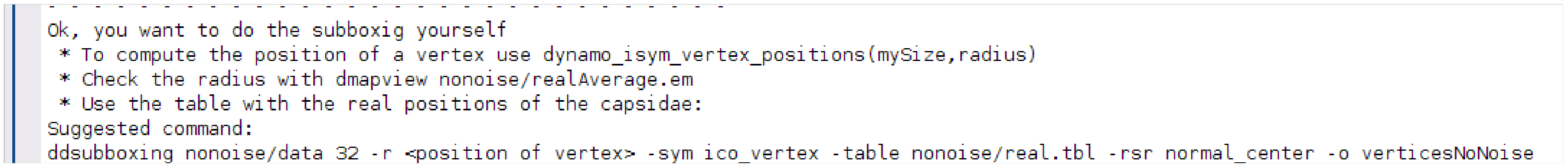
Actually we also have a read of the coordinates of an asymmetric unit (the N marker), but they have been picked rather arbitrarily.
A better option is to use the computed distance from the unit to the center (~25) and compute automatically the exact symmetrically determined position of the unit:



```
>>
>>
>> result = dynamo_isym_vertex_positions(64,25)

result =

  output with properties:

                     ok: 1
                 report: {}
    matrixVertexnTOvertex1: {1x12 cell}
              positions: [12x3 double]
             directions: [12x3 double]
               matrices: {1x12 cell}

>> result.positions(1,:)

ans =

   32.5000   45.6431   53.7664

>>
```

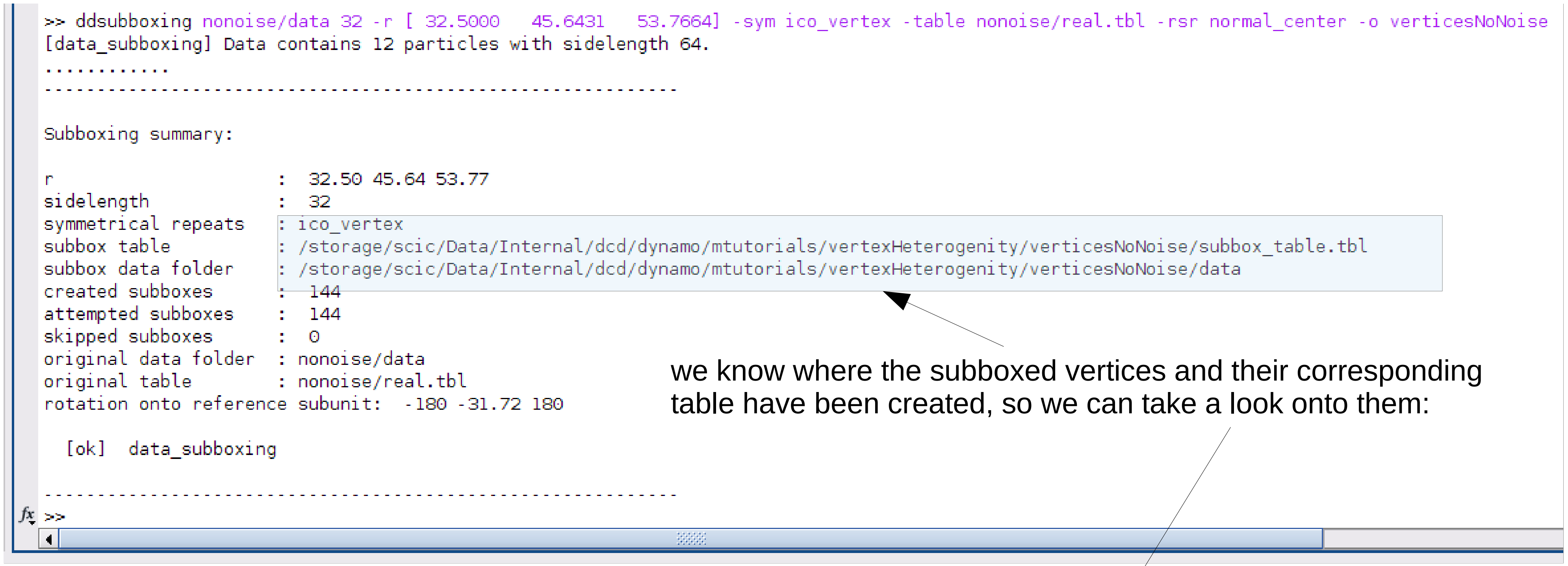
For the given sidelength and distance to the center, this is one of the symmetrically related positions in the capsid of a virus when it is aligned along the conventions of icosahedral symmetry!

now, with the position of the vertex, we can follow the suggestions of the tutorial....



```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Ok, you want to do the subboxig yourself
 * To compute the position of a vertex use dynamo_isym_vertex_positions(mySize,radius)
 * Check the radius with dmapview nonoise/realAverage.em
 * Use the table with the real positions of the capsidae:
Suggested command:
ddsubboxing nonoise/data 32 -r <position of vertex> -sym ico_vertex -table nonoise/real.tbl -rsr normal_center -o verticesNoNoise
```

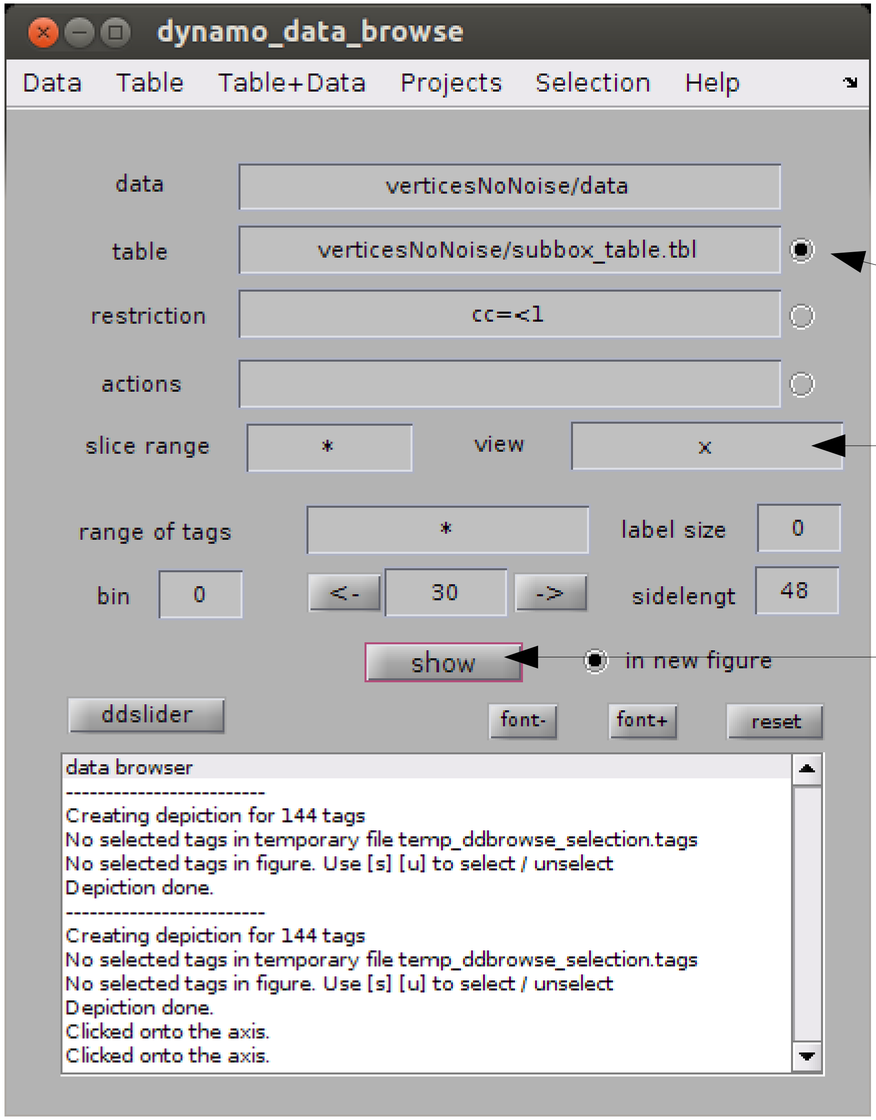
… and compute our subboxing:



```
>> ddsubboxing nonoise/data 32 -r [ 32.5000    45.6431    53.7664] -sym ico_vertex -table nonoise/real.tbl -rsr normal_center -o verticesNoNoise
[data_subboxing] Data contains 12 particles with sidelength 64.
............
------------------------------------------------------------

Subboxing summary:

r                      :  32.50 45.64 53.77
sidelength             :  32
symmetrical repeats    : ico_vertex
subbox table           : /storage/scic/Data/Internal/dcd/dynamo/mtutorials/vertexHeterogenity/verticesNoNoise/subbox_table.tbl
subbox data folder     : /storage/scic/Data/Internal/dcd/dynamo/mtutorials/vertexHeterogenity/verticesNoNoise/data
created subboxes       :  144
attempted subboxes     :  144
skipped subboxes       :  0
original data folder   : nonoise/data
original table         : nonoise/real.tbl
rotation onto reference subunit:  -180 -31.72 180

  [ok]  data_subboxing

------------------------------------------------------------
>>
```

we know where the subboxed vertices and their corresponding table have been created, so we can take a look onto them:

```
>> ddbrowse -d verticesNoNoise/data -t verticesNoNoise/subbox_table.tbl
```



data: verticesNoNoise/data

table: verticesNoNoise/subbox_table.tbl ●

restriction: cc=<1

actions:

slice range: *

view: x

range of tags: *

label size: 0

bin: 0

<- 30 ->

sidelengt: 48

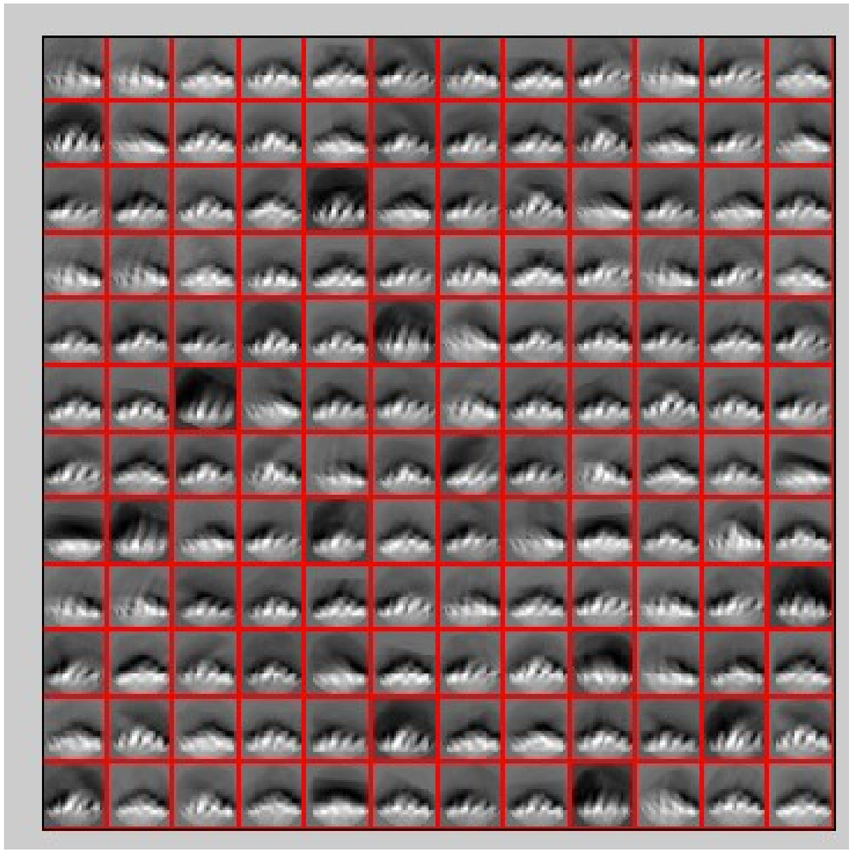
in new figure ●

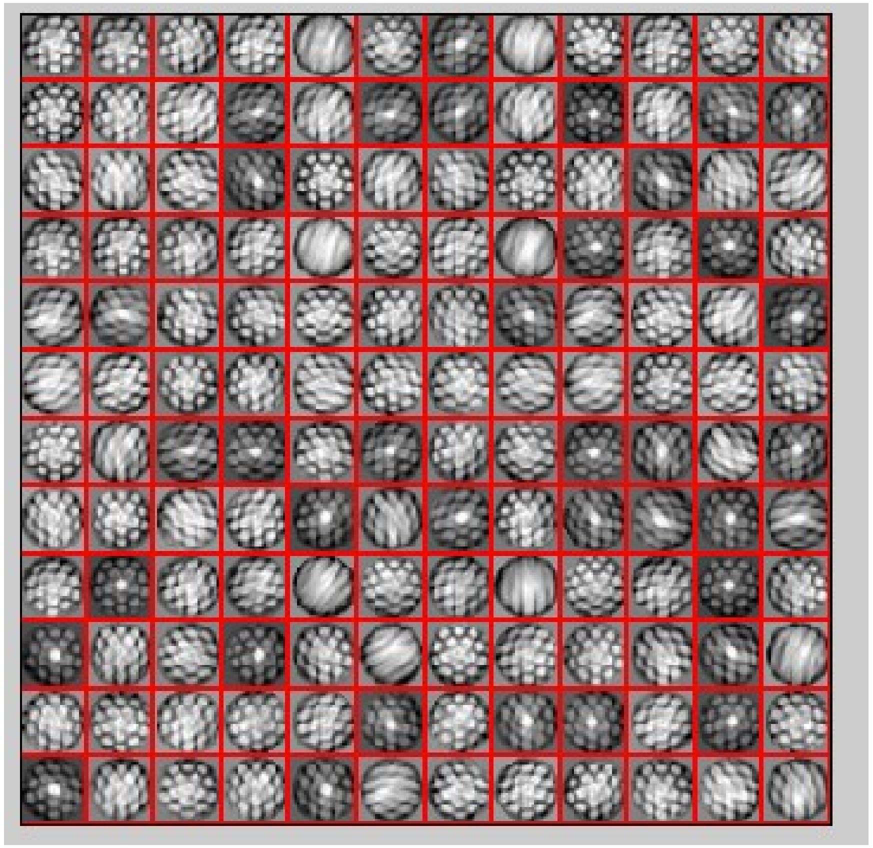
```
data browser
-------------------------
Creating depiction for 144 tags
No selected tags in temporary file temp_ddbrowse_selection.tags
No selected tags in figure. Use [s] [u] to select / unselect
Depiction done.
-------------------------
Creating depiction for 144 tags
No selected tags in temporary file temp_ddbrowse_selection.tags
No selected tags in figure. Use [s] [u] to select / unselect
Depiction done.
Clicked onto the axis.
Clicked onto the axis.
```

activate the table,
(otherwise the particles will not be aligned)

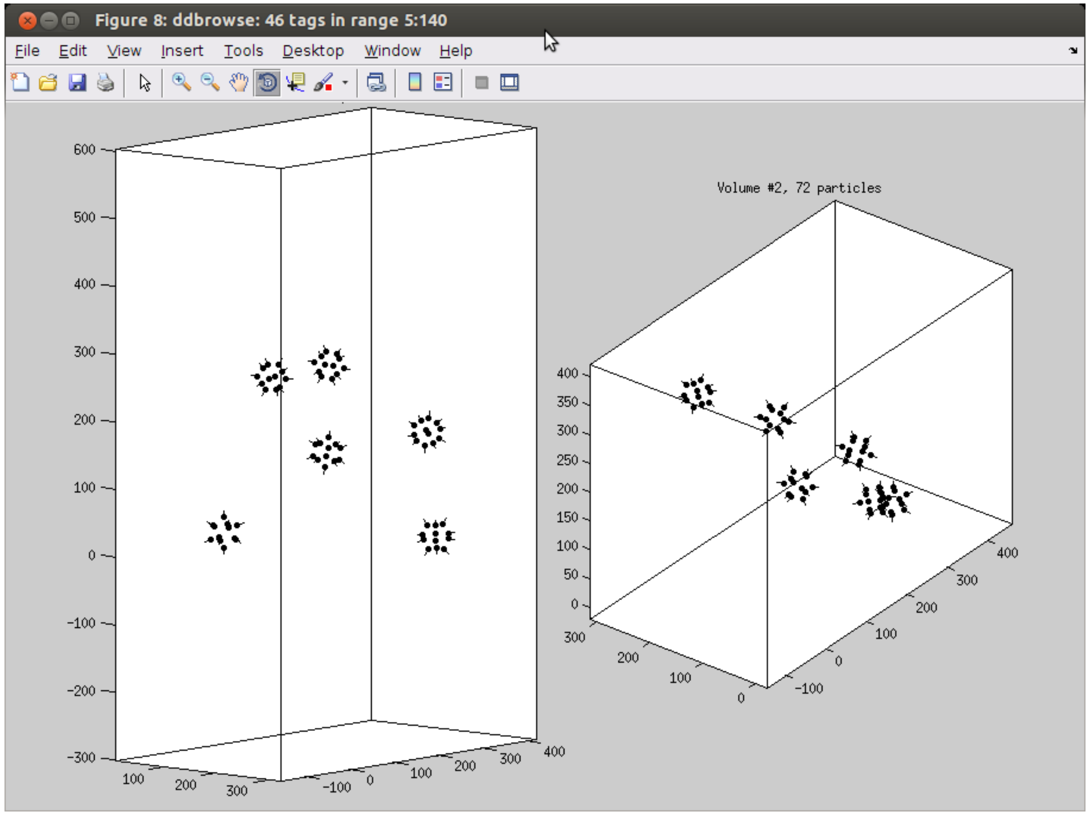
select a view

press to create the representation


x projection view of all subboxed vertex particles


y projection view of all subboxed vertex particles

```
dtplot verticesNoNoise/subbox_table.tbl -pf oriented_positions
```



Now, we repeat the synthetic data set with some more realistic noise:

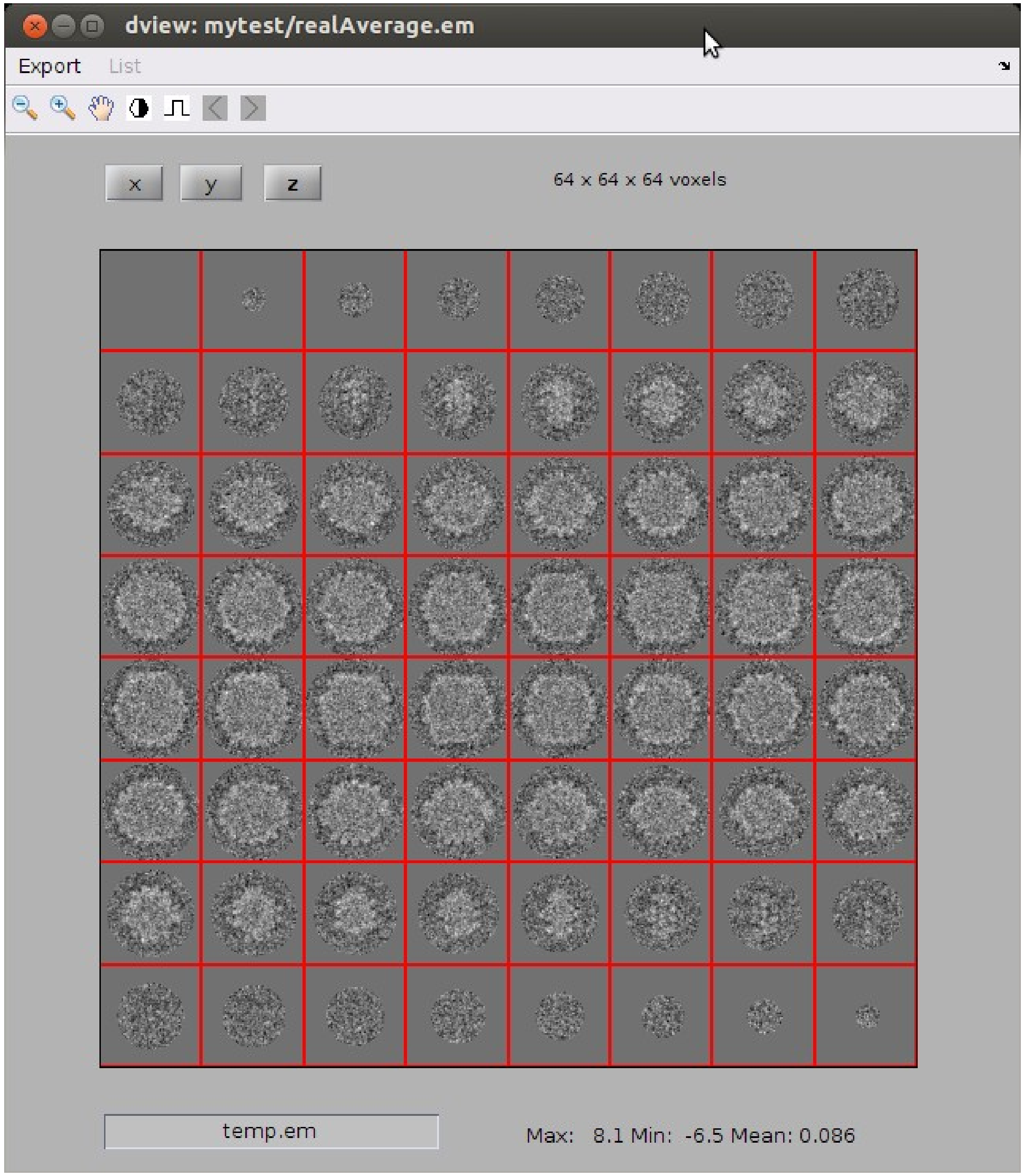
```
dpktut.herpes.tutorial('mytest','M',12,'maximumInsertions',6,....
    'sidelengthVertex',32,'noise',8,'strength',5,'subbox',true);
```

Now, we repeat the synthetic data set with some more realistic noise:

```
dpktut.herpes.tutorial('mytest','M',12,'maximumInsertions',6,....
    'sidelengthVertex',32,'noise',8,'strength',5);
```

This time we also let the tutorial to create a subboxing

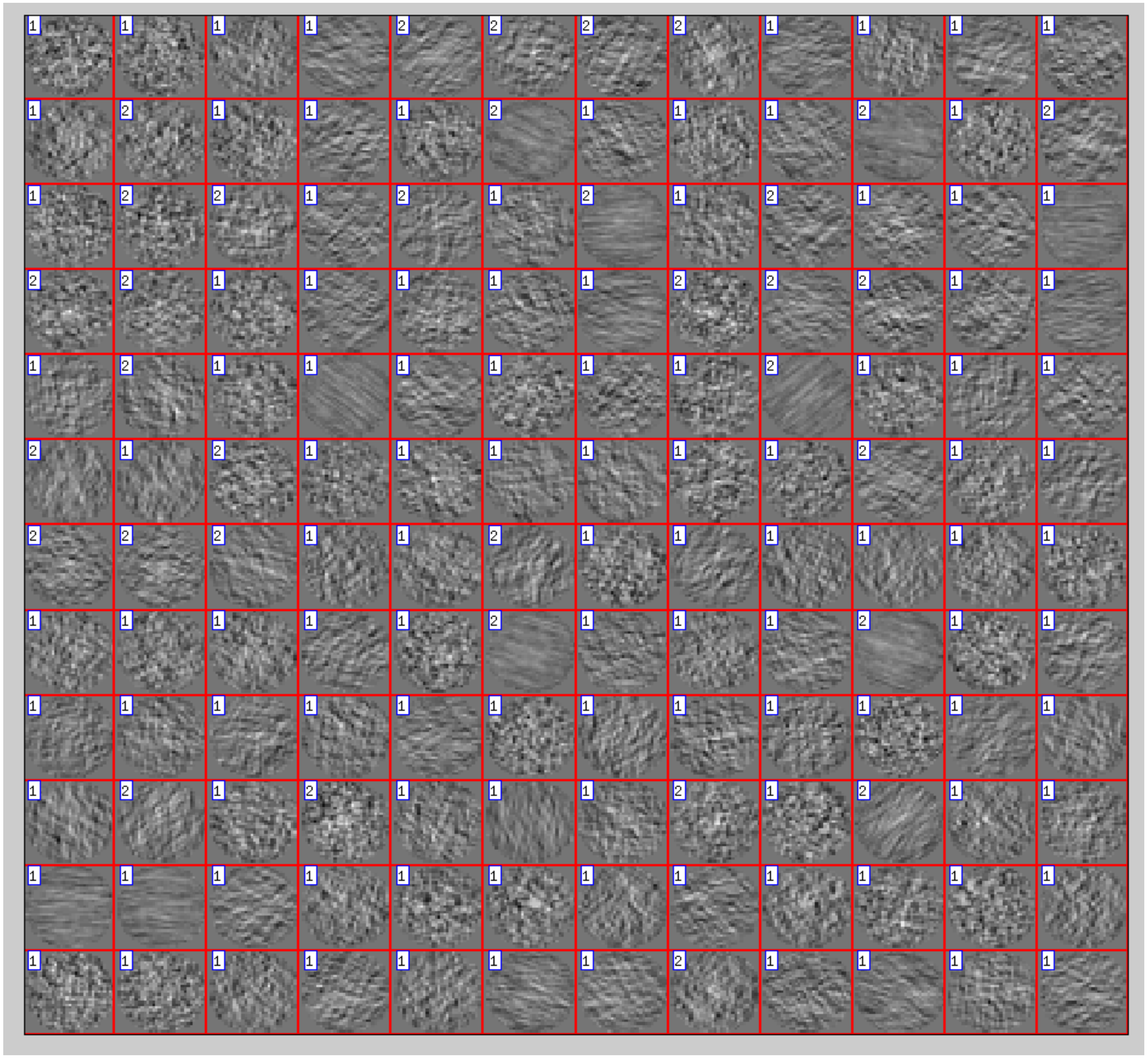
```
dview mytest/realAverage.em
```



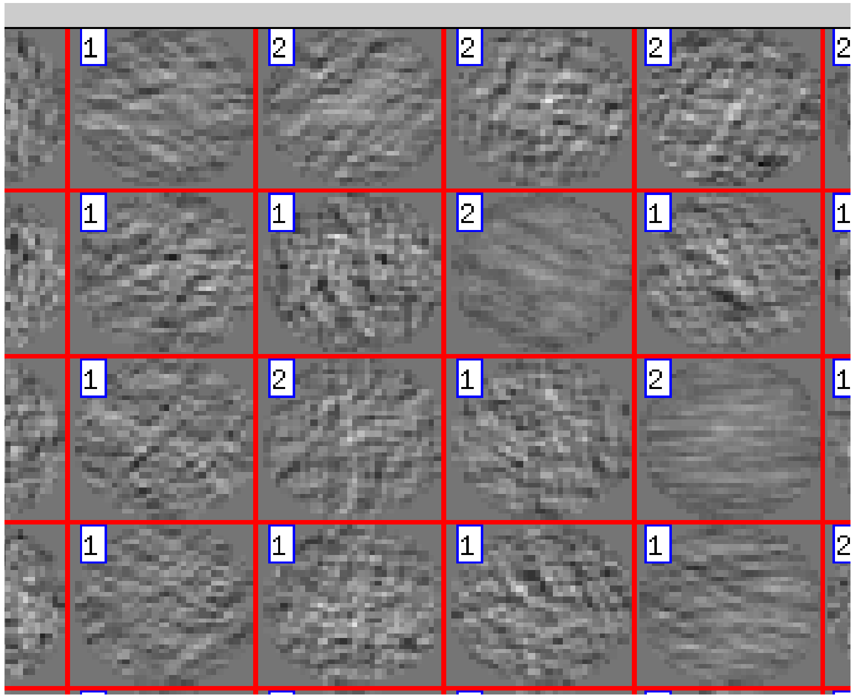
64 x 64 x 64 voxels

Max: 8.1 Min: -6.5 Mean: 0.086

```
dslices  mytest/vertices/data z -t mytest/vertices/subbox_table.tbl -otf 1 -align 1 -l class
```



Does not really look like anything!



