Managing projects from the command line: basics

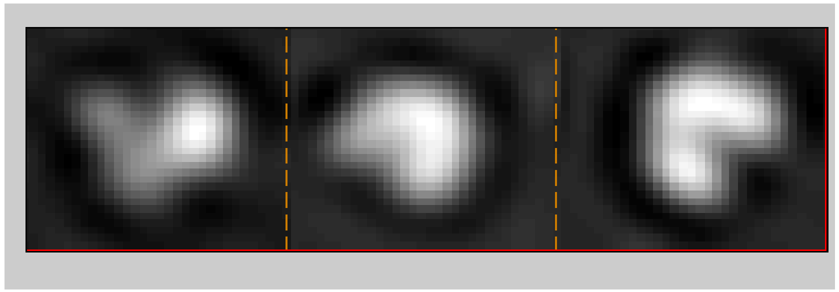
Let us create a small tutorial data set with an accompanying project.

We will use as template a ribosome in a 32x32x32 cube

```
>>dtutorial trib -p prib -template ribosome32.em -tight on;
```

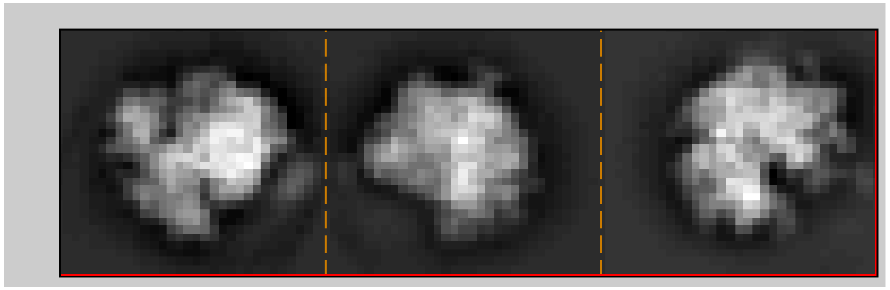
Notice the template assigned to the project:

```
>> ddb prib:template -j;
```