It is difficult to judge if class 2 (vertices with “insertion”) look only richer in mass than particles on class 1.

The question now is if there is signal at all in the synthetic data set.

What would be the best outcome from an optimal algorithm?

Let's see what happens if we use our a priori knowledge about:

- Orientations and shifts (i.e. we know the table)
- class membership

to average all the particles belonging to one class

```
daverage mytest/vertices/data -t mytest/vertices/subbox_table.tbl ….
 -tr class=1 -ws av1;
```

Also, we can write it for class 2

```
 daverage mytest/vertices/data -t mytest/vertices/subbox_table.tbl ….
  -tr class=2 -ws av2;
```

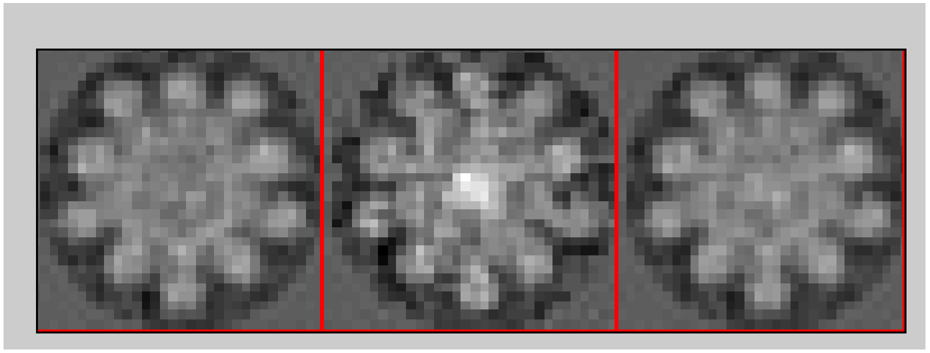
and for all vertex particles together

```
daverage mytest/vertices/data -t mytest/vertices/subbox_table.tbl -ws av;
```

We can keep the computed average as a file:

```
dwrite(av.average,'fullAverage.em');
```

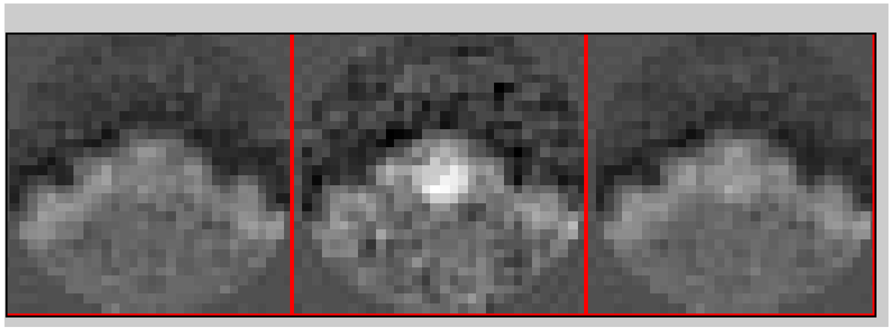
```
dslices({av1,av2,av},'jz',14:15,'dim',[1,3]);
```



The average of class 2 shows indeed the extra density.

The average of all particles taken together blurs the moiety out.

```
dslices({av1,av2,av},'jx',12:18,'dim',[1,3]);
```



PART II

Principal Component Analysis

We try to classify the subboxed, noisy vertices using PCA + kmeans

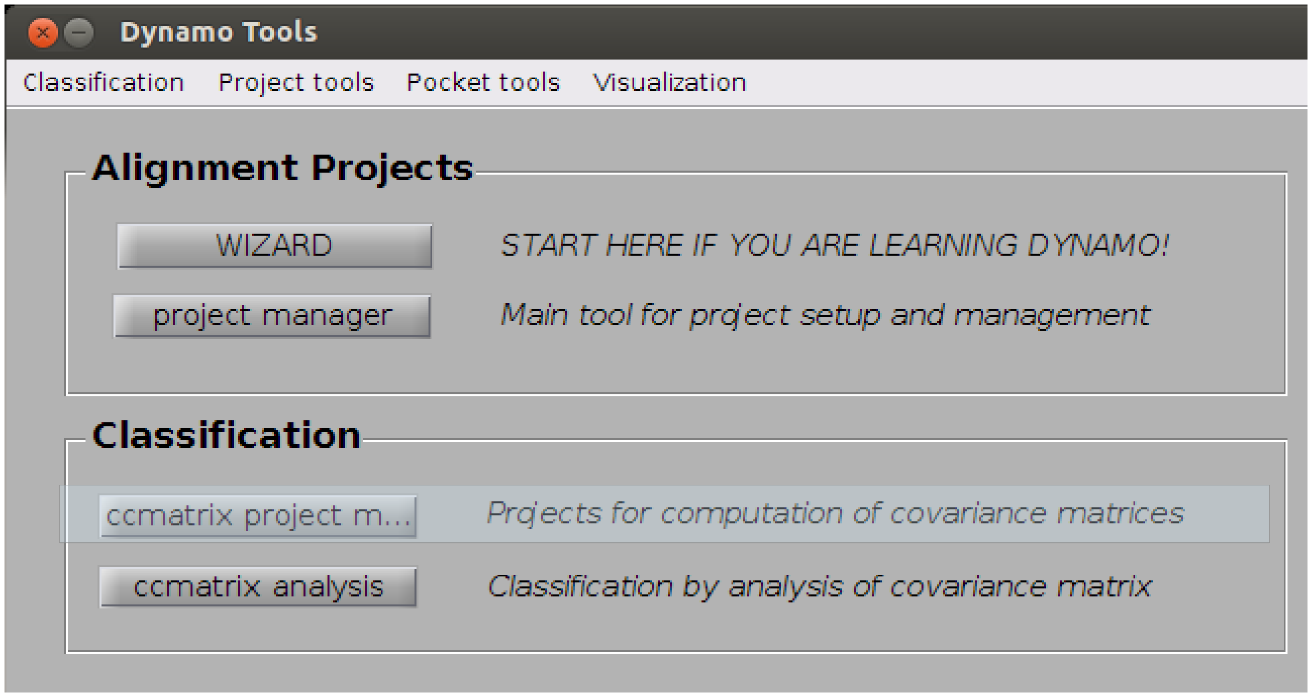
First, we illustrate how to use GUI for this end. In this case we will show the full pipeline from scratch.

First, we should invoke the GUI that controls the projects for computation of a ccmatrix:

```
>> dynamo_ccmatrix_project_manager();
```

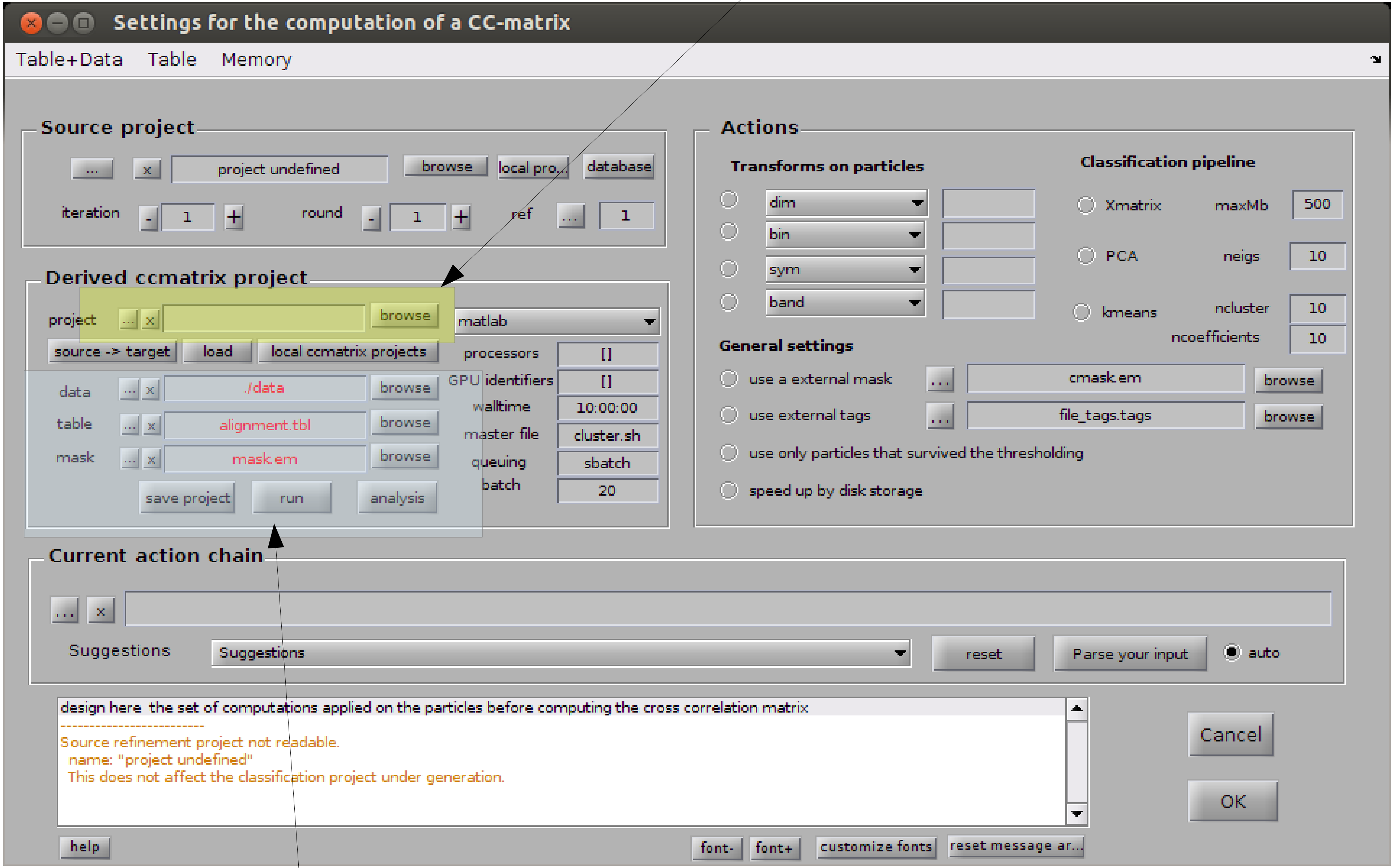
Alternatively (as not everybody remembers all commands!) you can just invoke the general dynamo menu and choose the classification option:

```
>> dynamo
```


Dynamo Tools
Classification
Project tools
Pocket tools
Visualization
Alignment Projects
WIZARD
START HERE IF YOU ARE LEARNING DYNAMO!
project manager
Main tool for project setup and management
Classification
ccmatrix project m...
Projects for computation of covariance matrices
ccmatrix analysis
Classification by analysis of covariance matrix

on the empty GUI, we start designing our project

we need to provide a name



Source project: project undefined; iteration 1; round 1; ref 1

project: matlab

data: ./data
table: alignment.tbl
mask: mask.em

processors: []
GPU identifiers: []
walltime: 10:00:00
master file: cluster.sh
queuing: sbatch
batch: 20

dim
bin
sym
band

maxMb: 500
neigs: 10
ncluster: 10
ncoefficients: 10

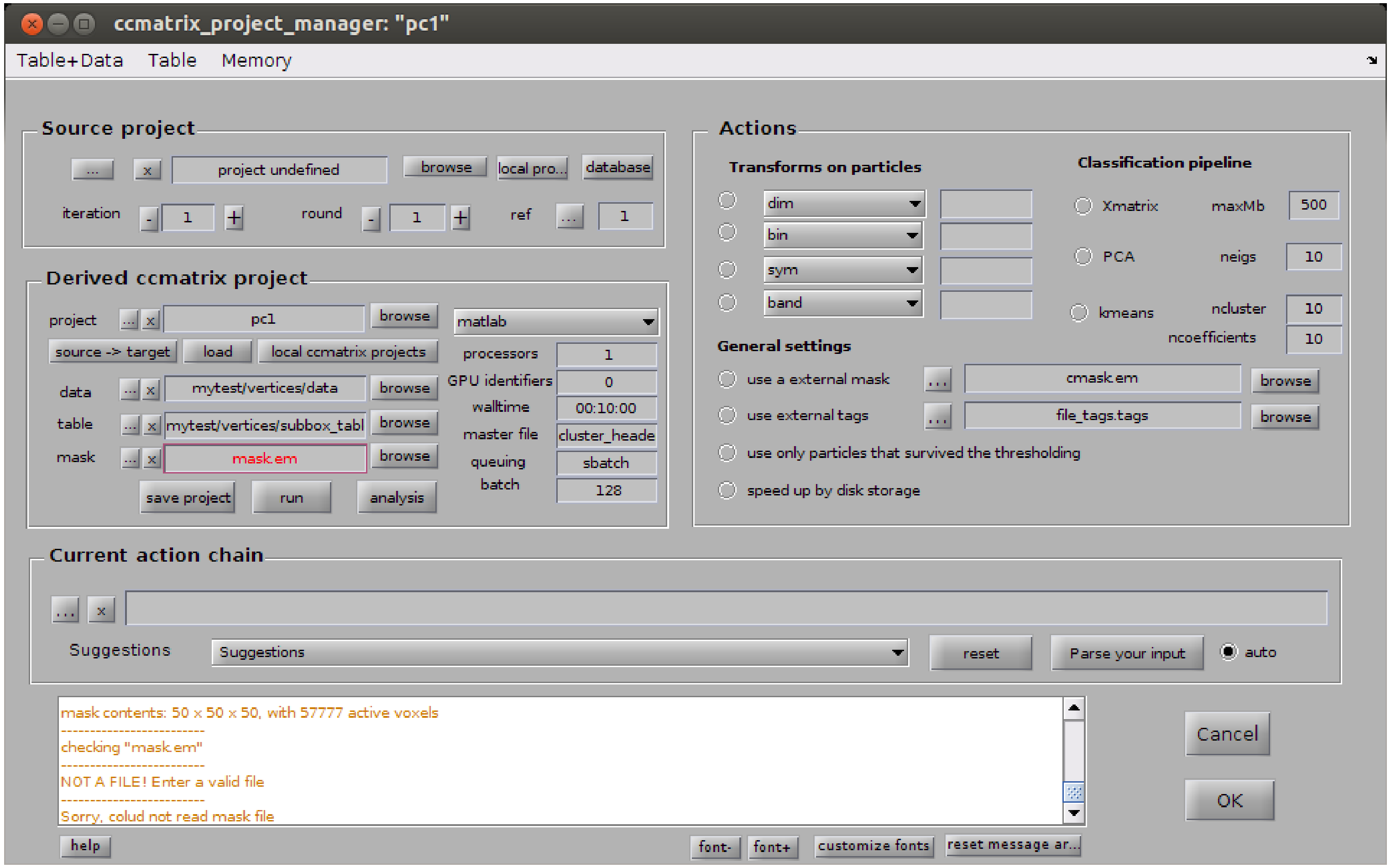
use a external mask: cmask.em
use external tags: file_tags.tags
use only particles that survived the thresholding
speed up by disk storage

Suggestions: Suggestions
auto

design here the set of computations applied on the particles before computing the cross correlation matrix
----
Source refinement project not readable.
name: "project undefined"
This does not affect the classification project under generation.

… and browse or provide the real files (table and classification mask) and data folders

data and table are the ones provided by the subboxing.
what do we do with the classification mask?



ccmatrix_project_manager: "pc1"

Source project

project	project undefined
iteration	1
round	1
ref	1

Derived ccmatrix project

project	pc1
data	mytest/vertices/data
table	mytest/vertices/subbox_tabl
mask	mask.em
	matlab
processors	1
GPU identifiers	0
walltime	00:10:00
master file	cluster_heade
queuing	sbatch
batch	128

Actions

Transforms on particles: dim, bin, sym, band

Classification pipeline:

Xmatrix	maxMb	500
PCA	neigs	10
kmeans	ncluster	10
	ncoefficients	10

General settings:

- use a external mask — cmask.em
- use external tags — file_tags.tags
- use only particles that survived the thresholding
- speed up by disk storage

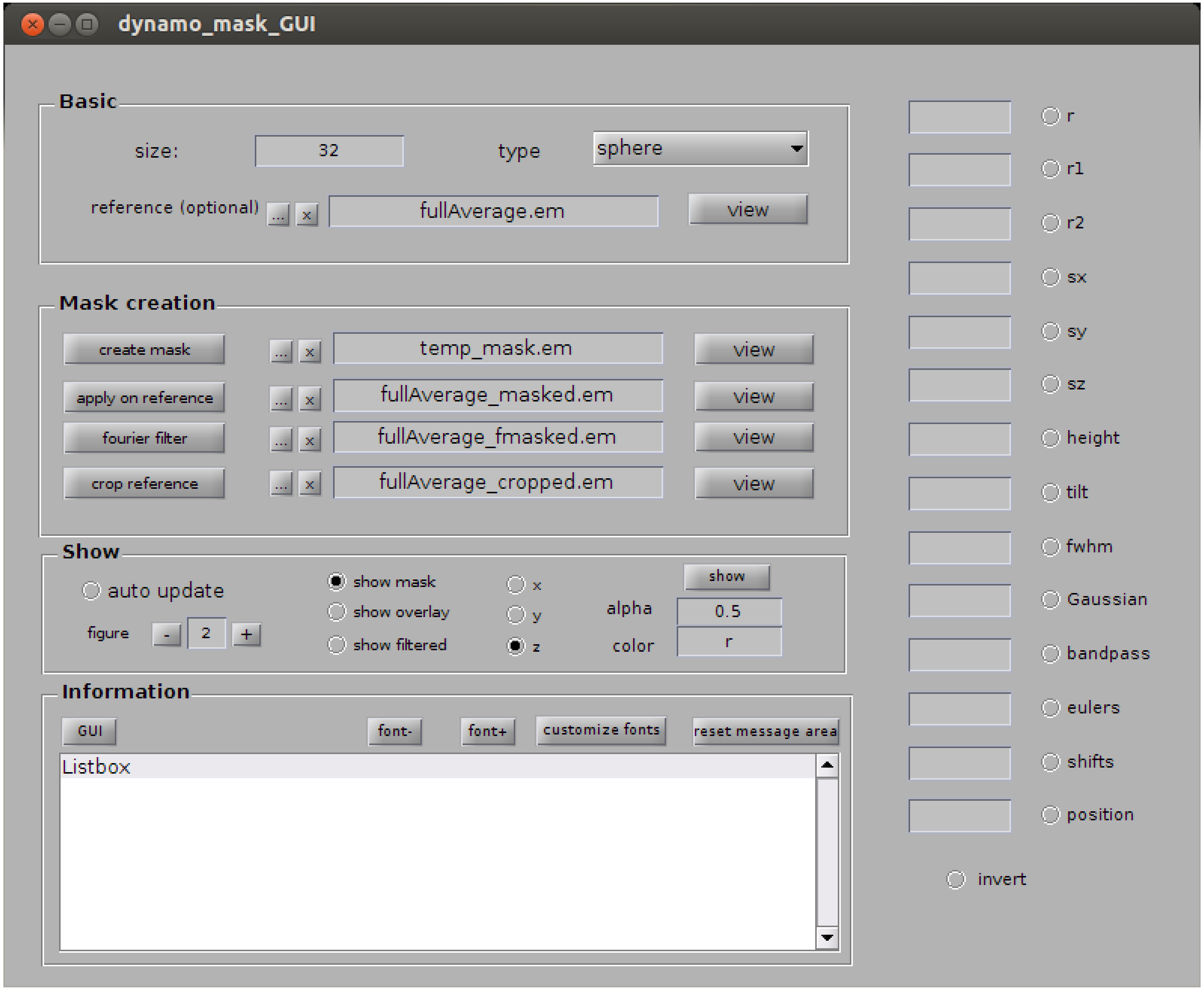
Suggestions: Suggestions — [✓ auto]

```
mask contents: 50 x 50 x 50, with 57777 active voxels
-------------------------
checking "mask.em"
-------------------------
NOT A FILE! Enter a valid file
-------------------------
Sorry, colud not read mask file
```

We will just design a mask and plug it into the GUI.

We will do it comparing it with the average of all the vertices, in order to create a mask tightly bound to the area where we expect the possible presence of inclusions, or in general the area where we expect that our sample shows structural diversity.

```
dmask -for fullAverage.em
```



dynamo_mask_GUI

Basic

size: 32 type sphere

reference (optional) fullAverage.em view

Mask creation

create mask temp_mask.em view

apply on reference fullAverage_masked.em view

fourier filter fullAverage_fmasked.em view

crop reference fullAverage_cropped.em view

Show

auto update

figure 2

[✓ show mask] show overlay show filtered

x y [✓ z]

show

alpha 0.5

color r

Information

GUI font- font+ customize fonts reset message area

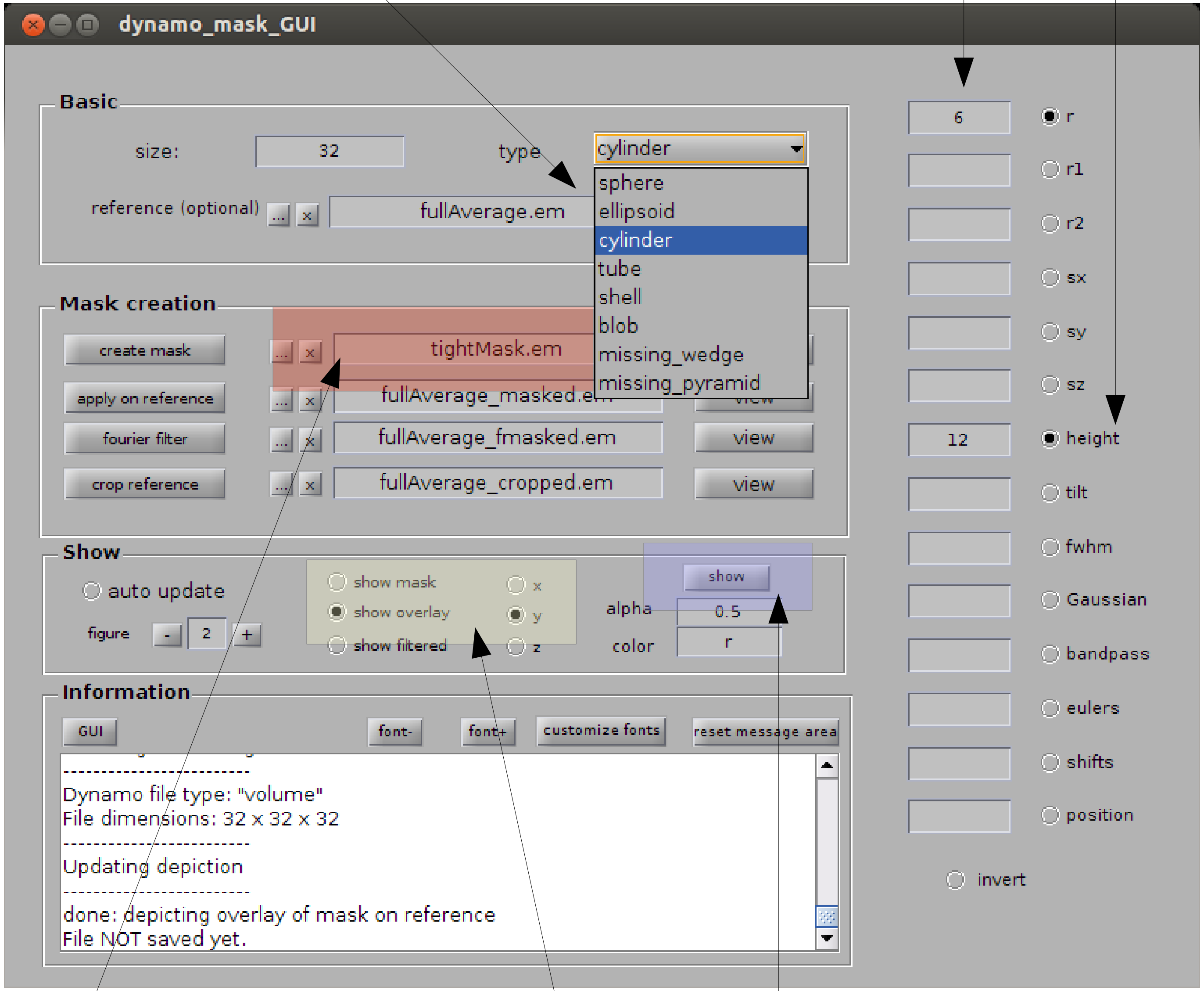
Listbox

r r1 r2 sx sy sz height tilt fwhm Gaussian bandpass eulers shifts position

invert

choose a cylinder as shape for the mask

select height and radius



size: 32

type: cylinder

reference (optional): fullAverage.em

tightMask.em

fullAverage_masked.em

fullAverage_fmasked.em

fullAverage_cropped.em

figure: 2

[✓ show overlay]

x [✓ y] z

alpha: 0.5

color: r

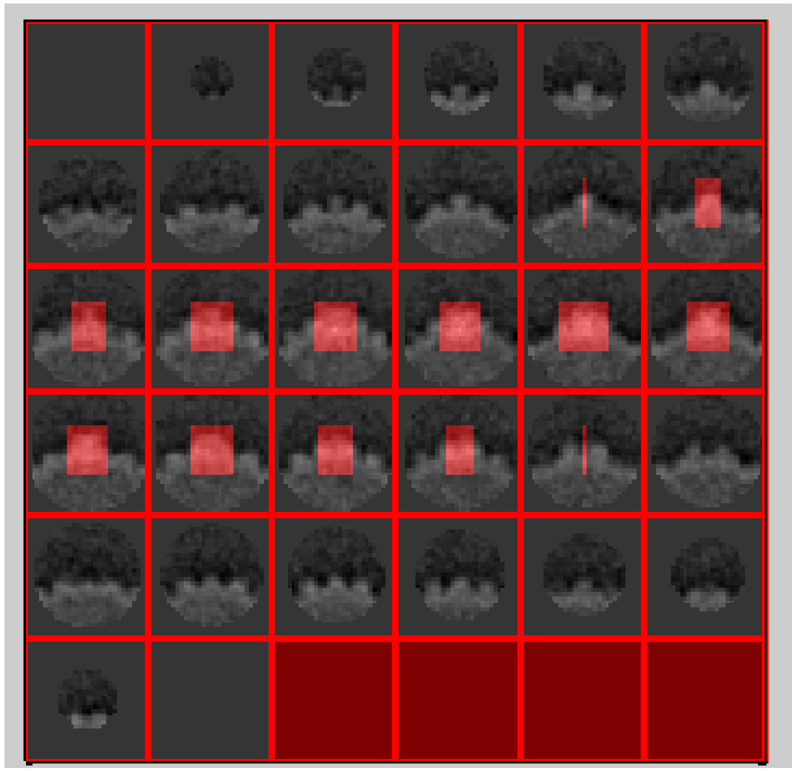
r: 6

height: 12

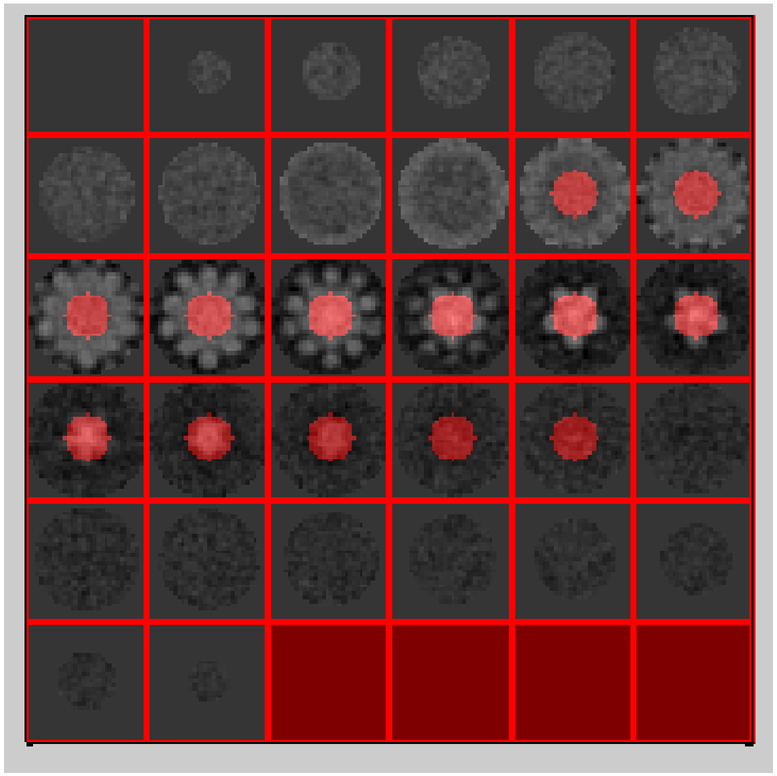
```
-------------------------
Dynamo file type: "volume"
File dimensions: 32 x 32 x 32
-------------------------
Updating depiction
-------------------------
done: depicting overlay of mask on reference
File NOT saved yet.
```

type a name for the mask file to be created

“overlay” represents the extent of the mask on the reference volume x-overlay view

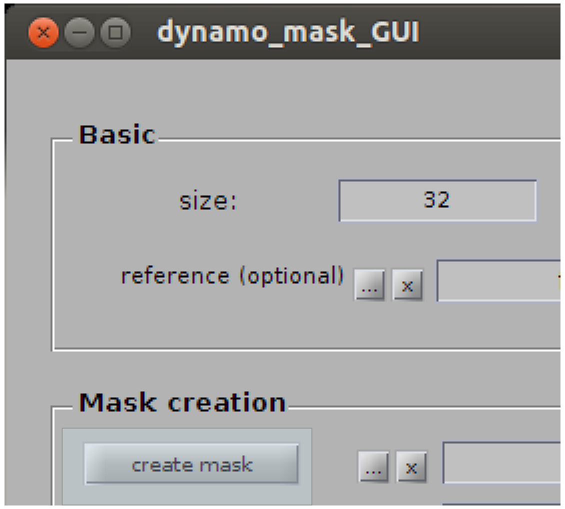


z-overlay view



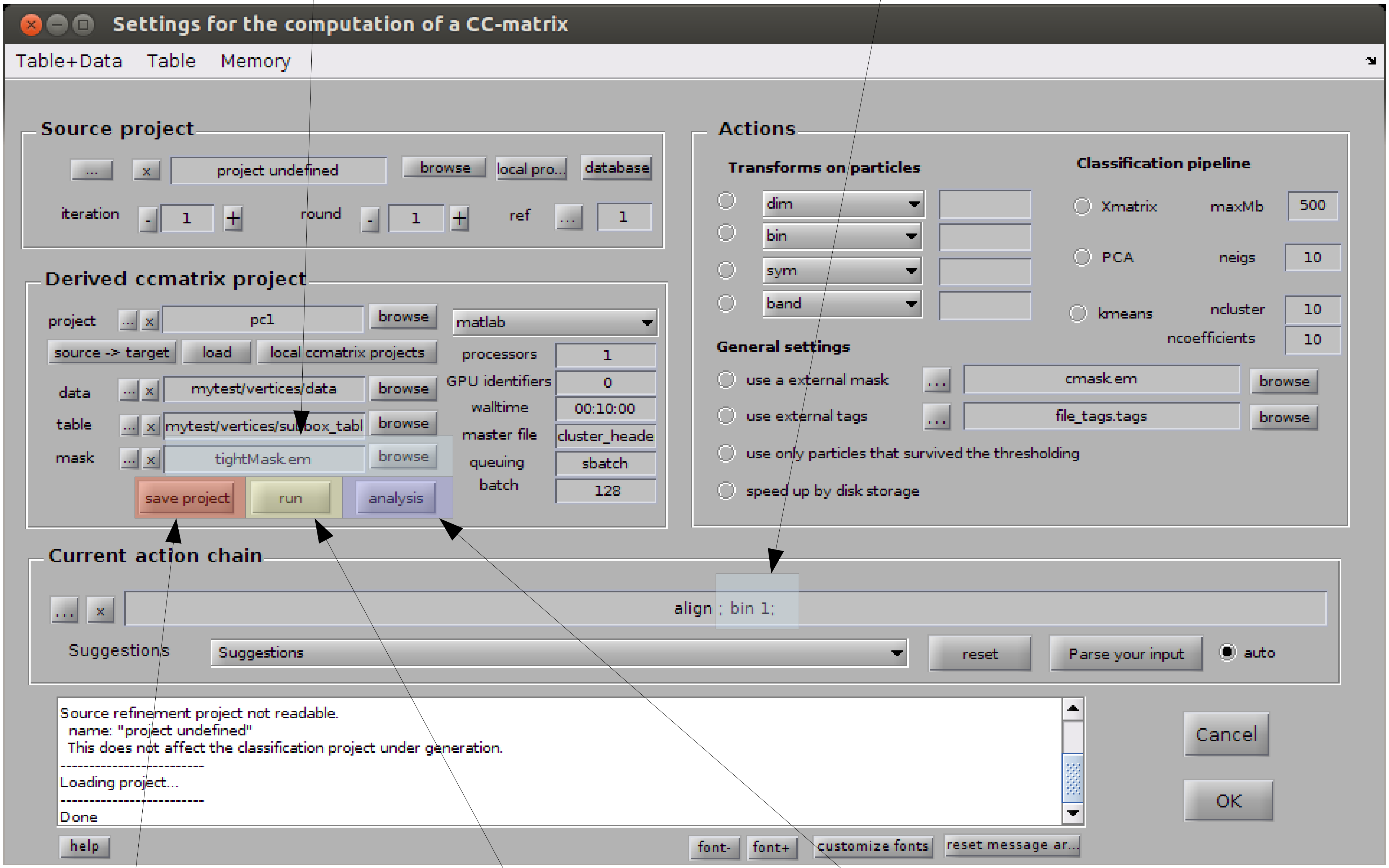
The overlay view is useful to make certain that our mask fits the area that we are interested in.

After playing with the parameters, we press on [create mask] to produce the mask file that we arew going to plug into our classification project.



We browse of type the mask file we just created.

We can set “bin 1” to spare computation time.
(The parameter “align” gets included by default, no need to add explicitly)



Settings for the computation of a CC-matrix

Table+Data Table Memory

Source project

project undefined

iteration 1 round 1 ref 1

Derived ccmatrix project

project: pc1

matlab

processors: 1

GPU identifiers: 0

walltime: 00:10:00

master file: cluster_heade

queuing: sbatch

batch: 128

data: mytest/vertices/data

table: mytest/vertices/subbox_tabl

mask: tightMask.em

Actions

Transforms on particles

dim, bin, sym, band

Classification pipeline

Xmatrix maxMb 500

PCA neigs 10

kmeans ncluster 10

ncoefficients 10

General settings

use a external mask: cmask.em

use external tags: file_tags.tags

use only particles that survived the thresholding

speed up by disk storage

Current action chain

align ; bin 1;

Suggestions

(●) auto

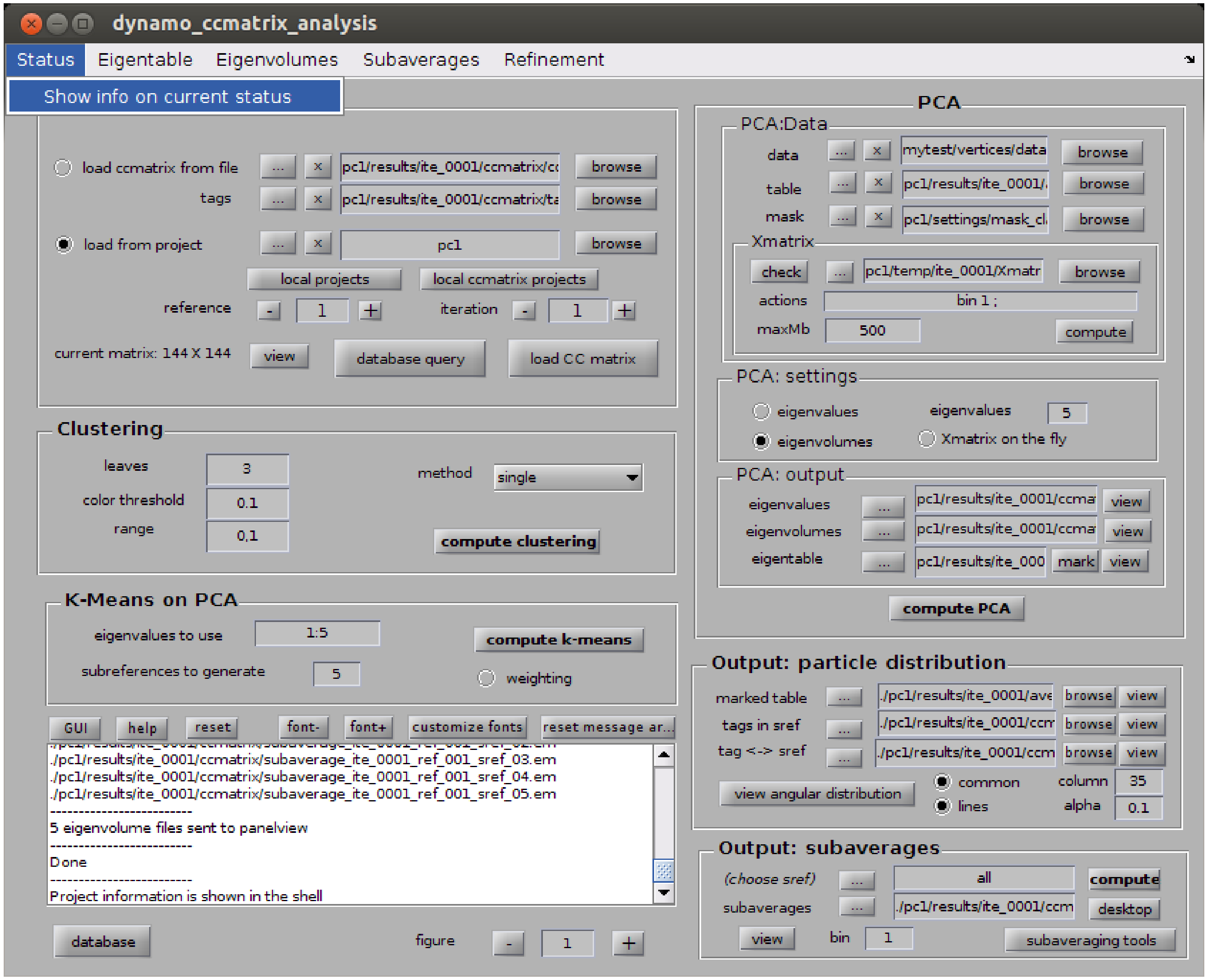
```
Source refinement project not readable.
  name: "project undefined"
  This does not affect the classification project under generation.
-------------------------
Loading project...
-------------------------
Done
```

we save the project...

...run it (can take some time)...

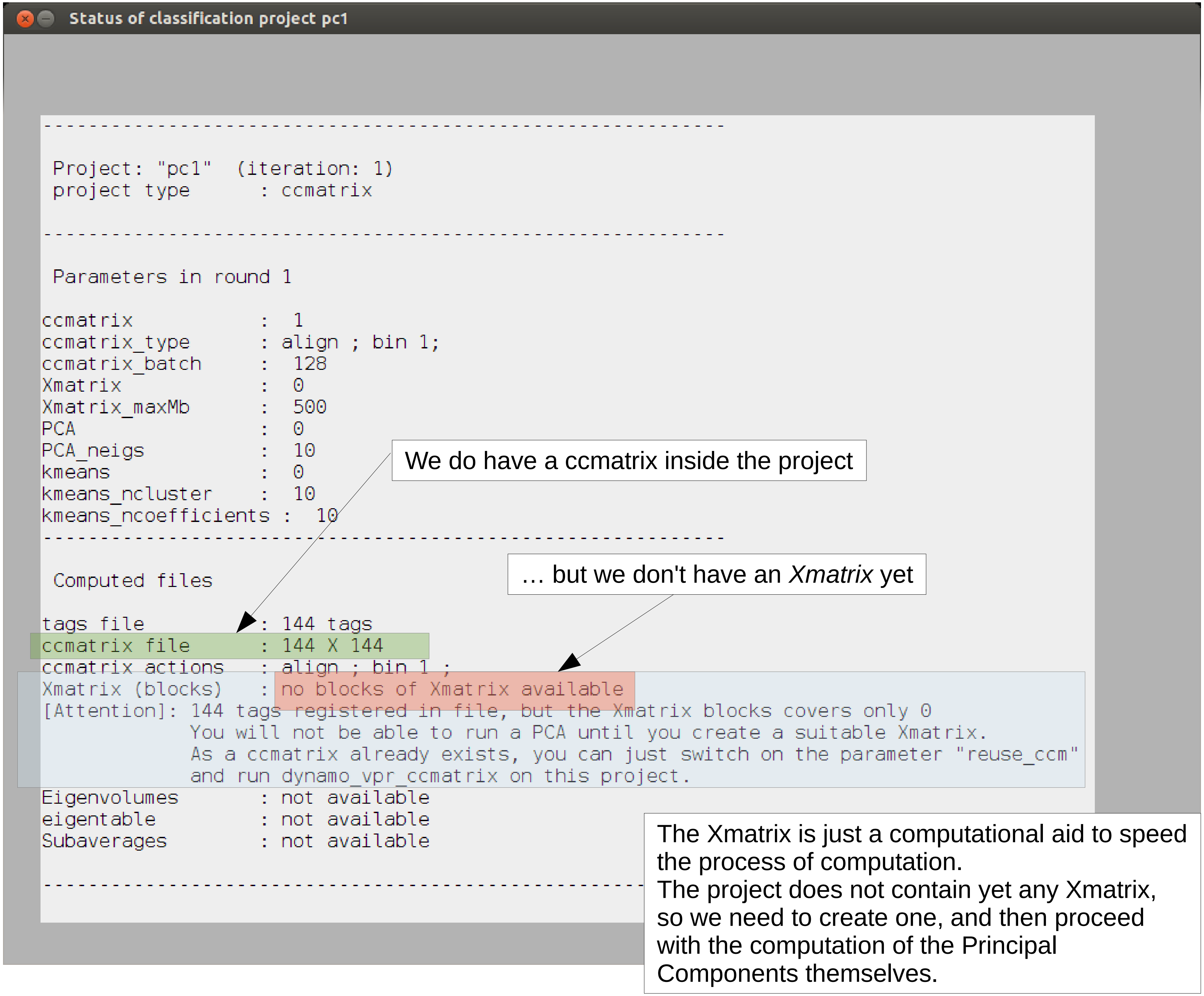
… and then open the analysis GUI

We can check in [Status] in which part of the classification pipeline we are right now.



load ccmatrix from file	pc1/results/ite_0001/ccmatrix/cc
tags	pc1/results/ite_0001/ccmatrix/ta
[✓ load from project]	pc1
reference	1
iteration	1
current matrix:	144 X 144
leaves	3
color threshold	0.1
range	0.1
method	single
eigenvalues to use	1:5
subreferences to generate	5
data	mytest/vertices/data
table	pc1/results/ite_0001/
mask	pc1/settings/mask_cl
Xmatrix	pc1/temp/ite_0001/Xmatr
actions	bin 1 ;
maxMb	500
eigenvalues	5
[✓ eigenvolumes]	
Xmatrix on the fly	
eigenvalues	pc1/results/ite_0001/ccma
eigenvolumes	pc1/results/ite_0001/ccma
eigentable	pc1/results/ite_000
marked table	./pc1/results/ite_0001/ave
tags in sref	./pc1/results/ite_0001/ccm
tag <-> sref	./pc1/results/ite_0001/ccm
[✓ common]	column 35
[✓ lines]	alpha 0.1
(choose sref)	all
subaverages	./pc1/results/ite_0001/ccm
bin	1

```
./pc1/results/ite_0001/ccmatrix/subaverage_ite_0001_ref_001_sref_03.em
./pc1/results/ite_0001/ccmatrix/subaverage_ite_0001_ref_001_sref_04.em
./pc1/results/ite_0001/ccmatrix/subaverage_ite_0001_ref_001_sref_05.em
------------------------
5 eigenvolume files sent to panelview
------------------------
Done
------------------------
Project information is shown in the shell
```



Status of classification project pc1

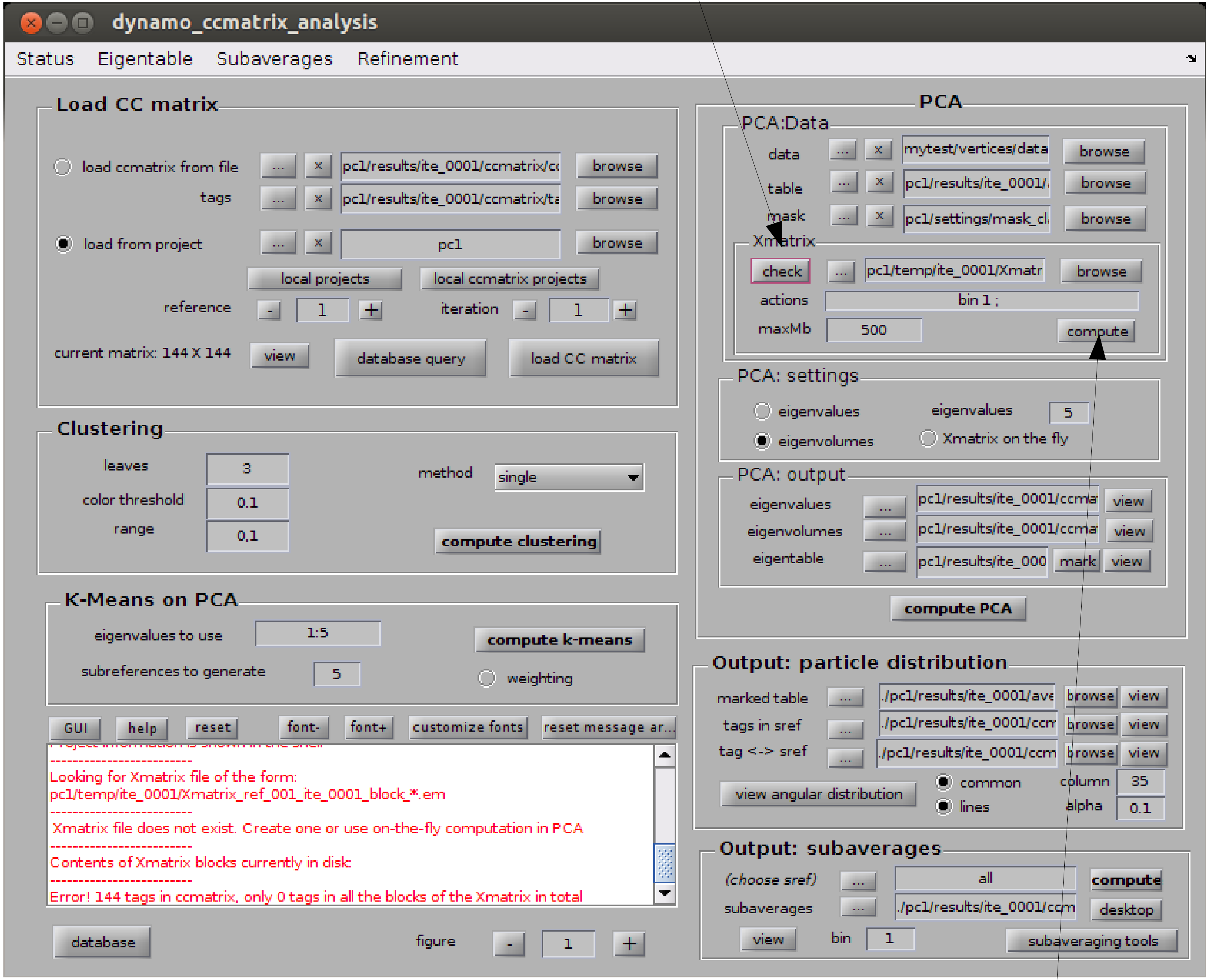
```
------------------------------------------------------------

 Project: "pc1"  (iteration: 1)
 project type      : ccmatrix

------------------------------------------------------------

 Parameters in round 1

ccmatrix            :  1
ccmatrix_type       : align ; bin 1;
ccmatrix_batch      :  128
Xmatrix             :  0
Xmatrix_maxMb       :  500
PCA                 :  0
PCA_neigs           :  10
kmeans              :  0
kmeans_ncluster     :  10
kmeans_ncoefficients :  10
------------------------------------------------------------

 Computed files

tags file           : 144 tags
ccmatrix file       : 144 X 144
ccmatrix actions    : align ; bin 1 ;
Xmatrix (blocks)    : no blocks of Xmatrix available
[Attention]: 144 tags registered in file, but the Xmatrix blocks covers only 0
             You will not be able to run a PCA until you create a suitable Xmatrix.
             As a ccmatrix already exists, you can just switch on the parameter "reuse_ccm"
             and run dynamo_vpr_ccmatrix on this project.
Eigenvolumes        : not available
eigentable          : not available
Subaverages         : not available

------------------------------------------------------------
```

We do have a ccmatrix inside the project

… but we don't have an *Xmatrix* yet

The Xmatrix is just a computational aid to speed the process of computation.
The project does not contain yet any Xmatrix, so we need to create one, and then proceed with the computation of the Principal Components themselves.

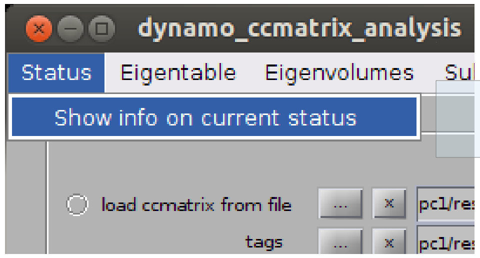
checking directly the status of the xmatrix confirms it's not there....

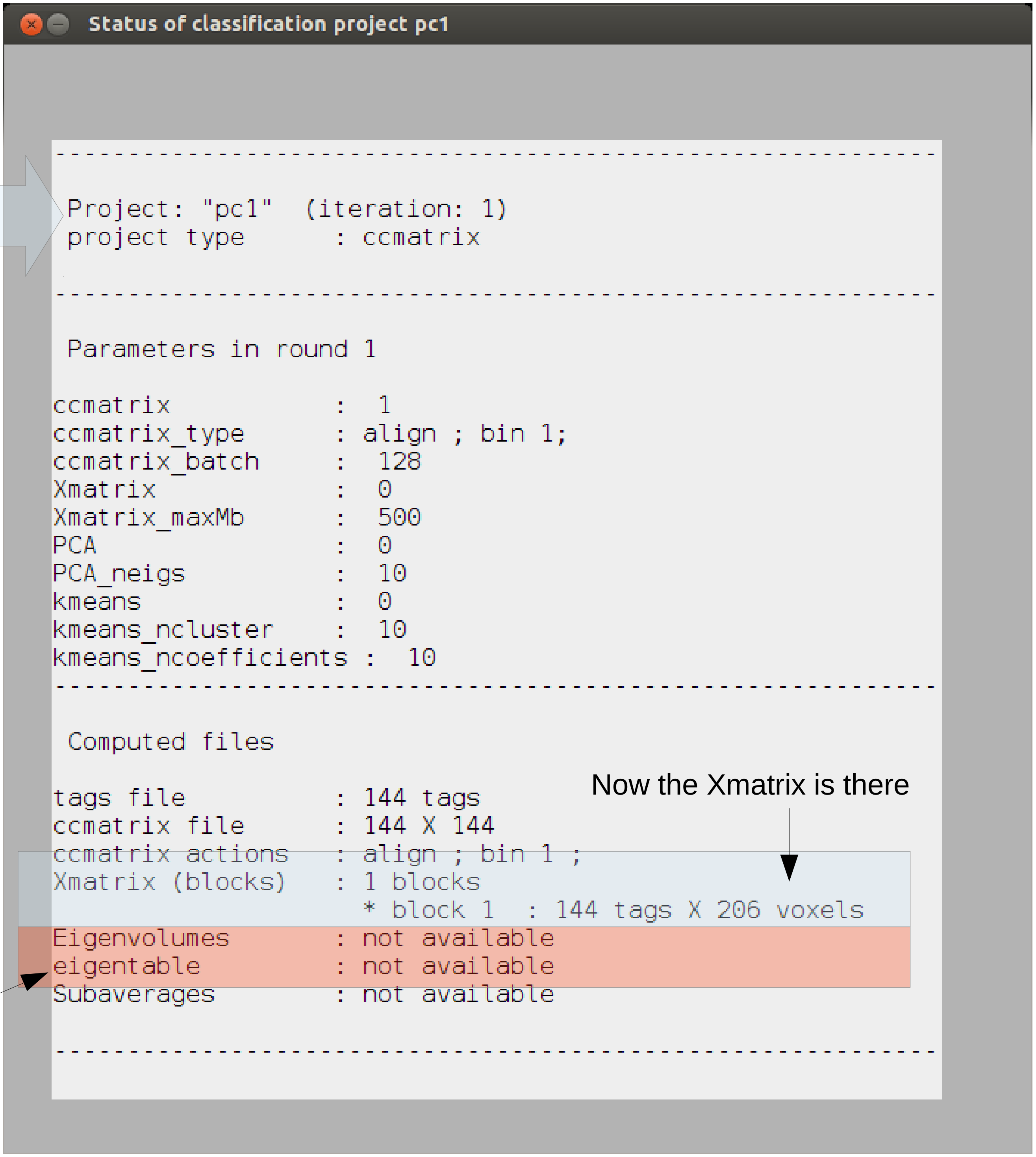


load ccmatrix from file	pc1/results/ite_0001/ccmatrix/cc
tags	pc1/results/ite_0001/ccmatrix/ta
[✓ load from project]	pc1
reference	1
iteration	1
leaves	3
color threshold	0.1
range	0.1
method	single
eigenvalues to use	1:5
subreferences to generate	5
data	mytest/vertices/data
table	pc1/results/ite_0001/
mask	pc1/settings/mask_cl
Xmatrix	pc1/temp/ite_0001/Xmatr
actions	bin 1 ;
maxMb	500
[✓ eigenvolumes] eigenvalues	5
eigenvalues	pc1/results/ite_0001/ccma
eigenvolumes	pc1/results/ite_0001/ccma
eigentable	pc1/results/ite_000
marked table	./pc1/results/ite_0001/ave
tags in sref	./pc1/results/ite_0001/ccm
tag <-> sref	./pc1/results/ite_0001/ccm
column	35
alpha	0.1
(choose sref)	all
subaverages	./pc1/results/ite_0001/ccm
bin	1
figure	1

```
-------------------------
Looking for Xmatrix file of the form:
pc1/temp/ite_0001/Xmatrix_ref_001_ite_0001_block_*.em
-------------------------
 Xmatrix file does not exist. Create one or use on-the-fly computation in PCA
-------------------------
Contents of Xmatrix blocks currently in disk
-------------------------
Error! 144 tags in ccmatrix, only 0 tags in all the blocks of the Xmatrix in total
```

sow we just compute it and carry on



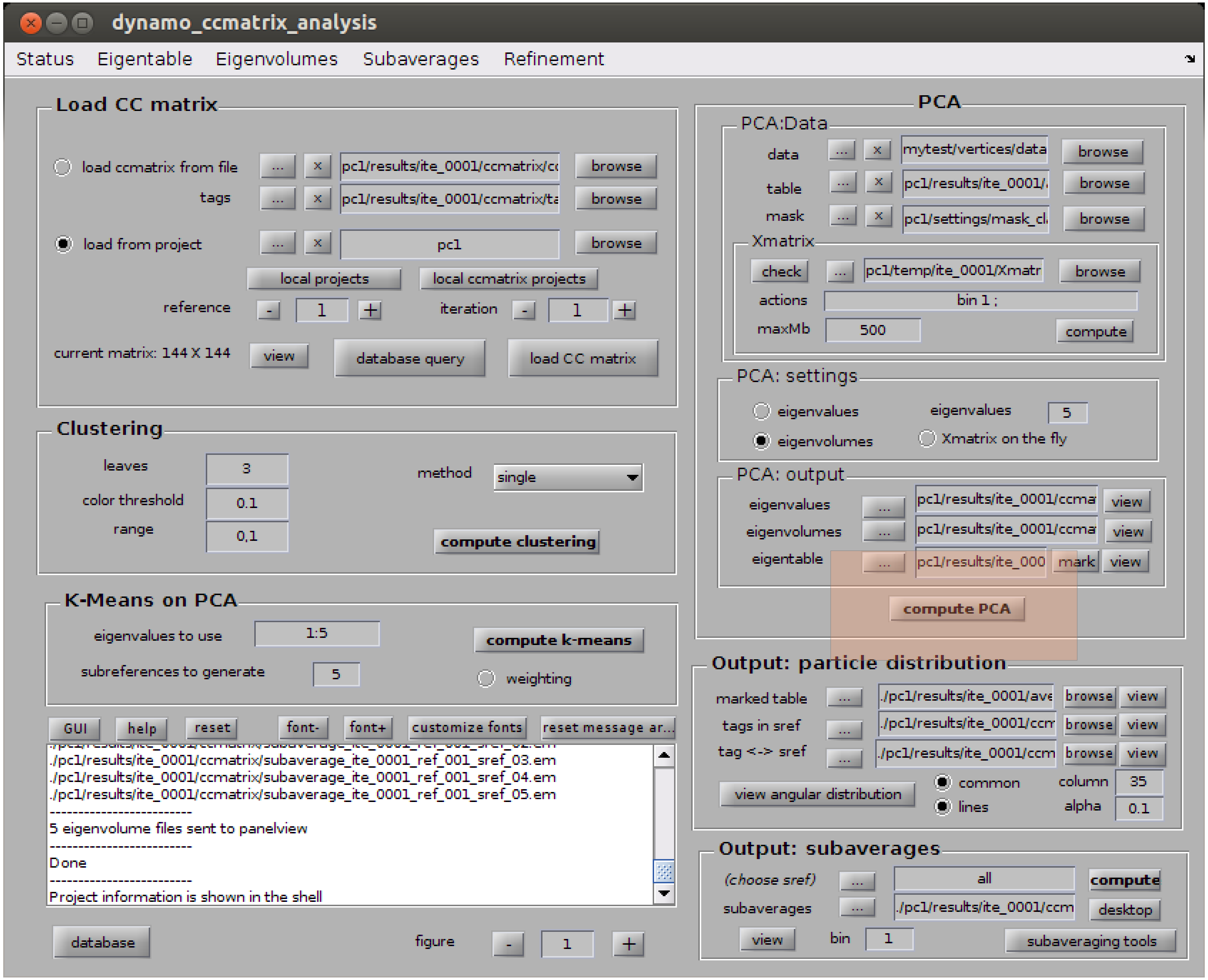


```
------------------------------------------------------------

 Project: "pc1"  (iteration: 1)
 project type       : ccmatrix

------------------------------------------------------------

 Parameters in round 1

ccmatrix              :  1
ccmatrix_type         : align ; bin 1;
ccmatrix_batch        :  128
Xmatrix               :  0
Xmatrix_maxMb         :  500
PCA                   :  0
PCA_neigs             :  10
kmeans                :  0
kmeans_ncluster       :  10
kmeans_ncoefficients :  10
------------------------------------------------------------

 Computed files

tags file             : 144 tags
ccmatrix file         : 144 X 144
ccmatrix actions      : align ; bin 1 ;
Xmatrix (blocks)      : 1 blocks
                        * block 1  : 144 tags X 206 voxels
Eigenvolumes          : not available
eigentable            : not available
Subaverages           : not available

------------------------------------------------------------
```

Now the Xmatrix is there

so we can proced with
the still missing PCA analysis

Press to compute eigenvolumes (principal components) and eigentable (particle coordinates on this basis)



Load CC matrix:
- load ccmatrix from file: pc1/results/ite_0001/ccmatrix/cc
- tags: pc1/results/ite_0001/ccmatrix/ta
- [● load from project]: pc1
- reference: 1
- iteration: 1
- current matrix: 144 X 144

Clustering:
- leaves: 3
- color threshold: 0.1
- range: 0.1
- method: single

K-Means on PCA:
- eigenvalues to use: 1:5
- subreferences to generate: 5
- weighting

PCA:Data:
- data: mytest/vertices/data
- table: pc1/results/ite_0001/.
- mask: pc1/settings/mask_cl.
- Xmatrix: pc1/temp/ite_0001/Xmatr
- actions: bin 1 ;
- maxMb: 500

PCA: settings:
- eigenvalues
- [● eigenvolumes]
- eigenvalues: 5
- Xmatrix on the fly

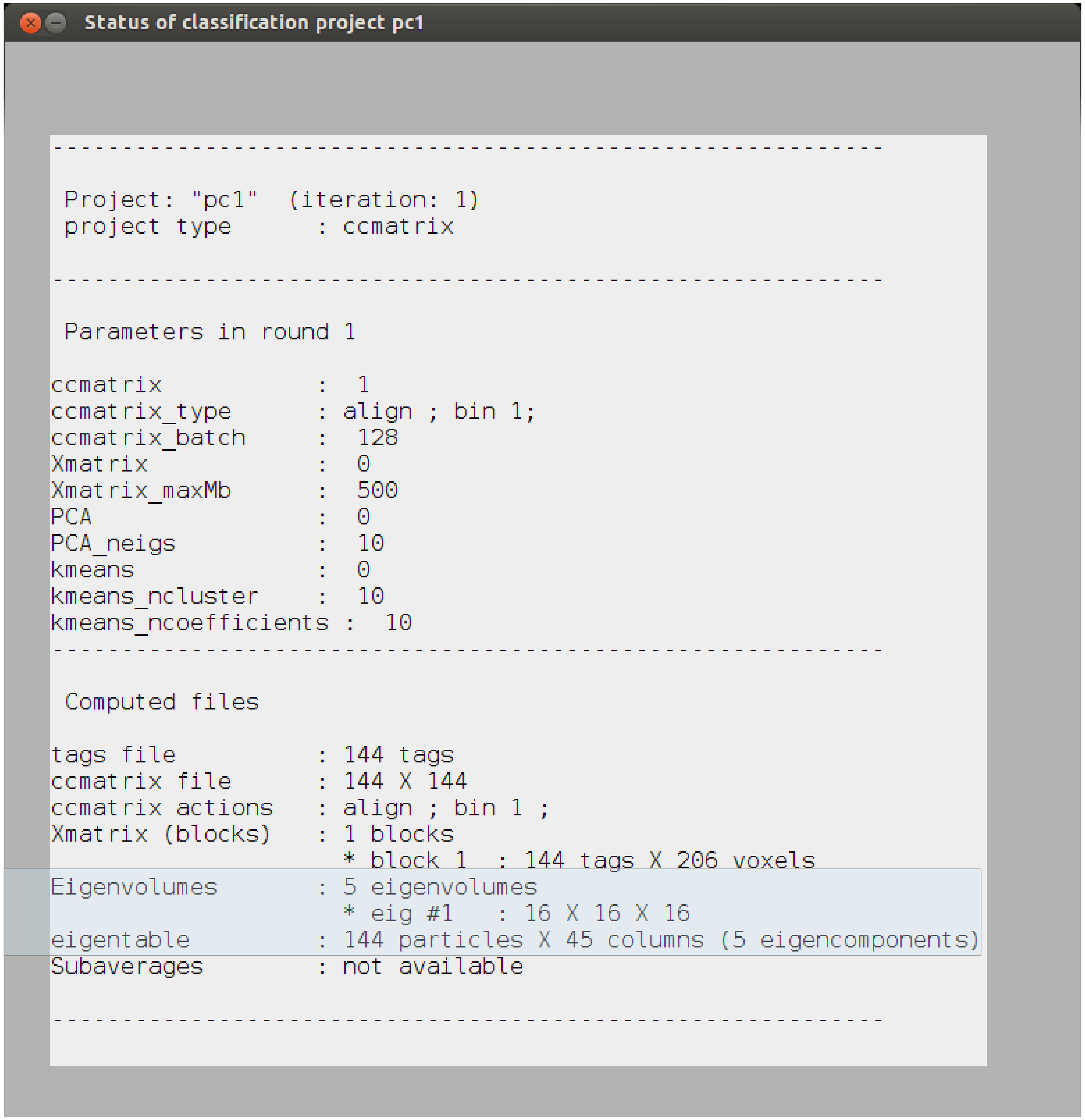
PCA: output:
- eigenvalues: pc1/results/ite_0001/ccma
- eigenvolumes: pc1/results/ite_0001/ccma
- eigentable: pc1/results/ite_000

Output: particle distribution:
- marked table: ./pc1/results/ite_0001/ave
- tags in sref: ./pc1/results/ite_0001/ccm
- tag <-> sref: ./pc1/results/ite_0001/ccm
- [● common]
- [● lines]
- column: 35
- alpha: 0.1

Output: subaverages:
- *(choose sref)*: all
- subaverages: ./pc1/results/ite_0001/ccm
- bin: 1

figure: 1

```
./pc1/results/ite_0001/ccmatrix/subaverage_ite_0001_ref_001_sref_03.em
./pc1/results/ite_0001/ccmatrix/subaverage_ite_0001_ref_001_sref_04.em
./pc1/results/ite_0001/ccmatrix/subaverage_ite_0001_ref_001_sref_05.em
------------------------
5 eigenvolume files sent to panelview
------------------------
Done
------------------------
Project information is shown in the shell
```

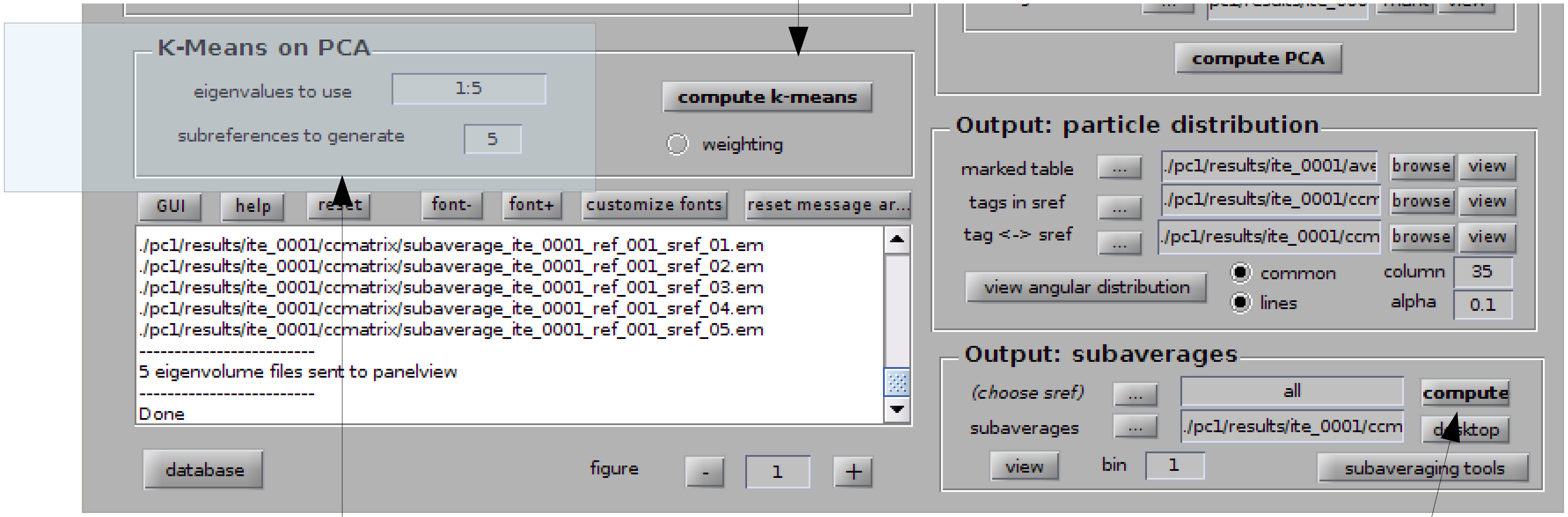

Status of classification project pc1

```
-----------------------------------------------------------


 Project: "pc1"  (iteration: 1)
 project type       : ccmatrix

-----------------------------------------------------------


 Parameters in round 1

ccmatrix             :  1
ccmatrix_type        : align ; bin 1;
ccmatrix_batch       :  128
Xmatrix              :  0
Xmatrix_maxMb        :  500
PCA                  :  0
PCA_neigs            :  10
kmeans               :  0
kmeans_ncluster      :  10
kmeans_ncoefficients :  10
-----------------------------------------------------------


 Computed files

tags file            : 144 tags
ccmatrix file        : 144 X 144
ccmatrix actions     : align ; bin 1 ;
Xmatrix (blocks)     : 1 blocks
                       * block 1  : 144 tags X 206 voxels
Eigenvolumes         : 5 eigenvolumes
                       * eig #1   : 16 X 16 X 16
eigentable           : 144 particles X 45 columns (5 eigencomponents)
Subaverages          : not available

-----------------------------------------------------------
```

Now we have the eigenvolumes and the eigentable that extends the initial table with new columns.

Columns 41 onwards in this table are the components of the particle in the basis of eigenvolumes.

we can compute our classification on the system of coordinates induced by the PCA

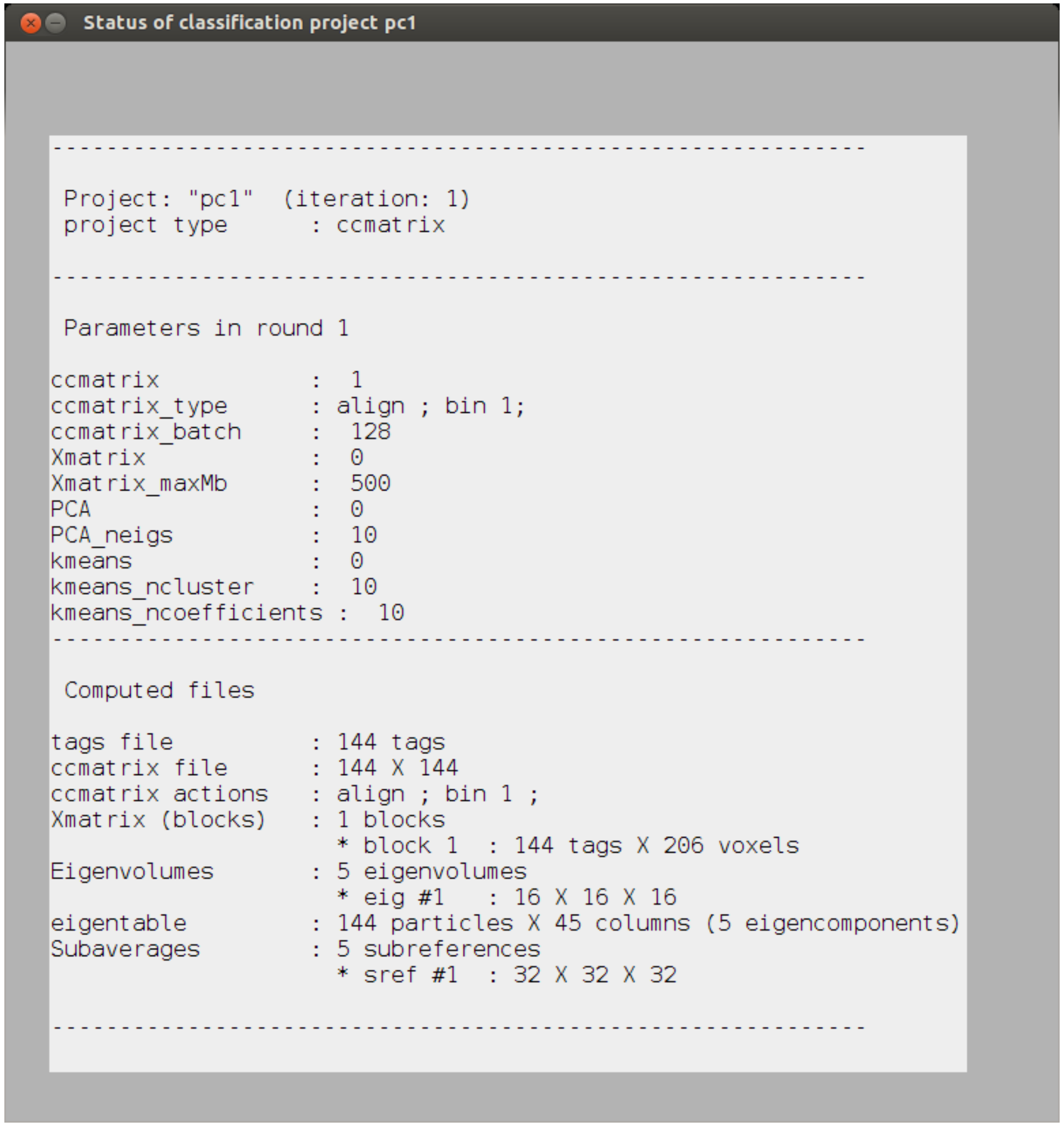

K-Means on PCA
eigenvalues to use 1:5
subreferences to generate 5
compute k-means
weighting
GUI
help
reset
font-
font+
customize fonts
reset message ar...
database
figure - 1 +
compute PCA
Output: particle distribution
marked table ./pcl/results/ite_0001/ave browse view
tags in sref ./pcl/results/ite_0001/ccm browse view
tag <-> sref ./pcl/results/ite_0001/ccm browse view
view angular distribution
common
lines
column 35
alpha 0.1
Output: subaverages
(choose sref) all compute
subaverages ./pcl/results/ite_0001/ccm desktop
view
bin 1
subaveraging tools

```
./pcl/results/ite_0001/ccmatrix/subaverage_ite_0001_ref_001_sref_01.em
./pcl/results/ite_0001/ccmatrix/subaverage_ite_0001_ref_001_sref_02.em
./pcl/results/ite_0001/ccmatrix/subaverage_ite_0001_ref_001_sref_03.em
./pcl/results/ite_0001/ccmatrix/subaverage_ite_0001_ref_001_sref_04.em
./pcl/results/ite_0001/ccmatrix/subaverage_ite_0001_ref_001_sref_05.em
------------------------
5 eigenvolume files sent to panelview
------------------------
Done
```

real life often requires a careful reflexion on these parameters!
inspection of the eigenvolumes is useful to decide which ones to use for classification.

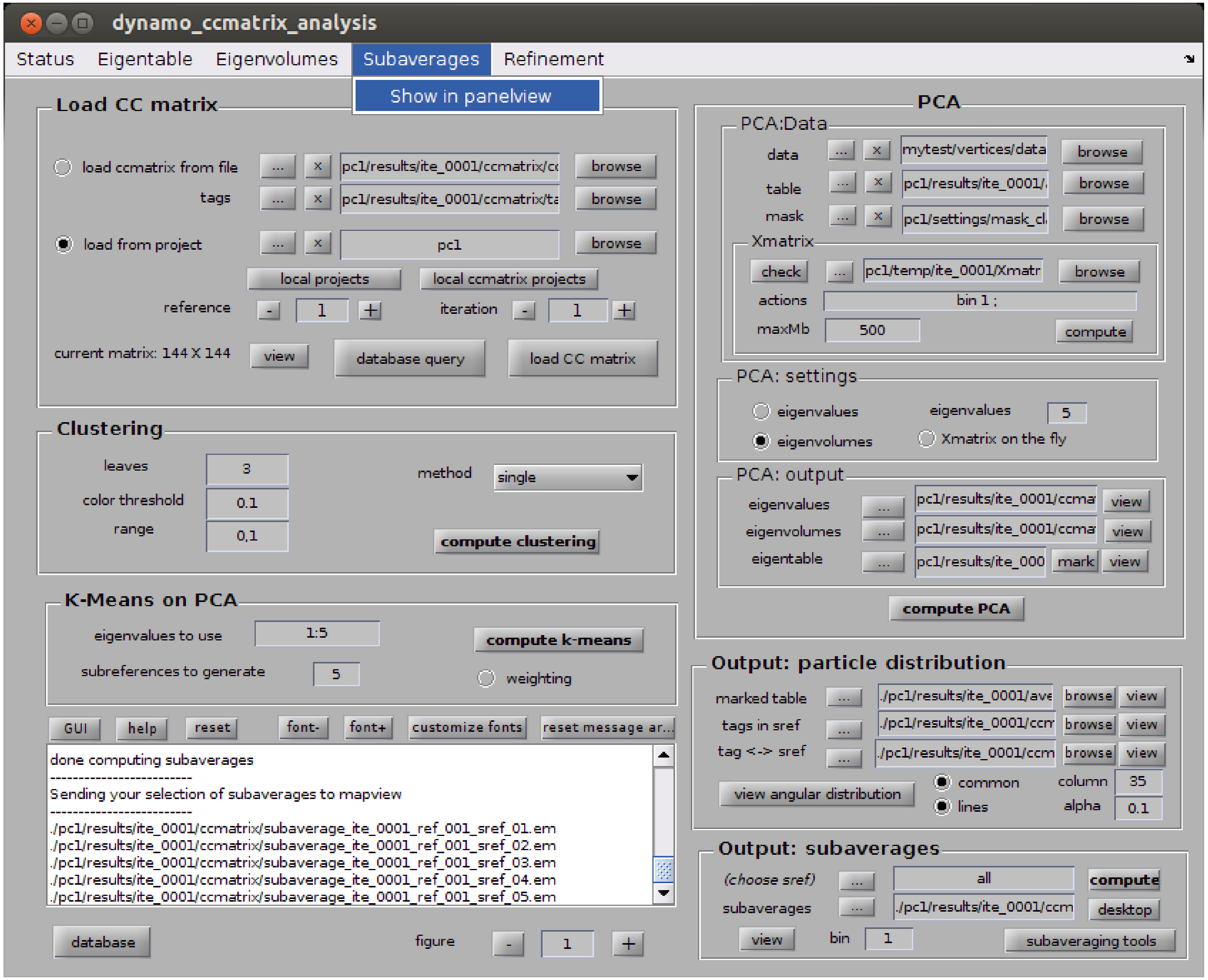
and when it's done we can compute the subaverages (class averages) induced by the classification that we just computed.

Now, the [Status] button shows that the classification is complete


Status of classification project pc1

```
-----------------------------------------------------------


 Project: "pc1"  (iteration: 1)
 project type      : ccmatrix

-----------------------------------------------------------


 Parameters in round 1

ccmatrix            :  1
ccmatrix_type       : align ; bin 1;
ccmatrix_batch      :  128
Xmatrix             :  0
Xmatrix_maxMb       :  500
PCA                 :  0
PCA_neigs           :  10
kmeans              :  0
kmeans_ncluster     :  10
kmeans_ncoefficients :  10
-----------------------------------------------------------


 Computed files

tags file           : 144 tags
ccmatrix file       : 144 X 144
ccmatrix actions    : align ; bin 1 ;
Xmatrix (blocks)    : 1 blocks
                      * block 1  : 144 tags X 206 voxels
Eigenvolumes        : 5 eigenvolumes
                      * eig #1   : 16 X 16 X 16
eigentable          : 144 particles X 45 columns (5 eigencomponents)
Subaverages         : 5 subreferences
                      * sref #1  : 32 X 32 X 32


-----------------------------------------------------------
```

panelview is the most comfortable way to show a family photo of the computed subaverages



Load CC matrix
- load ccmatrix from file: pcl/results/ite_0001/ccmatrix/cc
- tags: pcl/results/ite_0001/ccmatrix/ta
- [● load from project]: pcl
- reference: 1
- iteration: 1
- current matrix: 144 X 144

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

K-Means on PCA
- eigenvalues to use: 1:5
- subreferences to generate: 5
- weighting: ○

```
done computing subaverages
------------------------
Sending your selection of subaverages to mapview
------------------------
./pcl/results/ite_0001/ccmatrix/subaverage_ite_0001_ref_001_sref_01.em
./pcl/results/ite_0001/ccmatrix/subaverage_ite_0001_ref_001_sref_02.em
./pcl/results/ite_0001/ccmatrix/subaverage_ite_0001_ref_001_sref_03.em
./pcl/results/ite_0001/ccmatrix/subaverage_ite_0001_ref_001_sref_04.em
./pcl/results/ite_0001/ccmatrix/subaverage_ite_0001_ref_001_sref_05.em
```

figure: 1

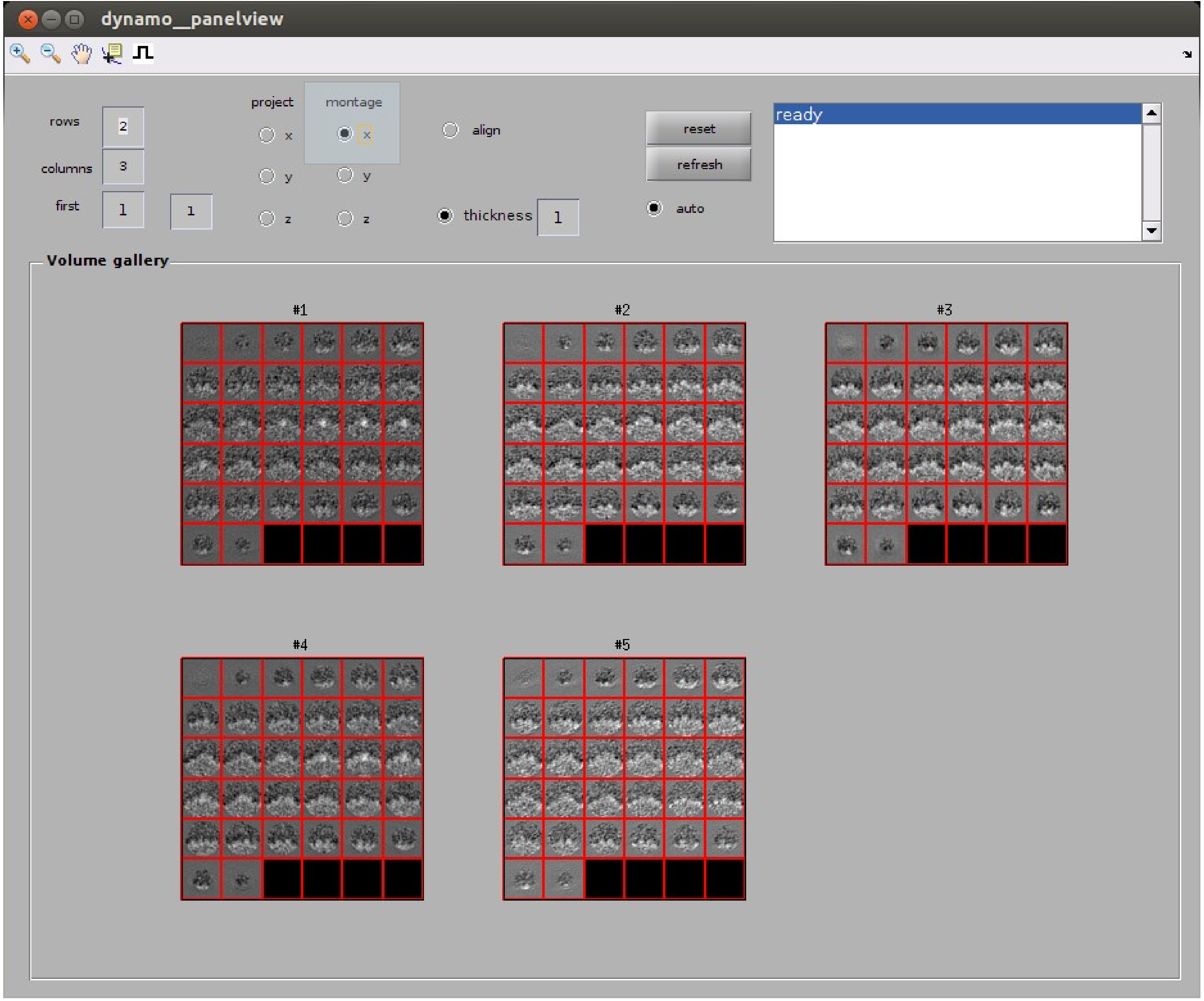
PCA:Data
- data: mytest/vertices/data
- table: pcl/results/ite_0001/
- mask: pcl/settings/mask_cl
- Xmatrix: pcl/temp/ite_0001/Xmatr
- actions: bin 1 ;
- maxMb: 500

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

PCA: output
- eigenvalues: pcl/results/ite_0001/ccma
- eigenvolumes: pcl/results/ite_0001/ccma
- eigentable: pcl/results/ite_000

Output: particle distribution
- marked table: ./pcl/results/ite_0001/ave
- tags in sref: ./pcl/results/ite_0001/ccm
- tag <-> sref: ./pcl/results/ite_0001/ccm
- [● common]
- [● lines]
- column: 35
- alpha: 0.1

Output: subaverages
- *(choose sref)*: all
- subaverages: ./pcl/results/ite_0001/ccm
- bin: 1



There are two classes showing the moiety, and three that don't

PART II(b)

- *PCA from command line*
- *Impact of the classification mask*

Once you feel confident with the classification pipeline, things can be done more efficiently through the command line.

For instance, let's check what happens if we choose to use a bigger mask for classification.

This time we will operate the full process (setting the project, running it and accessing the results) from the prompt.

creates a bigger, spherical mask..

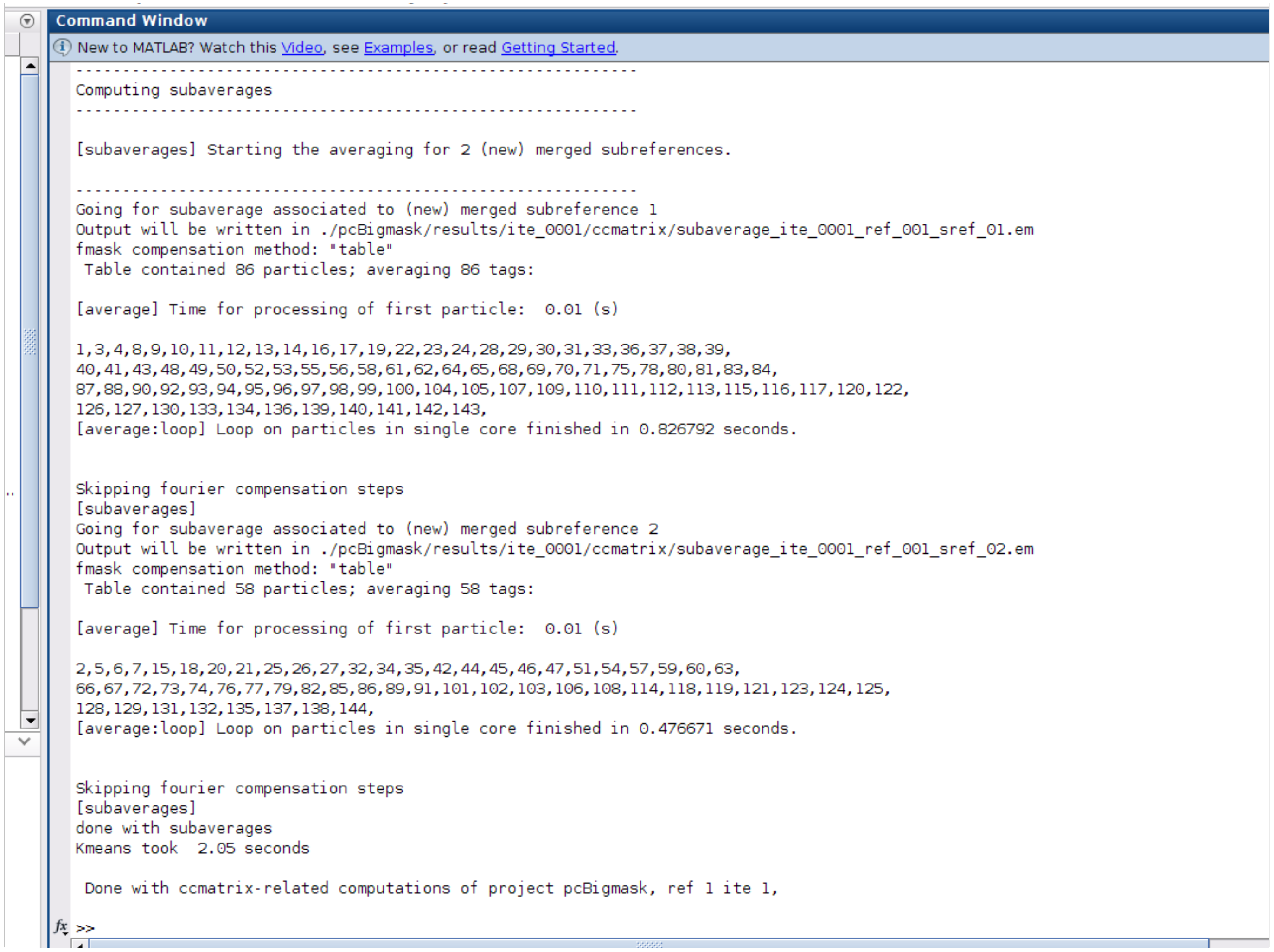
```
>> dmask -size 32 -r 16 -o maskBig.em

>> dvccmatrix pcBigmask -d mytest/vertices/data -t mytest/vertices/subbox_table.tbl
-m maskBig.em -steps all -bin 1;

>> dvunfold pcBigmask

>> pcBigmask
```

States that new classification project will include all steps of the classificacion pipeline:

- ccmatrix
- Xmatrix computation
- PCA computation
- kmeans
- creation of class averages

prepares hard disk for execution (`dvunfold`), and execution itself (can take some time)

creates a new classification project. We can pass directly the values for table (`-t`), data (`-d`), mask (`-d`), and also detail computational orders (as sym), in this case “`-bin1`”


Command Window
New to MATLAB? Watch this Video, see Examples, or read Getting Started.

```
----------------------------------------------------------
Computing subaverages
----------------------------------------------------------

[subaverages] Starting the averaging for 2 (new) merged subreferences.

----------------------------------------------------------
Going for subaverage associated to (new) merged subreference 1
Output will be written in ./pcBigmask/results/ite_0001/ccmatrix/subaverage_ite_0001_ref_001_sref_01.em
fmask compensation method: "table"
 Table contained 86 particles; averaging 86 tags:

[average] Time for processing of first particle:  0.01 (s)

1,3,4,8,9,10,11,12,13,14,16,17,19,22,23,24,28,29,30,31,33,36,37,38,39,
40,41,43,48,49,50,52,53,55,56,58,61,62,64,65,68,69,70,71,75,78,80,81,83,84,
87,88,90,92,93,94,95,96,97,98,99,100,104,105,107,109,110,111,112,113,115,116,117,120,122,
126,127,130,133,134,136,139,140,141,142,143,
[average:loop] Loop on particles in single core finished in 0.826792 seconds.


Skipping fourier compensation steps
[subaverages]
Going for subaverage associated to (new) merged subreference 2
Output will be written in ./pcBigmask/results/ite_0001/ccmatrix/subaverage_ite_0001_ref_001_sref_02.em
fmask compensation method: "table"
 Table contained 58 particles; averaging 58 tags:

[average] Time for processing of first particle:  0.01 (s)

2,5,6,7,15,18,20,21,25,26,27,32,34,35,42,44,45,46,47,51,54,57,59,60,63,
66,67,72,73,74,76,77,79,82,85,86,89,91,101,102,103,106,108,114,118,119,121,123,124,125,
128,129,131,132,135,137,138,144,
[average:loop] Loop on particles in single core finished in 0.476671 seconds.


Skipping fourier compensation steps
[subaverages]
done with subaverages
Kmeans took  2.05 seconds

 Done with ccmatrix-related computations of project pcBigmask, ref 1 ite 1,

>>
```

After execution of the project you recover control of the matlab shell.

and results can be accessed from the command line:

```
>>ddb pcBigmask:subaverage:sref=* -pv
```

command
(dynamo data base)

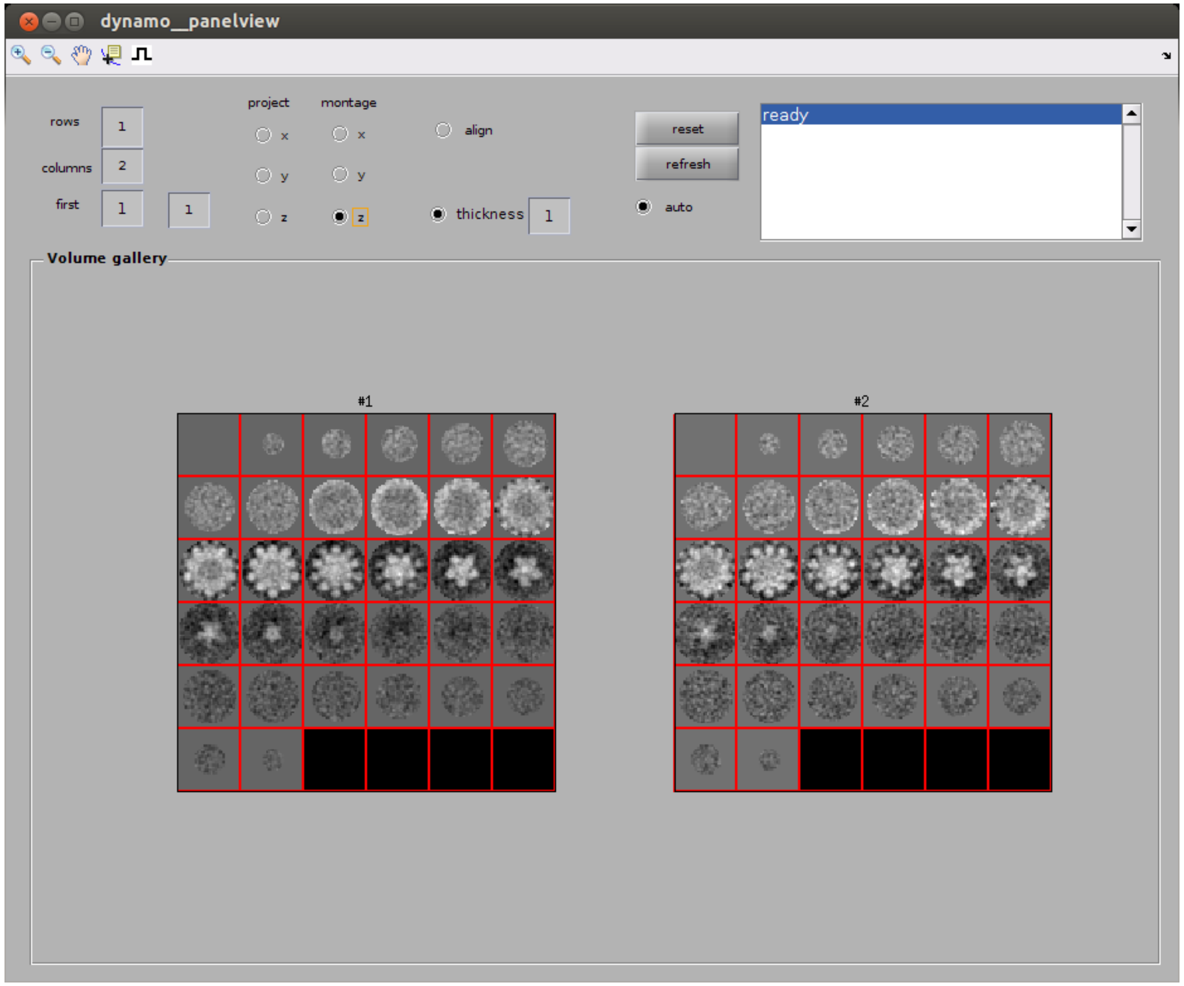
What to do with the retrieved item
in this case: send to `panelview` browser

in which project we look

which item class we look for

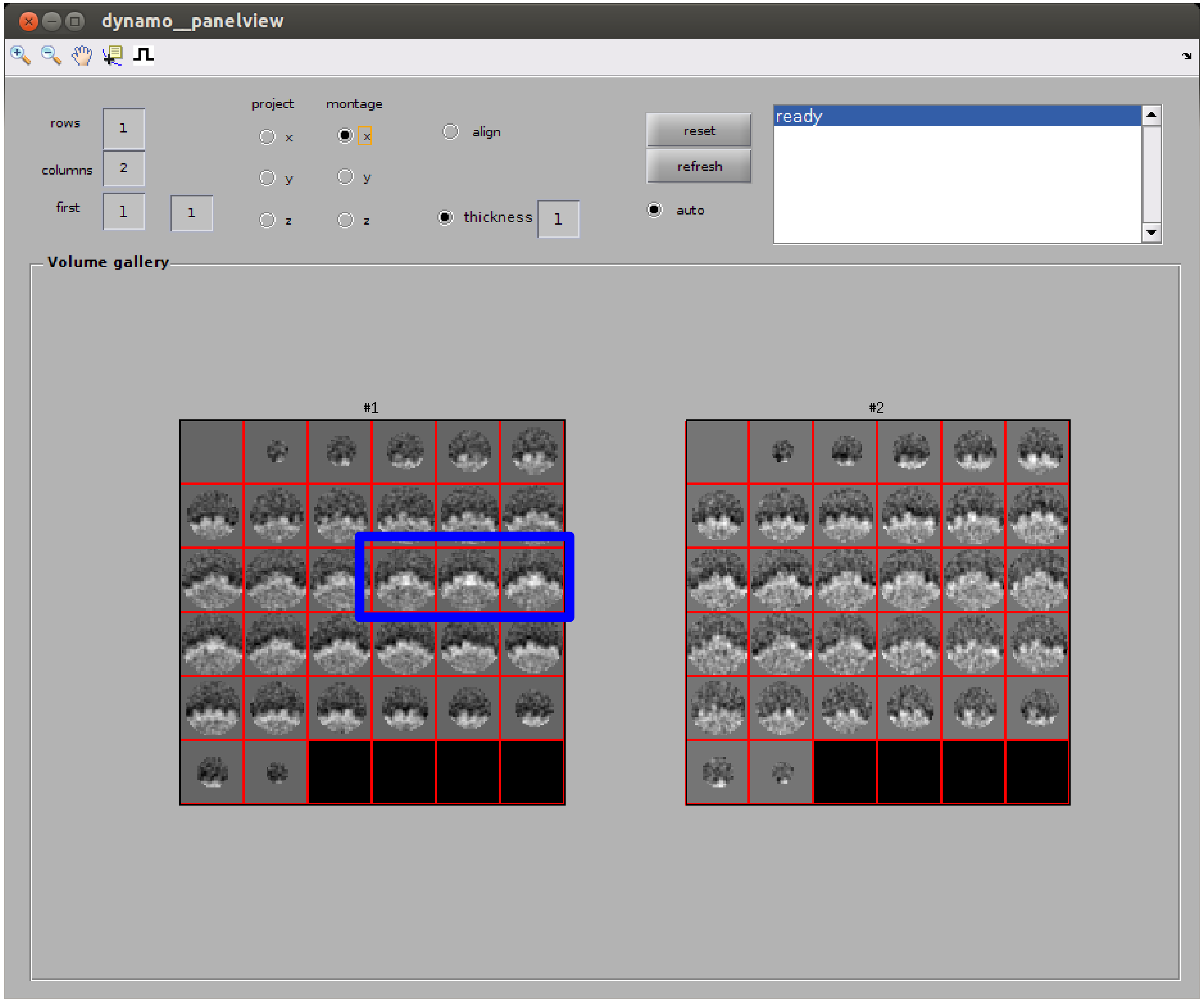
an identifier for the items that we want
(in this case “sref=*” meaning all subreferences)

type `doc ddb` for a complete description of the syntax of `ddb`.



We only have two subreferences (class averages) because the flag `-step all` was using default parameters.

Check
`doc dvccmatrix`
to see how to change the parameters from the command line: number of subreferences, identity of components to use, fine tuning of the Xmatrix computation, etc...

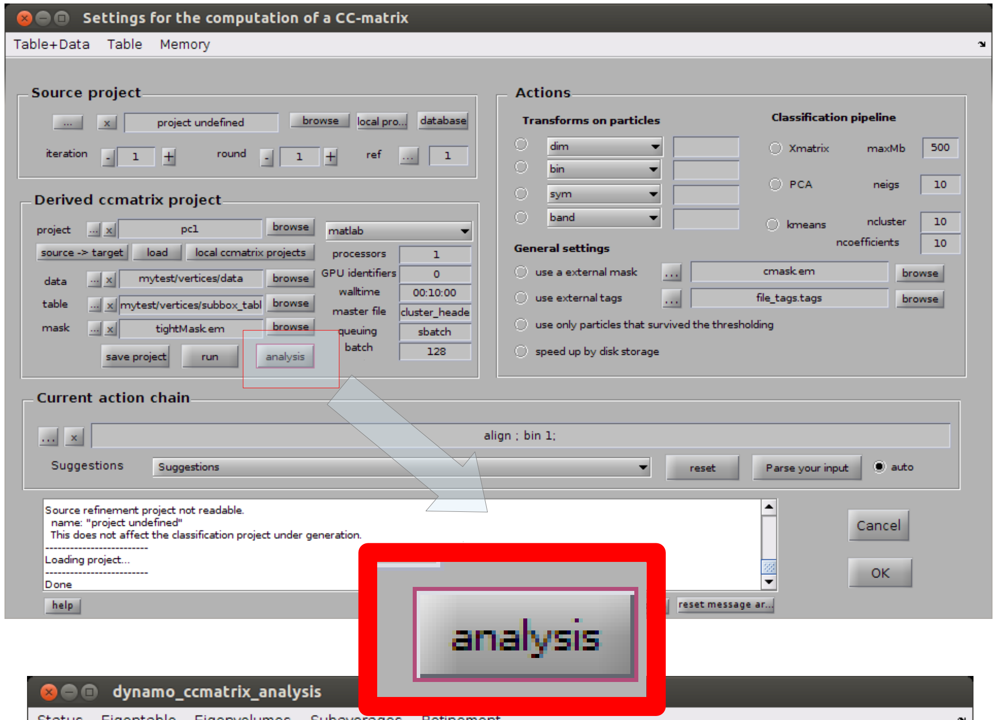


the wider mask is still able to classify according to the central densities.
Still, the classification is of less quality than the previous one (less contrast of the central embedded density)

In any case, it is easy to access the ccmatrix GUIs even if you started working from the command line

To open the ccmatrix design GUI:
(to compute a ccmatrix)

```
dgui pc1
```



Source project: project undefined; iteration 1; round 1; ref 1

Derived ccmatrix project: project pc1; data mytest/vertices/data; table mytest/vertices/subbox_tabl; mask tightMask.em

matlab; processors 1; GPU identifiers 0; walltime 00:10:00; master file cluster_heade; queuing sbatch; batch 128

Transforms on particles: dim; bin; sym; band

Classification pipeline: Xmatrix, maxMb 500; PCA, neigs 10; kmeans, ncluster 10, ncoefficients 10

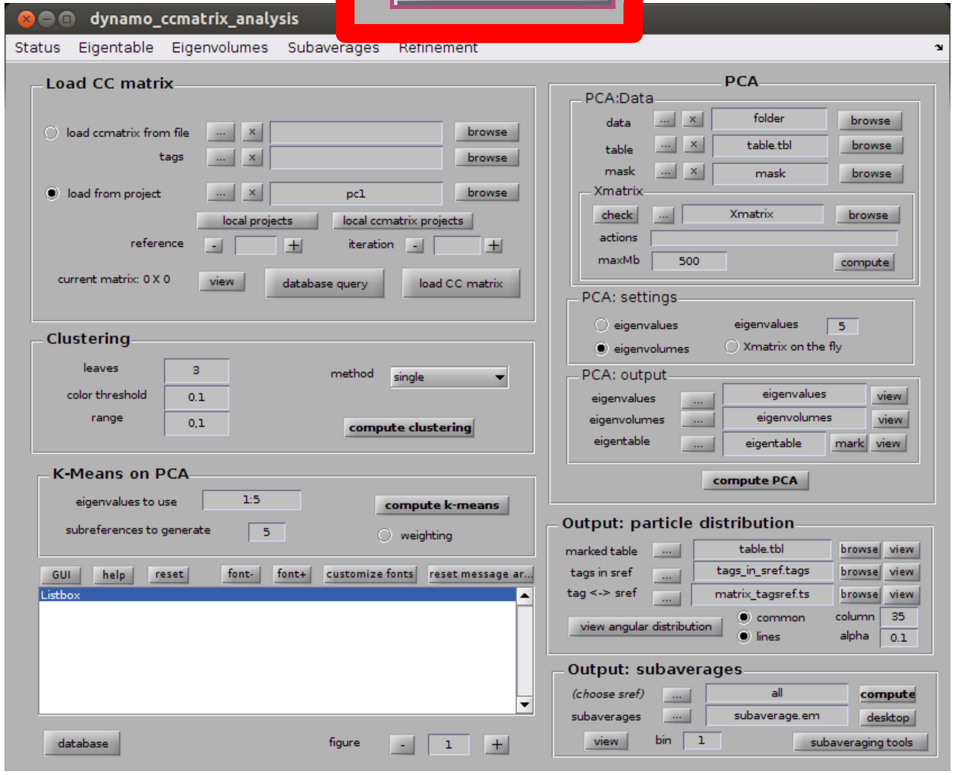
General settings: use a external mask cmask.em; use external tags file_tags.tags; use only particles that survived the thresholding; speed up by disk storage

Current action chain: align ; bin 1; — Suggestions — [✓ auto]

```
Source refinement project not readable.
 name: "project undefined"
 This does not affect the classification project under generation.
-------------------------
Loading project...
-------------------------
Done
```

To open the ccmatrix analysis GUI
(for PCA after a ccmatrix has been computed):

```
dccmatrix_analysis -p pc1
```



Load CC matrix: load ccmatrix from file; tags; [✓ load from project] pc1; reference; iteration; current matrix: 0 X 0

Clustering: leaves 3; color threshold 0.1; range 0.1; method single

K-Means on PCA: eigenvalues to use 1:5; subreferences to generate 5; weighting

Listbox

figure 1

PCA:Data: data folder; table table.tbl; mask mask; 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

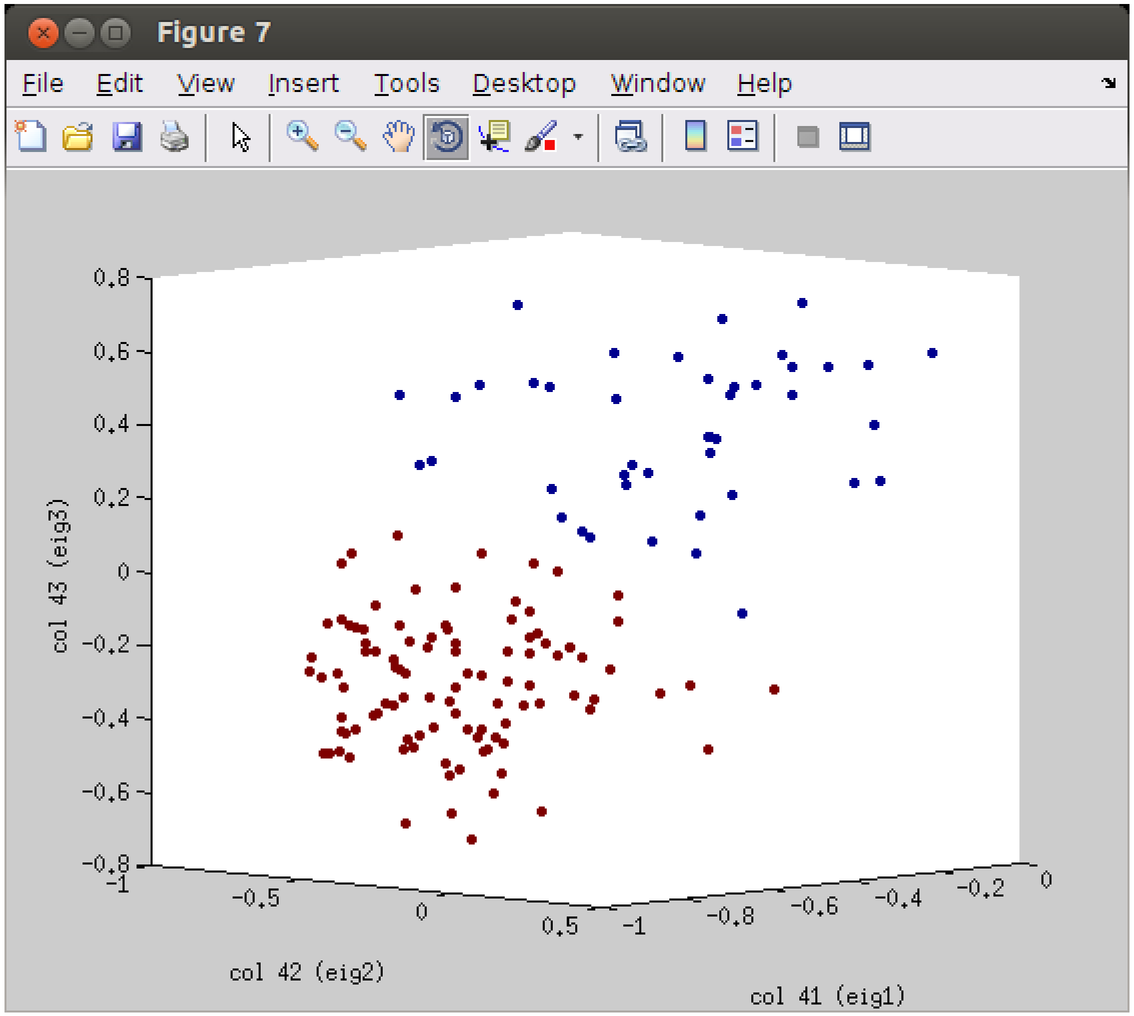
A final exercise to get familiar with command line driven classification:

Repeat the classification using the symmetry c5 expected form the vertices

```
>> dvccmatrix pc5 -d mytest/vertices/data -t
mytest/vertices/subbox_table.tbl -m tightMask.em -steps all -bin 1 -sym c5;

>> dvunfold pc5

>> pc5
```

We depict now (for instance) a scatterplot of the eigencomponents superposed to a colorcode of the classification

```
ddb pc5:eigentable -eigenvalues
```



PART III

Multireference analysis

We will create a multireference project from the command line.

this requires 10 lines of code (or just one if you really like enormous lines)

```
>>dvpr pmulti -nref 3 -d  mytest/vertices/data -m maskBig.em
>>dvput pmulti cmask tightMask.em;
>>dwrite_multireference fullAverage.em  template folder_seeds -refs 1:3 -noise 0.5
>>dwrite_multireference mytest/vertices/subbox_table.tbl  table folder_seeds -refs 1:3
>>dwrite_multireference full fmask folder_seeds -refs 1:3
>>dvput pmulti seeds folder_seeds;
>>dvput pmulti -cr 0 -cs 1 -ir 0 -is 1 -limm 2 -dim -32 -rf 0 -sym c5 ite_r1 10;
>>dvunfold pmulti;
>>pmulti
```

We comment them in the following slides

We create a project called `pmulti`, which will have 3 references.

Note:
By defaulft, a project initiated with several references will run a MultiReference Analysis, letting particles to swap between multireference channels after each refinement iterations.

```
>>dvpr pmulti -nref 3 -d  mytest/vertices/data -m maskBig.em
>>dvput pmulti cmask tightMask.em;
```

we can pass already in this command all the project parameters that we want. You can pass them with the full parameter name, or use the syntax of `dvpr` for short flags

In this case we just pass the parameters '`data_folder`' (shortened to the flag '`-d`') and '`file_mask`' (shortened to the flag '`-m`').

Type `doc dvpr` to see a list of flags accepted by this command.

The rest of the parameters can be passed with dvput.
Check dvhelp to see a list of parameter names and shortnames.
In this case we are passing a classification mask.

Remember that in a MRA project two masks are typically used:
* Alignment mask to drive the alignment.
* Classification mask to focus on the part where we expect structural hetereogeneity.

The next lines format the “seeds” (the sets of initial tables, references and fourier masks) as required by a multiple reference alignment.

In the easiest input format, for any of these parameters (`file_table_initial`,`file_tempalte_initial`, `file_fmask_initial`) you can provide the name of a folder that contains a set of files for each reference.

The files have to be named following a given naming convention.
For instance, in a folder called “`myFolder`”, you could have two files:

```
myFolder/table_initial_ref_001.tbl
myFolder/table_initial_ref_002.tbl
```

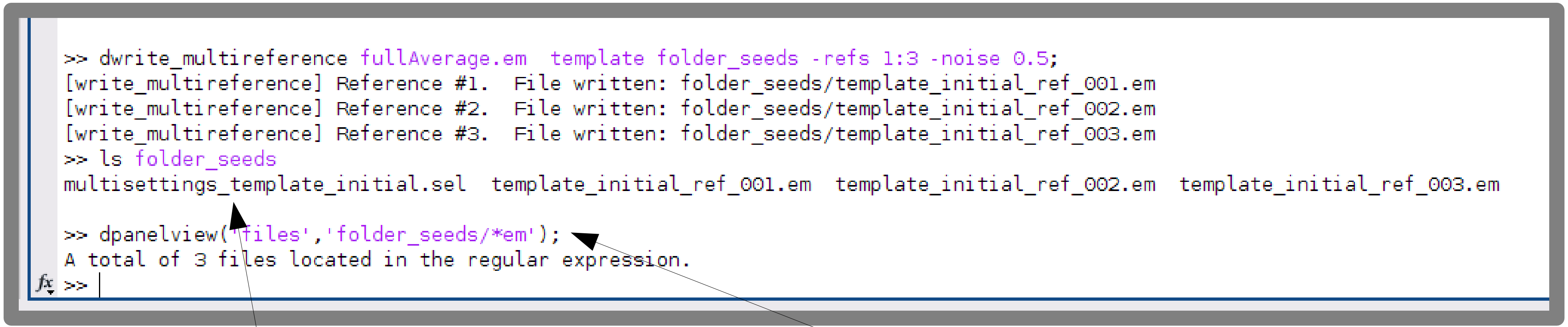
and then you must pass `myFolder` as value of the project parameter `file_table_initial`

You can prepare all your files manually, and the use dvput to enter them into the project.
However, the command `dwrite_multireference` automatizes many tasks
(file creation and file name formatting) that you typically need when dealing with the seeds of a MRA project

```
>>dwrite_multireference fullAverage.em  template folder_seeds -refs 1:3 -noise 0.5
```

This command:

- creates three copies of the file `fullAverage.em` and adds random noise of amplitude 0.5;
- writes them in the folder `folder_seeds` with the right naming convention to serve as template in a project

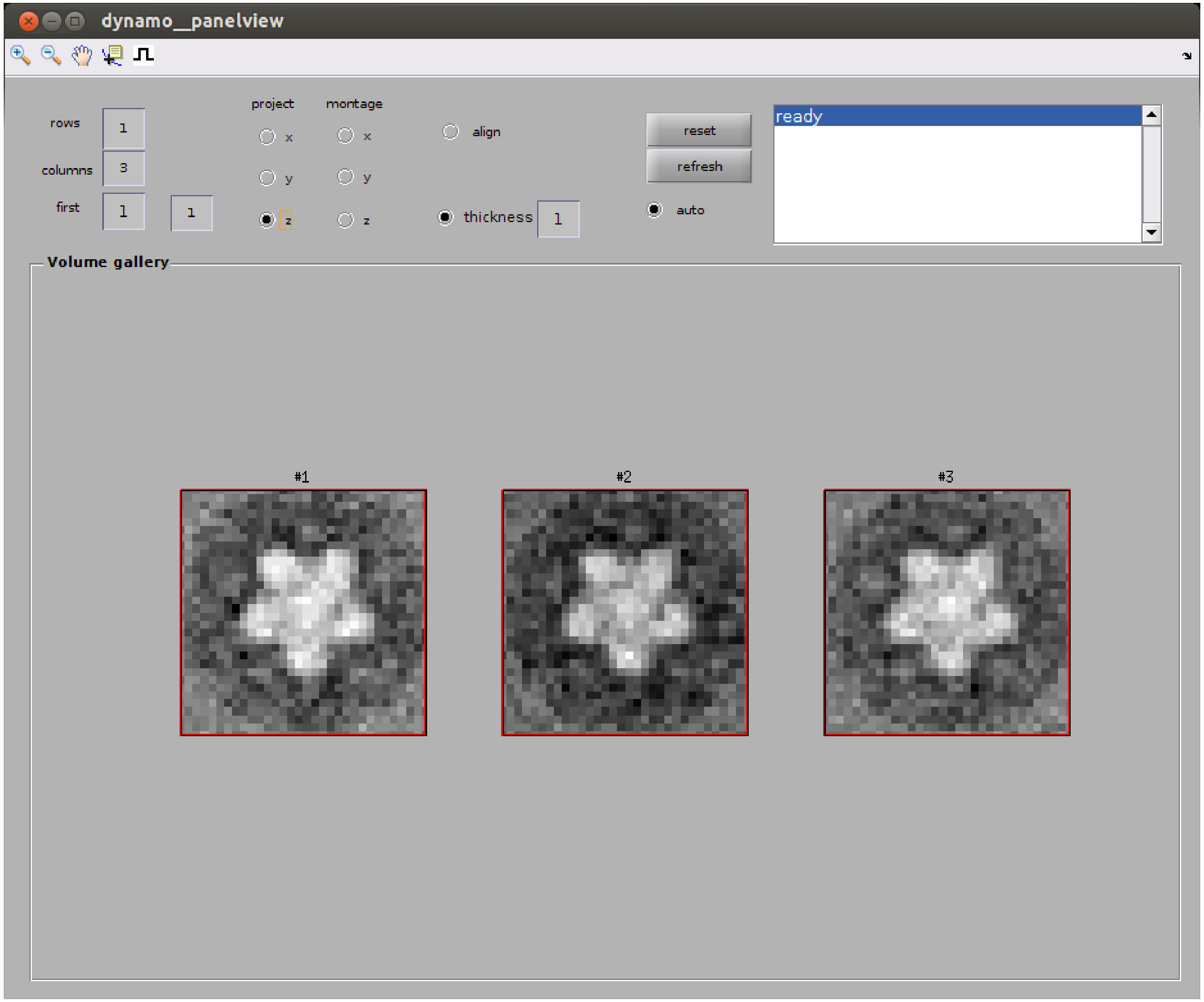


```
>> dwrite_multireference fullAverage.em  template folder_seeds -refs 1:3 -noise 0.5;
[write_multireference] Reference #1.  File written: folder_seeds/template_initial_ref_001.em
[write_multireference] Reference #2.  File written: folder_seeds/template_initial_ref_002.em
[write_multireference] Reference #3.  File written: folder_seeds/template_initial_ref_003.em
>> ls folder_seeds
multisettings_template_initial.sel  template_initial_ref_001.em  template_initial_ref_002.em  template_initial_ref_003.em

>> dpanelview('files','folder_seeds/*em');
A total of 3 files located in the regular expression.
>> 
```

Footnote:
`dwrite_multireference` creates also a .sel file that can be used to define multiple files as input for a *Dynamo* project parameter. Both ways are equally valid, so we just ignore the .sel file during this tutorial

depiction of the contents, see next slide

So, our starting maps are three different realizations of the same level of noise imposed on the average of all the particles



The next lines are also commodities to create and format multiple files as Dynamo input parameters:

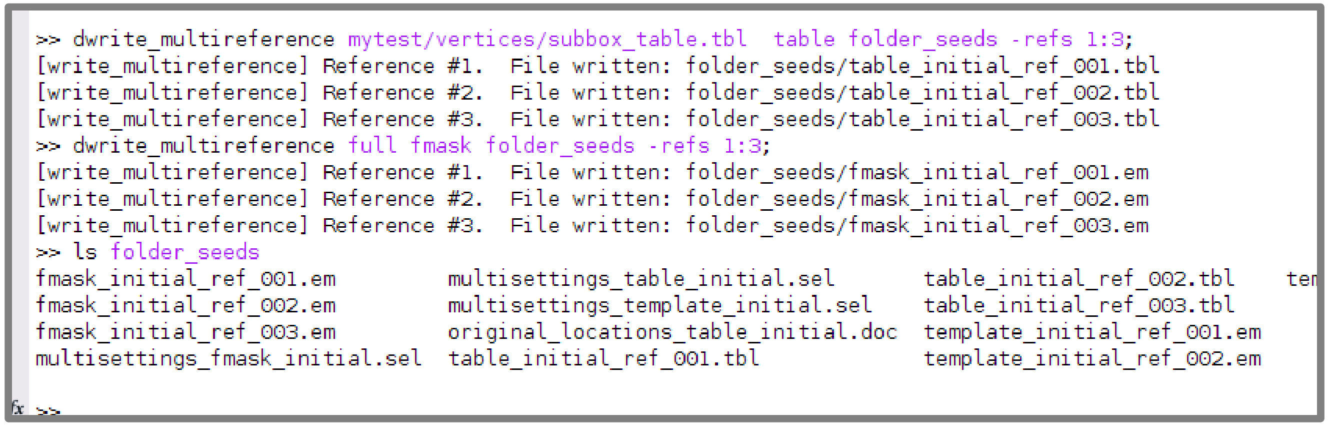
Creates three copies of the initial table, formats them (and sets them in the same folde as before)

```
>>dwrite_multireference mytest/vertices/subbox_table.tbl  table folder_seeds -refs 1:3

>>dwrite_multireference full fmask folder_seeds -refs 1:3

>>dvput pmulti seeds folder_seeds;
```

creates a set of full fourier masks and puts them into the same folder used previously.

shorthand to indicate that all “seeds” (initial tables, templates and fmasks) are in the same folder:

`folder_seeds`



```
>> dwrite_multireference mytest/vertices/subbox_table.tbl  table folder_seeds -refs 1:3;
[write_multireference] Reference #1.  File written: folder_seeds/table_initial_ref_001.tbl
[write_multireference] Reference #2.  File written: folder_seeds/table_initial_ref_002.tbl
[write_multireference] Reference #3.  File written: folder_seeds/table_initial_ref_003.tbl
>> dwrite_multireference full fmask folder_seeds -refs 1:3;
[write_multireference] Reference #1.  File written: folder_seeds/fmask_initial_ref_001.em
[write_multireference] Reference #2.  File written: folder_seeds/fmask_initial_ref_002.em
[write_multireference] Reference #3.  File written: folder_seeds/fmask_initial_ref_003.em
>> ls folder_seeds
fmask_initial_ref_001.em        multisettings_table_initial.sel      table_initial_ref_002.tbl    ter
fmask_initial_ref_002.em        multisettings_template_initial.sel   table_initial_ref_003.tbl
fmask_initial_ref_003.em        original_locations_table_initial.doc template_initial_ref_001.em
multisettings_fmask_initial.sel table_initial_ref_001.tbl            template_initial_ref_002.em

fx >>
```

The next line codes all the numerical parameters in this experiment that depart from default values.

dimensionality of the particles
Can be used to induce a binning to
accelerate calculations

limit of shifts for paticles tune to mode 2:
(measure shifts from the center of the particles)

round 1 will have 10 iterations
(other rounds are initialized to zero)

```
>>dvput pmulti -cr 0 -cs 1 -ir 0 -is 1 -limm 2 -dim -32 -rf 0 -sym c5 ite_r1 10;
```

angular parameters

In this case (and just to accelerate computations),
 we are not letting particles move, as we want to gauge
the capacity of MRA to induce a classification.

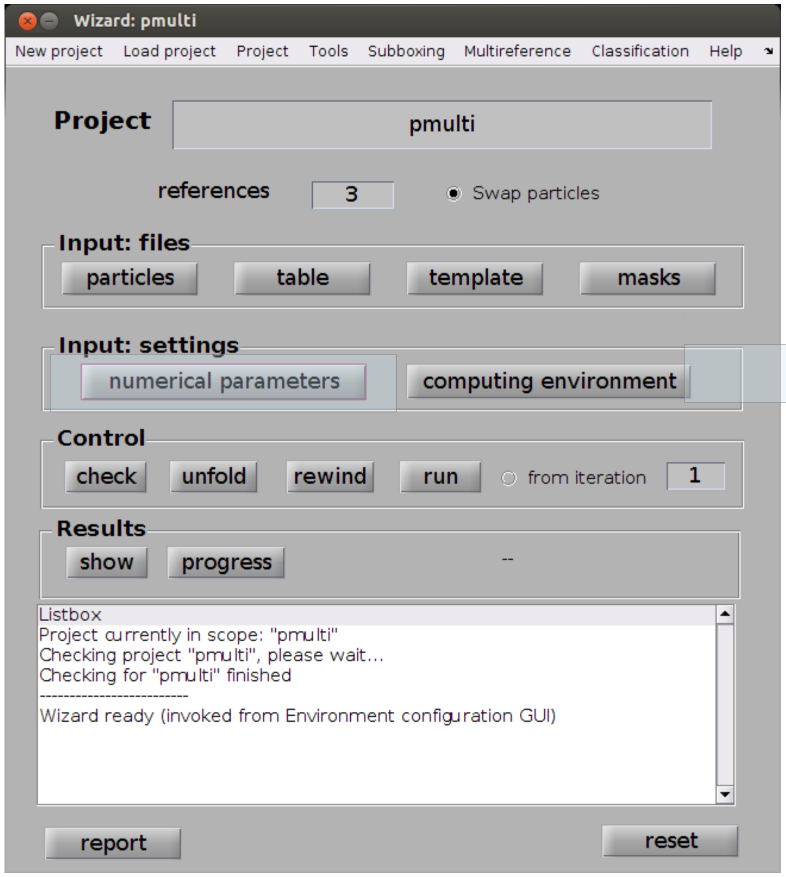
- `cr: cone_range` (i.e. apertute of cone or orientations)
- `cs: cone_sampling` (discretization interval)
- `ir: inplane_range` (azymutal angle, i.e., rotations about the axis)
- `is: inplane_sampling`

explicitely apply the c5 symmetry
operator expected on each vertex

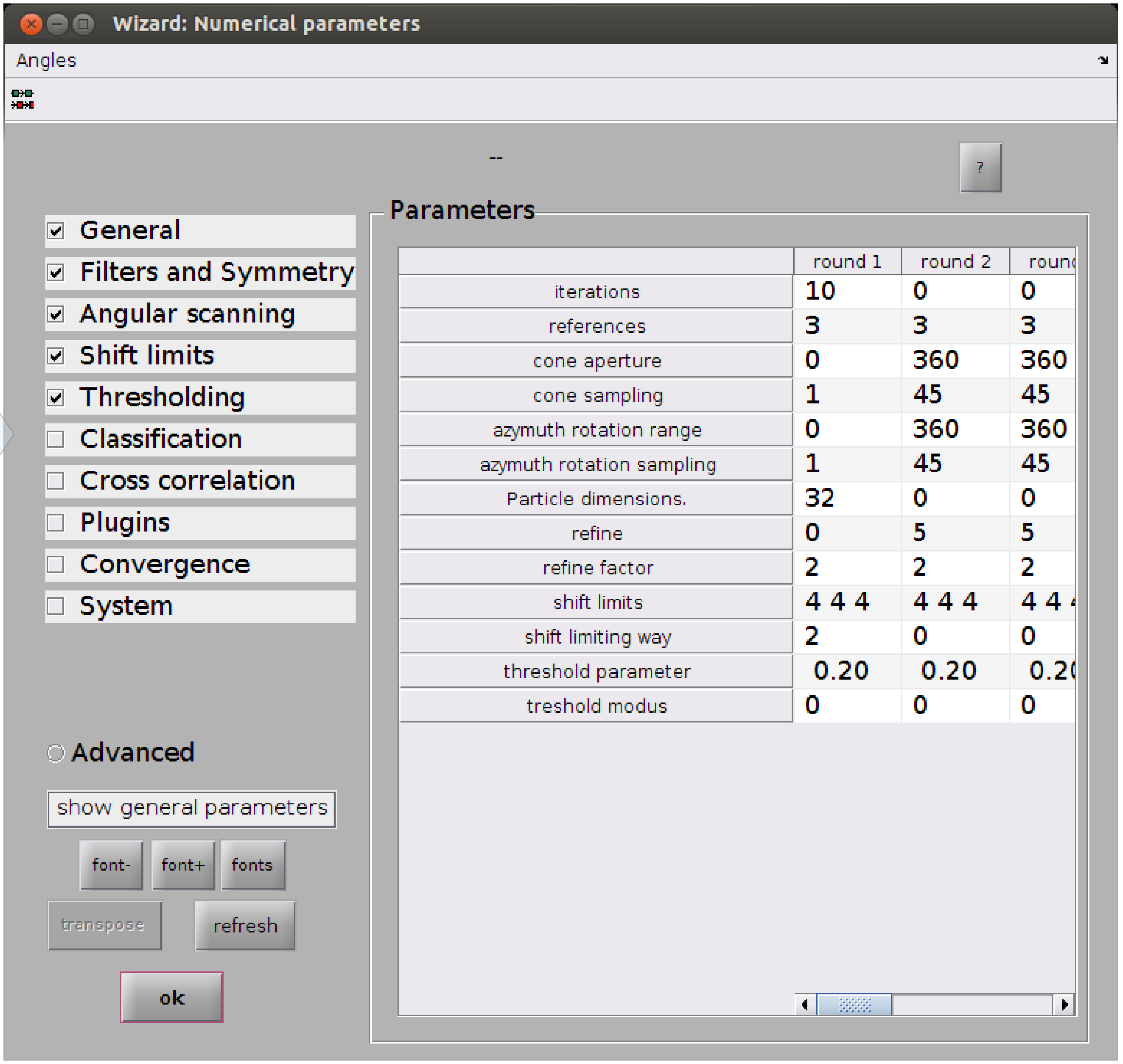
No multigrid refinement

But remember that you can intertwin command line with GUI representation at convenience

```
>>dcp pmulti
```



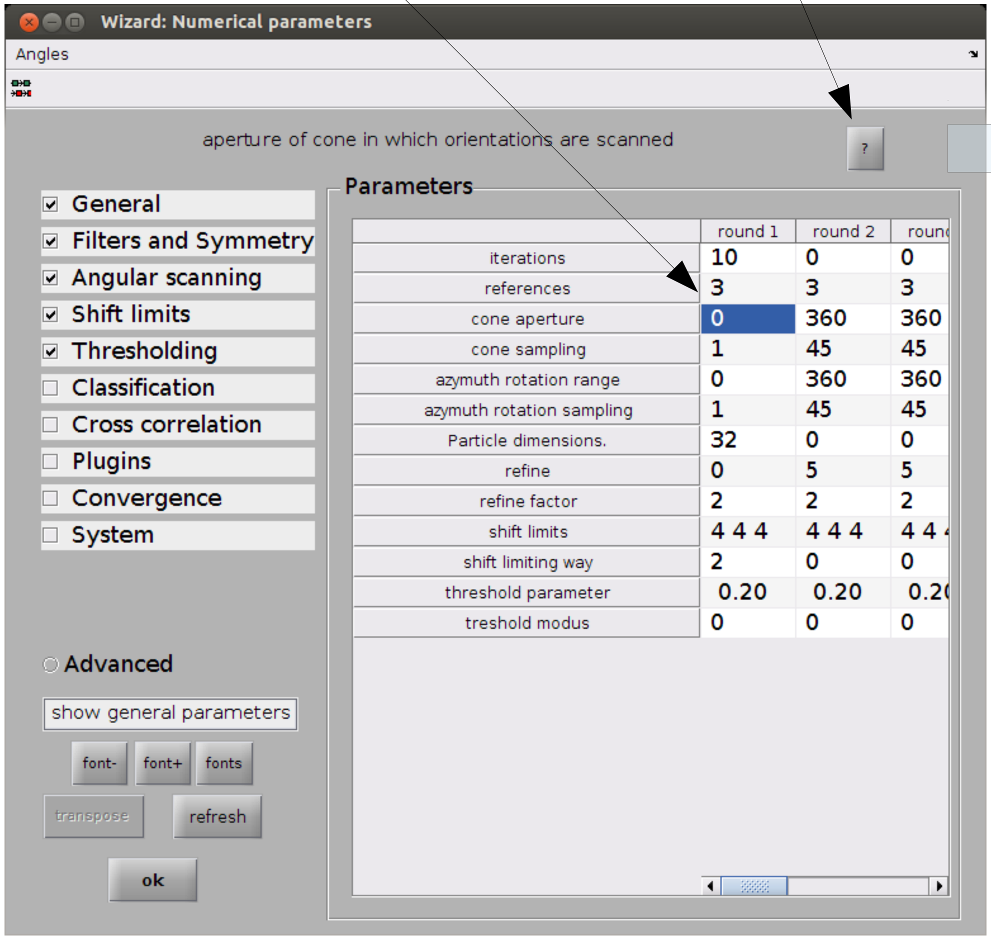
```
Listbox
Project currently in scope: "pmulti"
Checking project "pmulti", please wait...
Checking for "pmulti" finished
-------------------------
Wizard ready (invoked from Environment configuration GUI)
```



	round 1	round 2	roun
iterations	10	0	0
references	3	3	3
cone aperture	0	360	360
cone sampling	1	45	45
azymuth rotation range	0	360	360
azymuth rotation sampling	1	45	45
Particle dimensions.	32	0	0
refine	0	5	5
refine factor	2	2	2
shift limits	4 4 4	4 4 4	4 4
shift limiting way	2	0	0
threshold parameter	0.20	0.20	0.2
treshold modus	0	0	0

in the Wizard for numerical parameters you can see all available parameters, edit them and get help on selected parameters press [?]

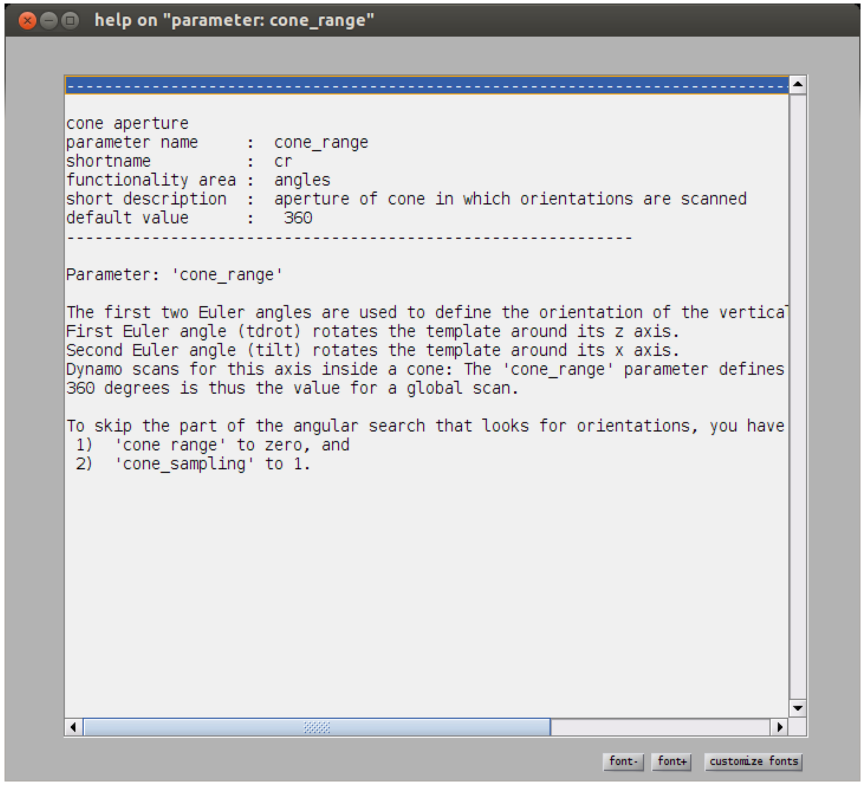
select a parameter



aperture of cone in which orientations are scanned

- [✓] General
- [✓] Filters and Symmetry
- [✓] Angular scanning
- [✓] Shift limits
- [✓] Thresholding
- [ ] Classification
- [ ] Cross correlation
- [ ] Plugins
- [ ] Convergence
- [ ] System

	round 1	round 2	roun
iterations	10	0	0
references	3	3	3
cone aperture	0	360	360
cone sampling	1	45	45
azymuth rotation range	0	360	360
azymuth rotation sampling	1	45	45
Particle dimensions.	32	0	0
refine	0	5	5
refine factor	2	2	2
shift limits	4 4 4	4 4 4	4 4
shift limiting way	2	0	0
threshold parameter	0.20	0.20	0.2
treshold modus	0	0	0



```
---------------------------------------------------------------------------

cone aperture
parameter name     :  cone_range
shortname          :  cr
functionality area :  angles
short description  :  aperture of cone in which orientations are scanned
default value      :   360
------------------------------------------------------------

Parameter: 'cone_range'

The first two Euler angles are used to define the orientation of the vertica
First Euler angle (tdrot) rotates the template around its z axis.
Second Euler angle (tilt) rotates the template around its x axis.
Dynamo scans for this axis inside a cone: The 'cone_range' parameter defines
360 degrees is thus the value for a global scan.

To skip the part of the angular search that looks for orientations, you have
 1)  'cone range' to zero, and
 2)  'cone_sampling' to 1.
```

A handy way to recover results is using the `dynamo_db` command.
This command allows a wide range of searches and actions on items in the database associated with a project (automatically generated by *Dynamo*).

Query:
What do we want to retrieve?

Action:
What do we want Dynamo to do with the retrieved items?

In this case: send to `mapview`

Command
Dynamo shorthand.
"`db`" stands for database.

```
ddb pmulti:a:ref=* -m
```

Project name

Item to retrieve:
"a" stands for average.

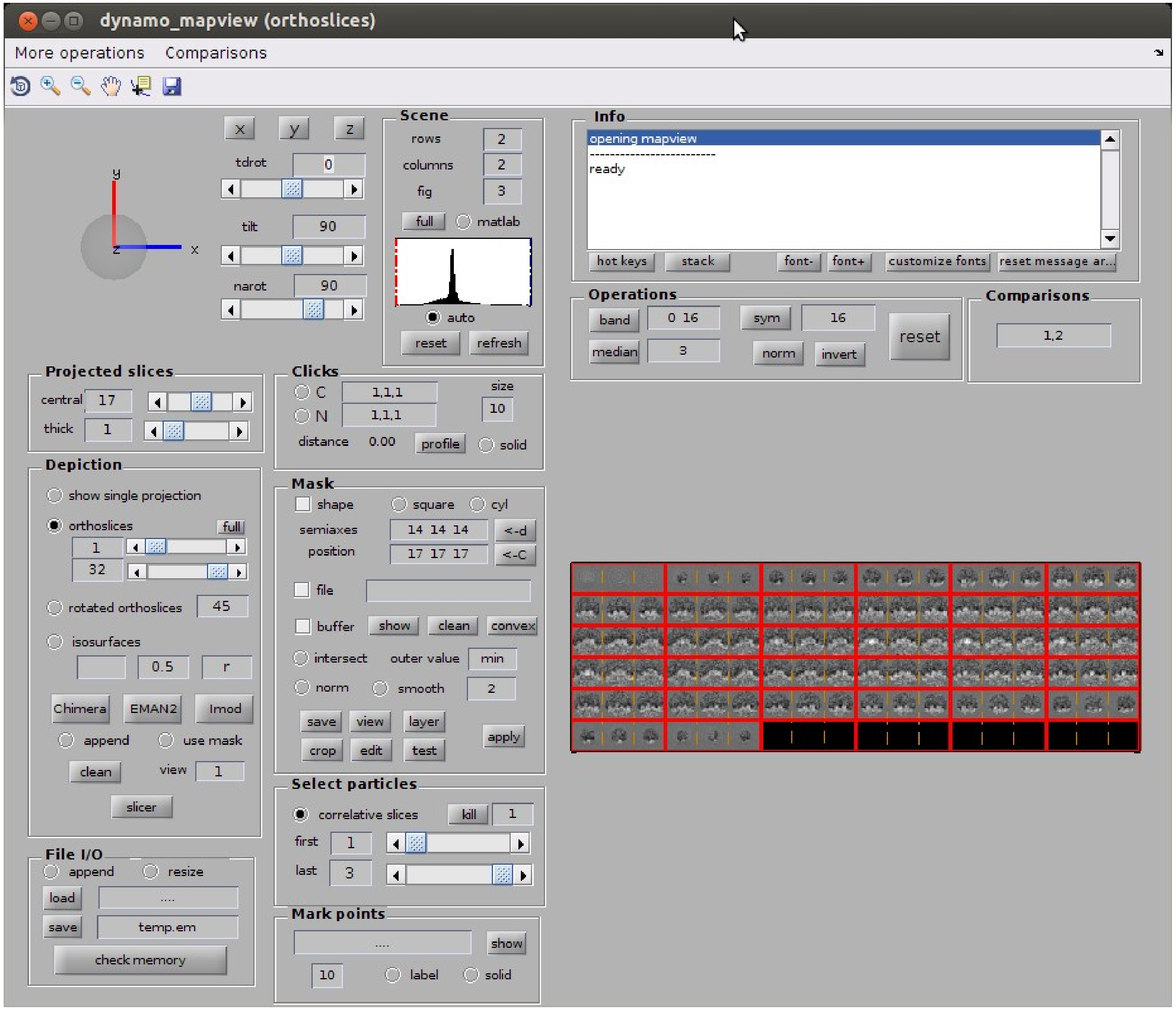
Identifier:
Which references we want to retrieve
(in this case "*": all of them).
We could have added an iteration identifier.

```
ddb pmulti:a:ite=10:ref=* -m
```

but for the item "average" Dynamo accesses by default the last available iteration

Type `doc db` to see more options of `ddb`.

You should see something like this:



opening mapview
----
ready

tdrot 0
tilt 90
narot 90

rows 2
columns 2
fig 3

band 0 16
sym 16
median 3
Comparisons 1,2

Projected slices: central 17, thick 1

Depiction:
show single projection
[✓ orthoslices] full 1 32
rotated orthoslices 45
isosurfaces 0.5 r
append
use mask
view 1

File I/O: append, resize, load ...., save temp.em

Clicks: C 1,1,1; N 1,1,1; size 10; distance 0.00; solid

Mask: shape, square, cyl; semiaxes 14 14 14 <-d; position 17 17 17 <-C; file; buffer show clean convex; intersect; outer value min; norm; smooth 2

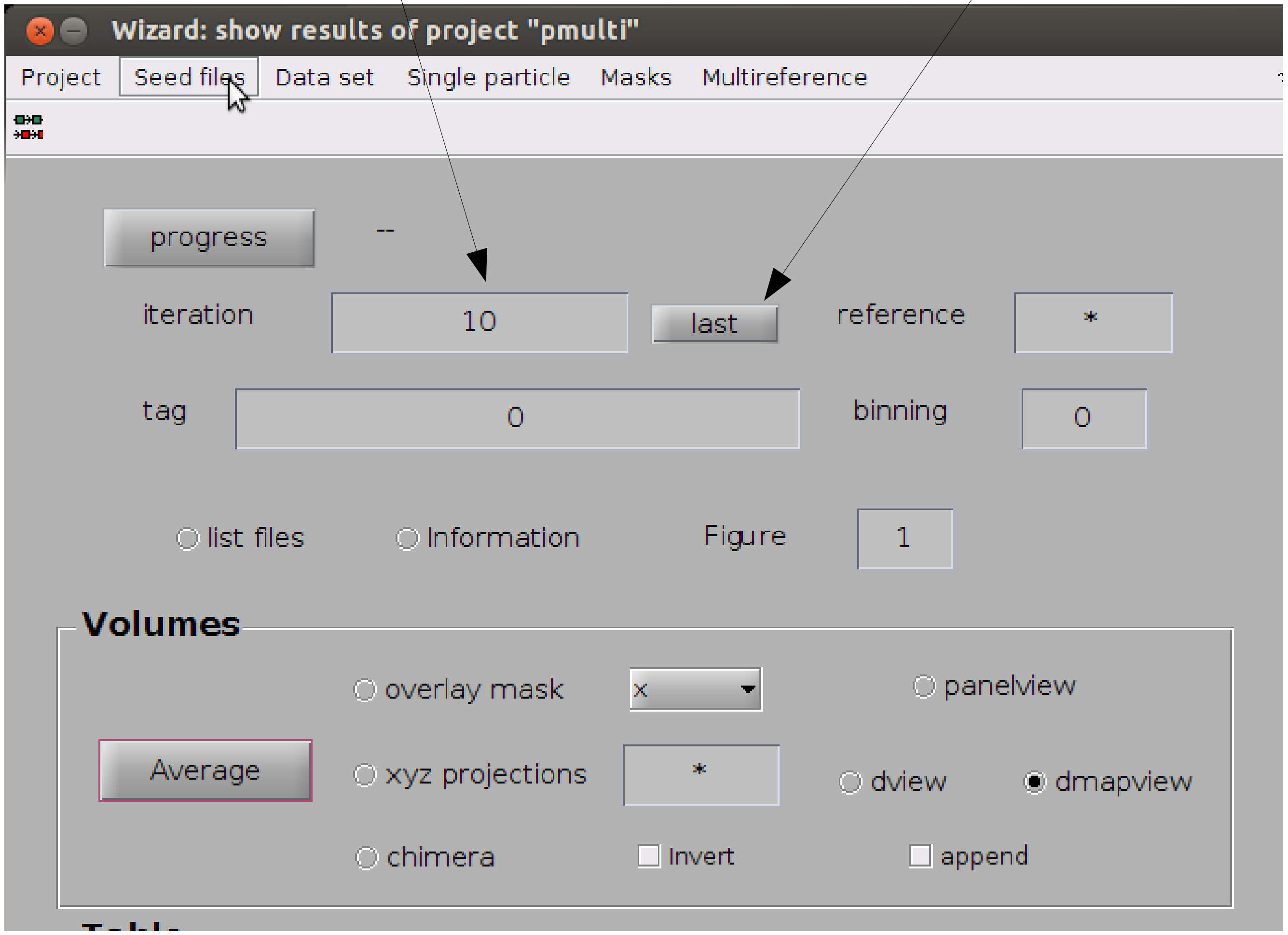
Select particles: [✓ correlative slices] kill 1; first 1; last 3

Mark points: ....; 10; label; solid

If you don't like the commandline tools, you can access the data also with GUIs

Type the iteration you want ot depict

… or press to find the last available



Wizard: show results of project "pmulti"

Project | Seed files | Data set | Single particle | Masks | Multireference

progress --

iteration: 10 | last | reference: *

tag: 0 | binning: 0

○ list files ○ Information Figure: 1

Volumes

Average

○ overlay mask: x

○ xyz projections: *

○ chimera

☐ Invert

○ panelview

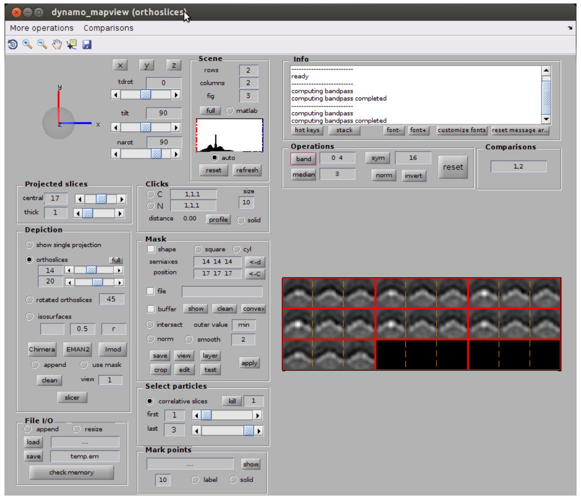
○ dview [✓ ● dmapview]

☐ append

Table

Choose all references

Select mapview
For visualization


dynamo_mapview (orthoslices)

More operations   Comparisons

Scene: rows 2, columns 2, fig 3

tdrot 0, tilt 90, narot 90

Info:

```
------------------------
ready
------------------------
computing bandpass
computing bandpass completed
------------------------
computing bandpass
computing bandpass completed
```

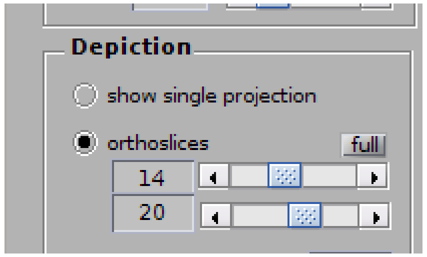
Operations: band 0 4, sym 16, median 3, norm, invert, reset

Comparisons: 1,2

Projected slices: central 17, thick 1

Clicks: C 1,1,1; N 1,1,1; size 10; distance 0.00

Mask: semiaxes 14 14 14; position 17 17 17

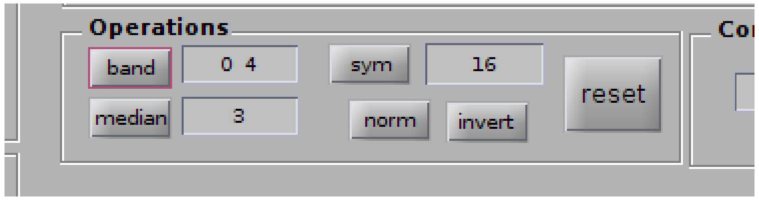

Depiction

show single projection

orthoslices [✓] full

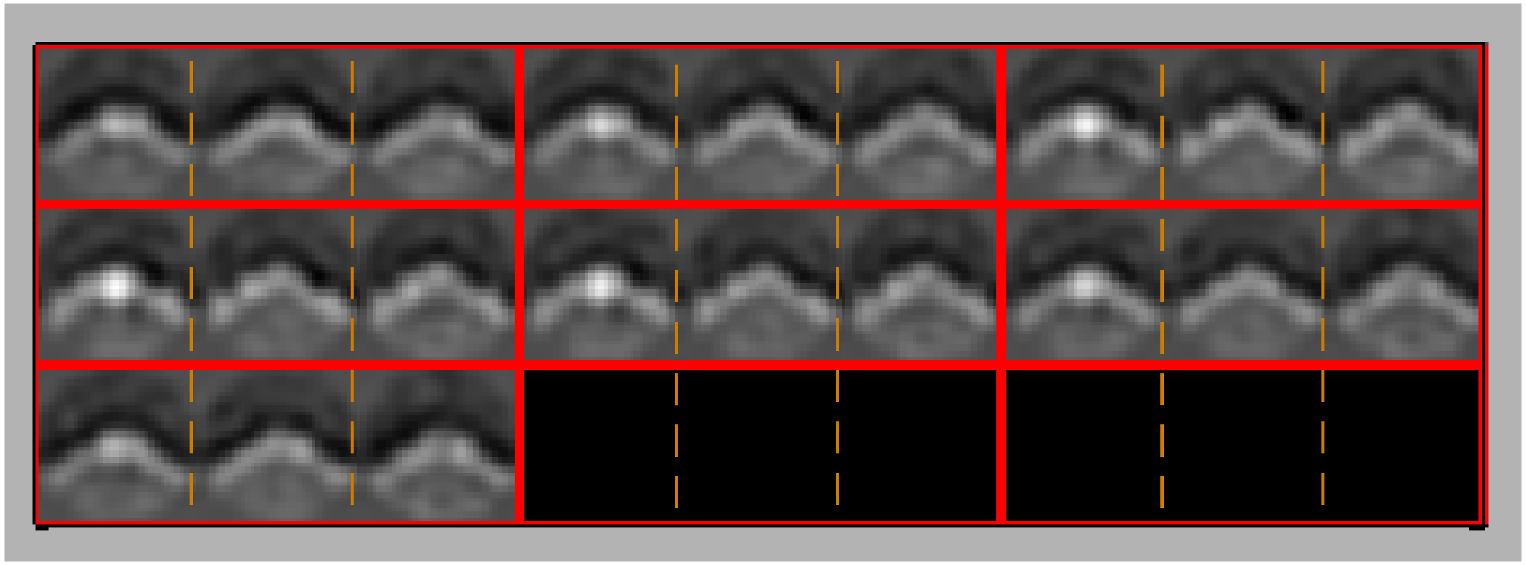
14

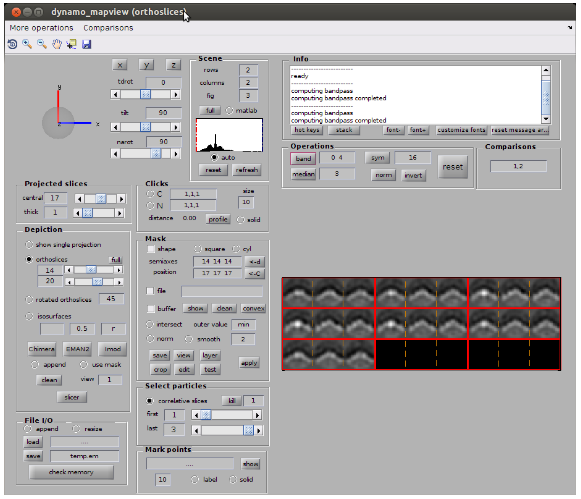
20


Operations

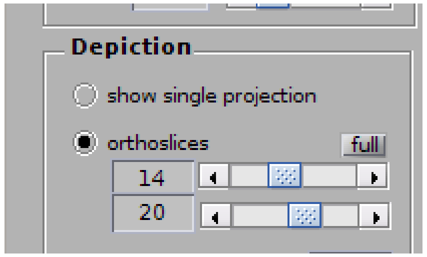
band 0 4 sym 16 reset

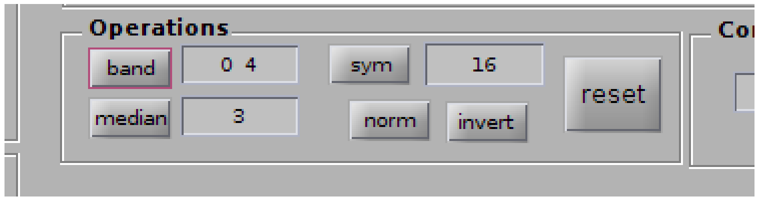
median 3 norm invert

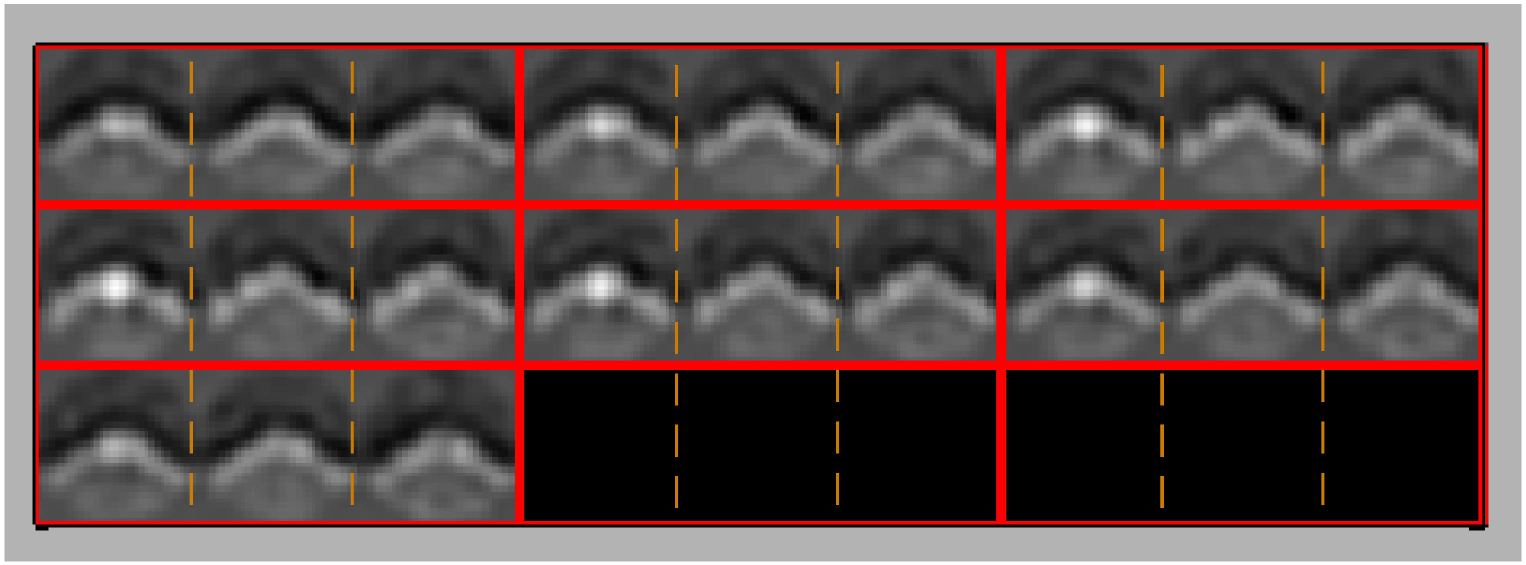




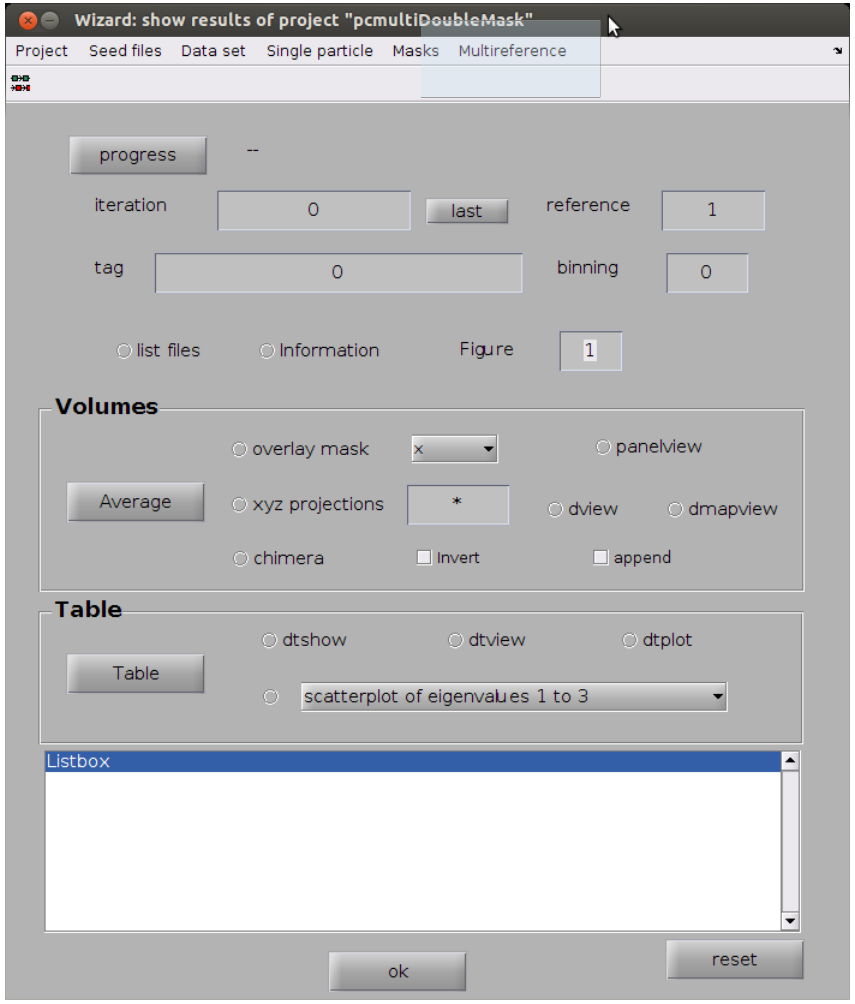
```
------------------------
ready
------------------------
computing bandpass
computing bandpass completed
------------------------
computing bandpass
computing bandpass completed
```







dcp pcmulti -show



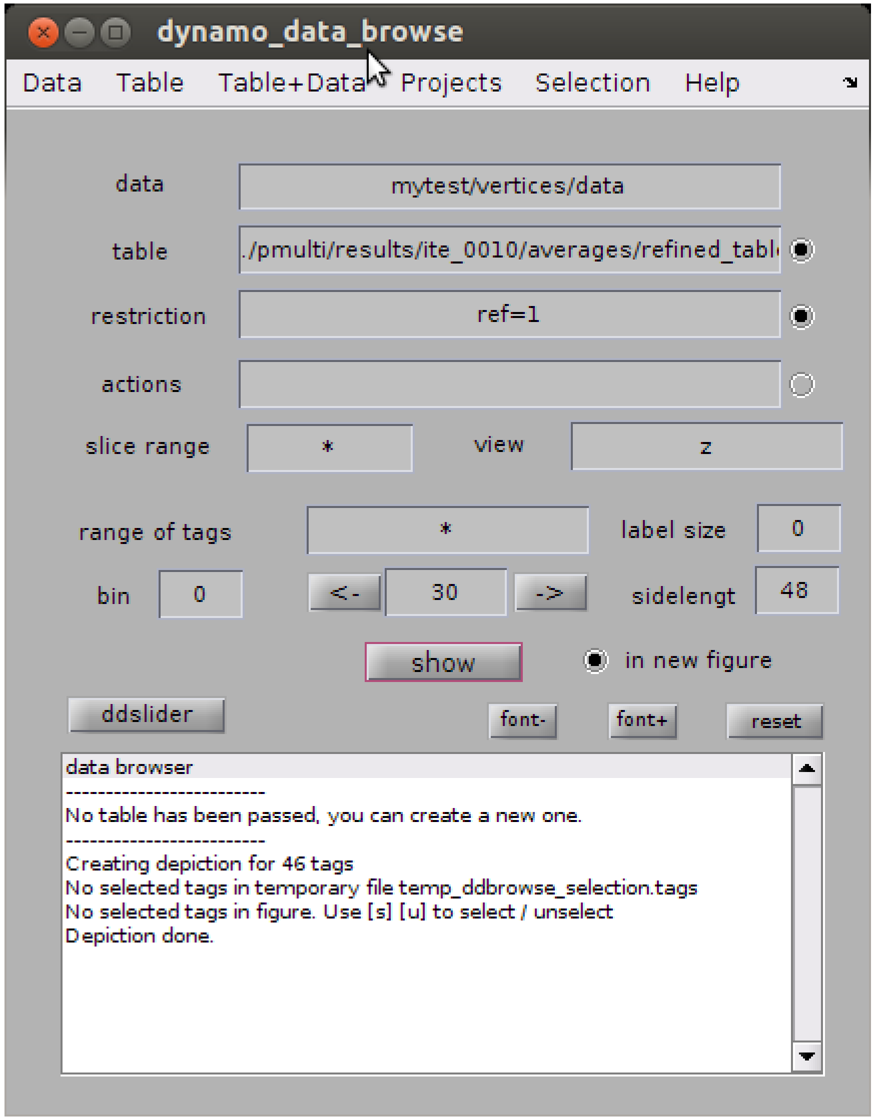
Can we check which particles belong to reference 1?

Possibly our first temptation could be to take a look on the particles themselves.

```
ddbrowse -p pmulti -ref 1
```

Calling ddbrowse with a -p flag ("project")
Automatically selects the refined_table
Found on that project at its last iteration.

It also fetches the right data folder.

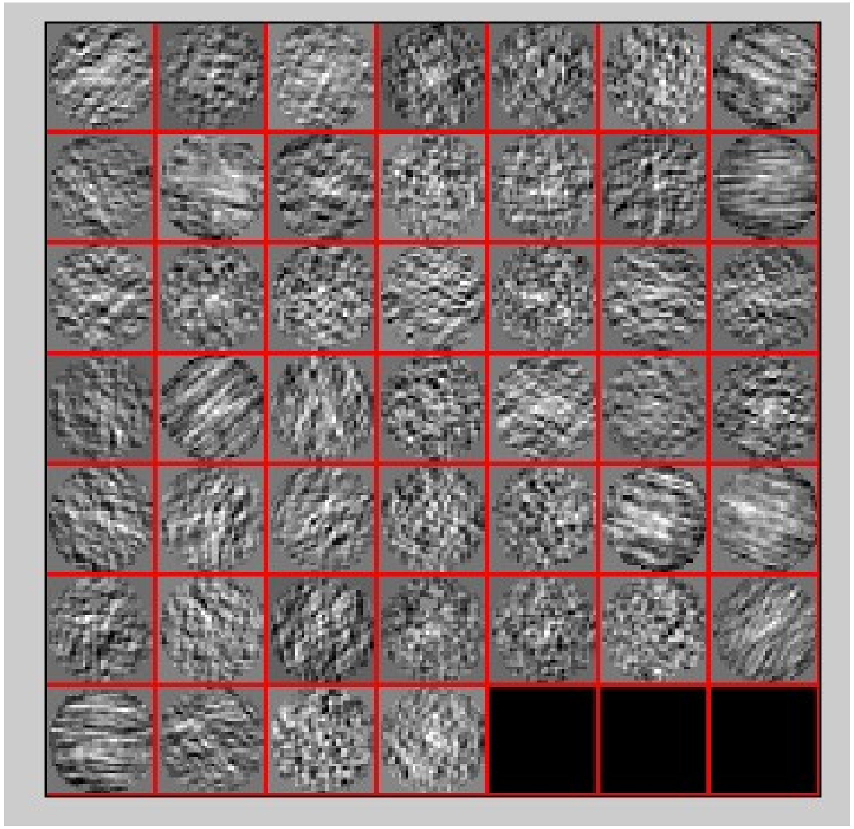
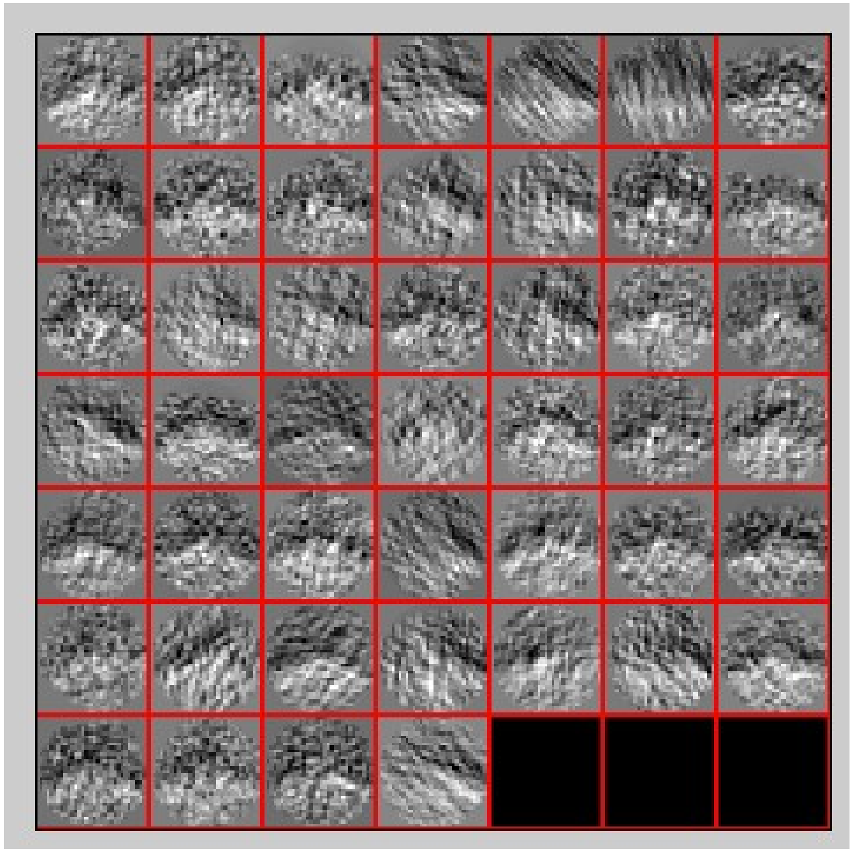


data: mytest/vertices/data
table: ./pmulti/results/ite_0010/averages/refined_tabl ●
restriction: ref=1 ●
actions: ○
slice range: * view: z
range of tags: * label size: 0
bin: 0 <- 30 -> sidelengt: 48
● in new figure

```
data browser
------------------------
No table has been passed, you can create a new one.
------------------------
Creating depiction for 46 tags
No selected tags in temporary file temp_ddbrowse_selection.tags
No selected tags in figure. Use [s] [u] to select / unselect
Depiction done.
```

Passing explicitely a "ref" also fills the "restriction" area, telling Dynamo that we are interested only on those particles marked as "belonging to reference 1" in the table (i.e. have the value "1" in column 34").

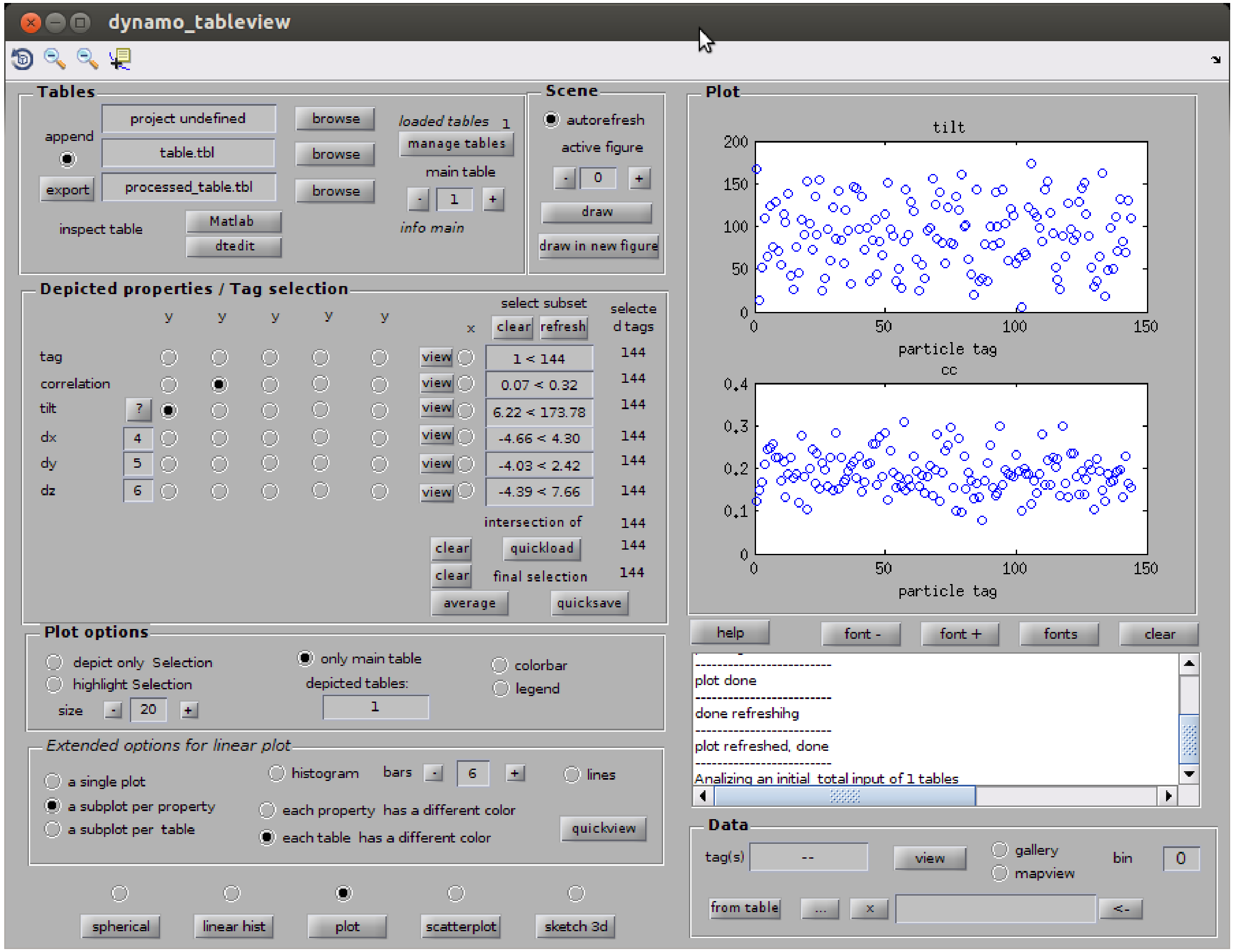
Remember that the table by itself would contain alignment information of ALL the particles against reference 1, not only of those that finally contribute to this reference channel.

… but it's kind of difficult to ascertain if this reference is really hitting the original class 2
(of vertex particles that are known by construction to contain an insertion)

Can we check the particles that belong into reference 1?

```
ddb pmulti:rt:ref=1 -v
```



project undefined
table.tbl
processed_table.tbl
loaded tables 1
main table 1
active figure 0

property	y	y	y	y	y	x	range	selected tags
tag							1 < 144	144
correlation		●					0.07 < 0.32	144
tilt	●						6.22 < 173.78	144
dx 4							-4.66 < 4.30	144
dy 5							-4.03 < 2.42	144
dz 6							-4.39 < 7.66	144

intersection of 144
144
final selection 144

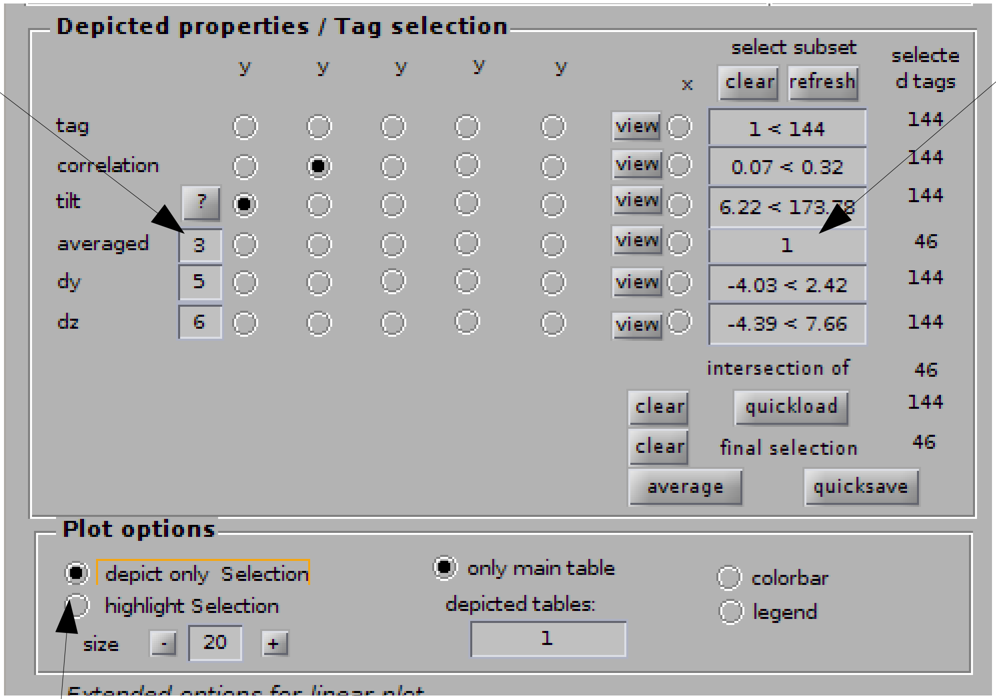
size 20
[✓ only main table]
depicted tables: 1
[✓ a subplot per property]
bars 6
[✓ each table has a different color]
[✓ plot]

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

tag(s) --
bin 0

First of all, we want to focus on those particles that actually contribute to the average in reference 1:

In the customizable field write 3
 (the column number that marks with “1” the particles actually used
  in an average)
The text “averaged” should appear automatically.



Depicted properties / Tag selection

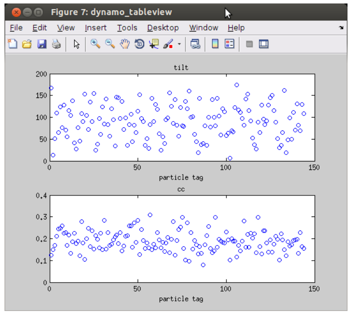
		y	y	y	y	y		x	select subset: clear refresh	selected tags
tag		○	○	○	○	○	view	○	1 < 144	144
correlation		○	●	○	○	○	view	○	0.07 < 0.32	144
tilt	?	●	○	○	○	○	view	○	6.22 < 173.78	144
averaged	3	○	○	○	○	○	view	○	1	46
dy	5	○	○	○	○	○	view	○	-4.03 < 2.42	144
dz	6	○	○	○	○	○	view	○	-4.39 < 7.66	144

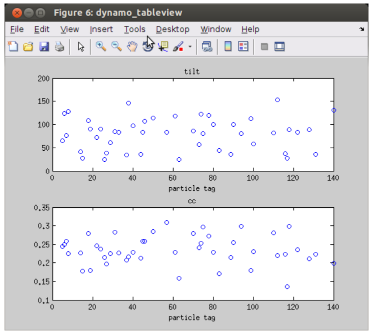
intersection of 46
clear quickload 144
clear final selection 46
average quicksave

Plot options

[✓ depict only Selection] highlight Selection
size - 20 +
[✓ only main table]
depicted tables: 1
colorbar
legend

Extended options for linear plot





Switch on so that only particles in the selection will get depicted!

The depiction area should represent now only those particles contributing to the average

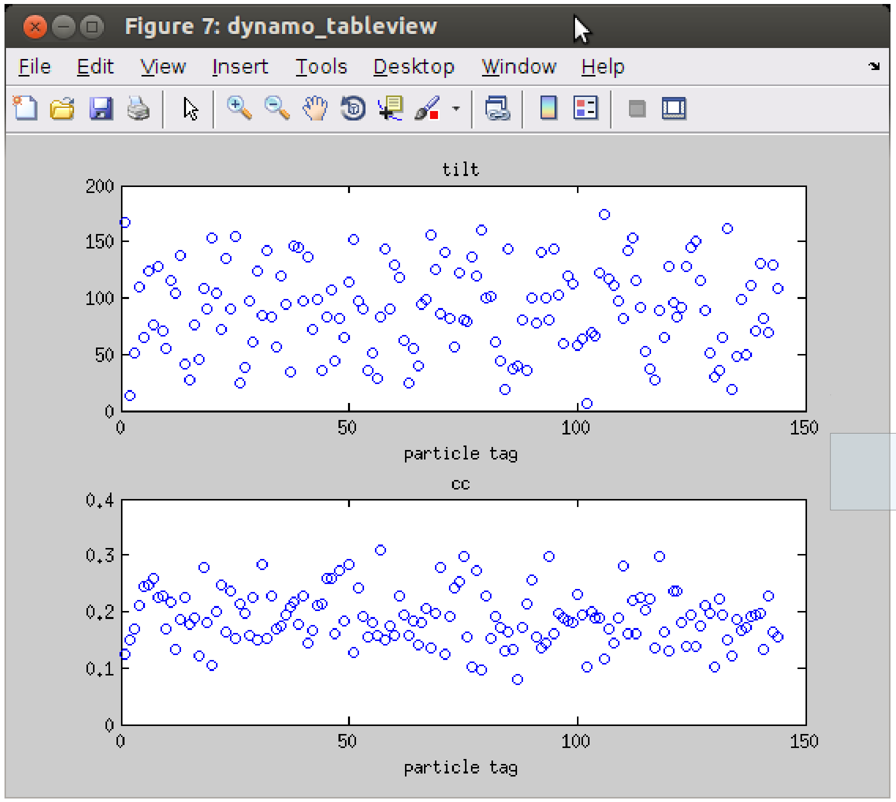


Figure 7: dynamo_tableview
File Edit View Insert Tools Desktop Window Help
tilt
particle tag
cc
particle tag

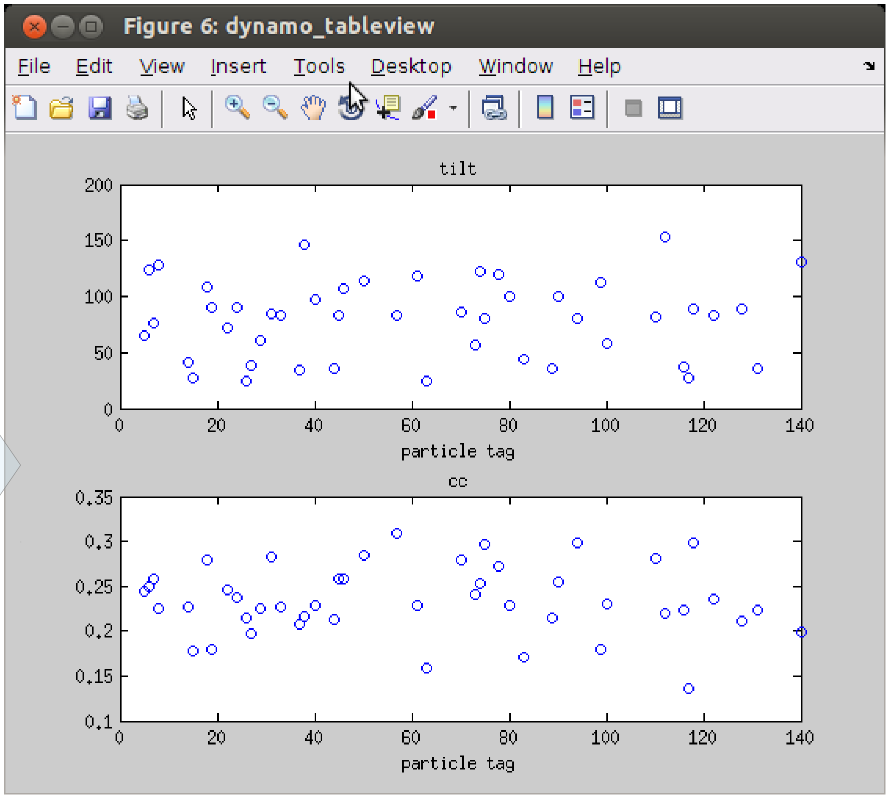
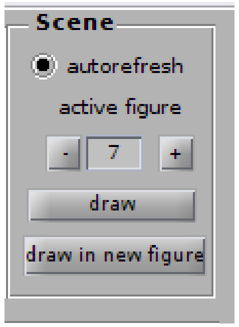
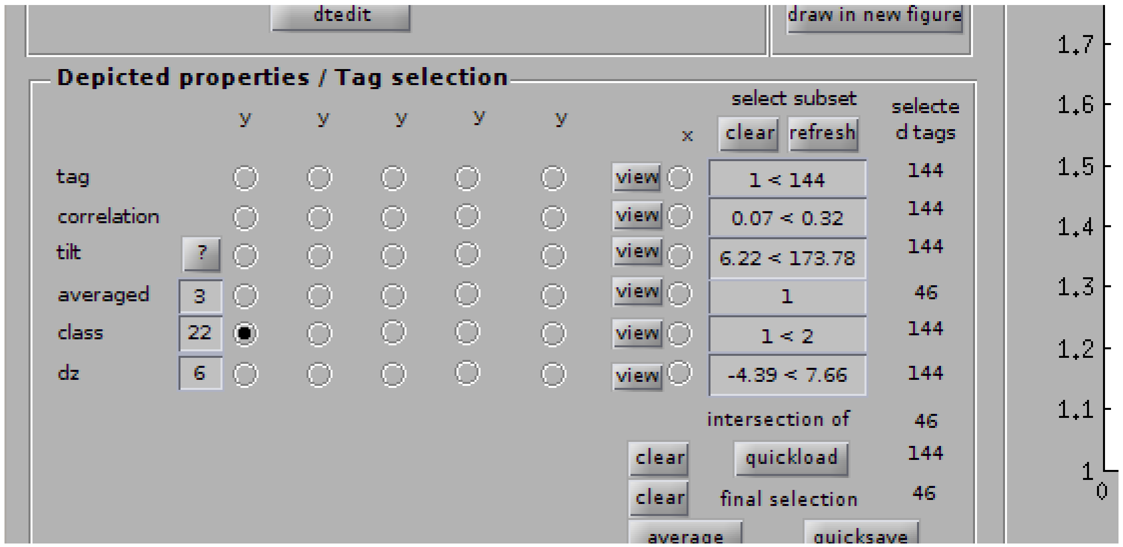


Figure 6: dynamo_tableview
File Edit View Insert Tools Desktop Window Help
tilt
particle tag
cc
particle tag



Tip:
Use these controls in order to produce different figures.
(figure 0 is the tableview window itself)

but we are still seeing the “tilt” and the “cc” (correlation) of each particle



Depicted properties / Tag selection

		y	y	y	y	y	x	select subset	selected tags
tag		○	○	○	○	○	○	1 < 144	144
correlation		○	○	○	○	○	○	0.07 < 0.32	144
tilt	?	○	○	○	○	○	○	6.22 < 173.78	144
averaged	3	○	○	○	○	○	○	1	46
class	22	●	○	○	○	○	○	1 < 2	144
dz	6	○	○	○	○	○	○	-4.39 < 7.66	144

intersection of 46

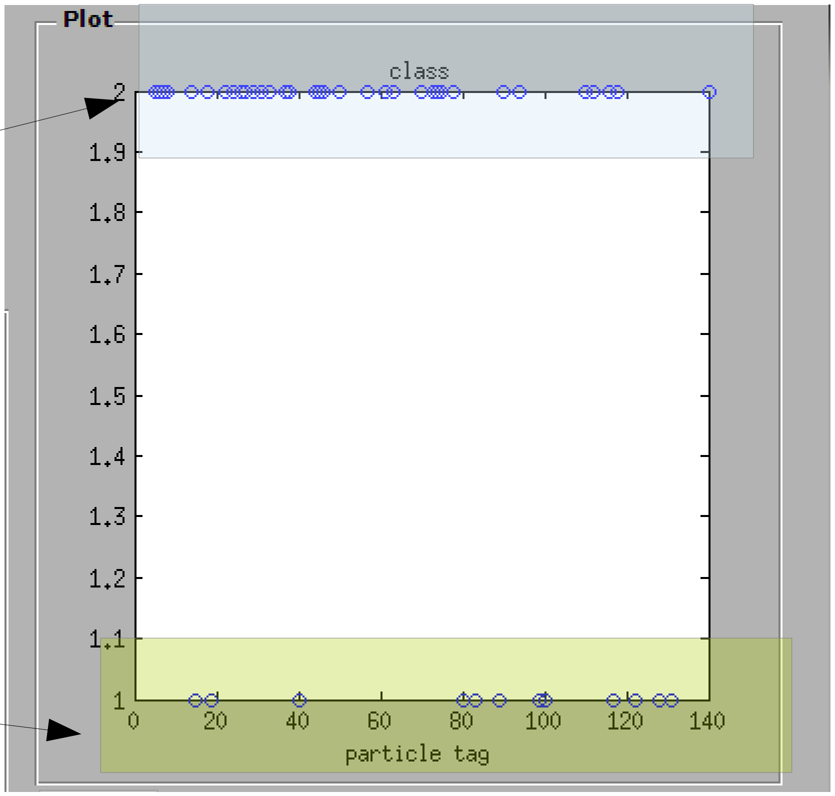
quickload 144

final selection 46

You should see something like this:

most particles in this reference channel contain particles In original class 2 (I.e. insertions)

although there are also a few particles that had been missclassified (they belonged to the original class 1, without insertions)


Plot
class
particle tag
2
1.9
1.8
1.7
1.6
1.5
1.4
1.3
1.2
1.1
1
0
20
40
60
80
100
120
140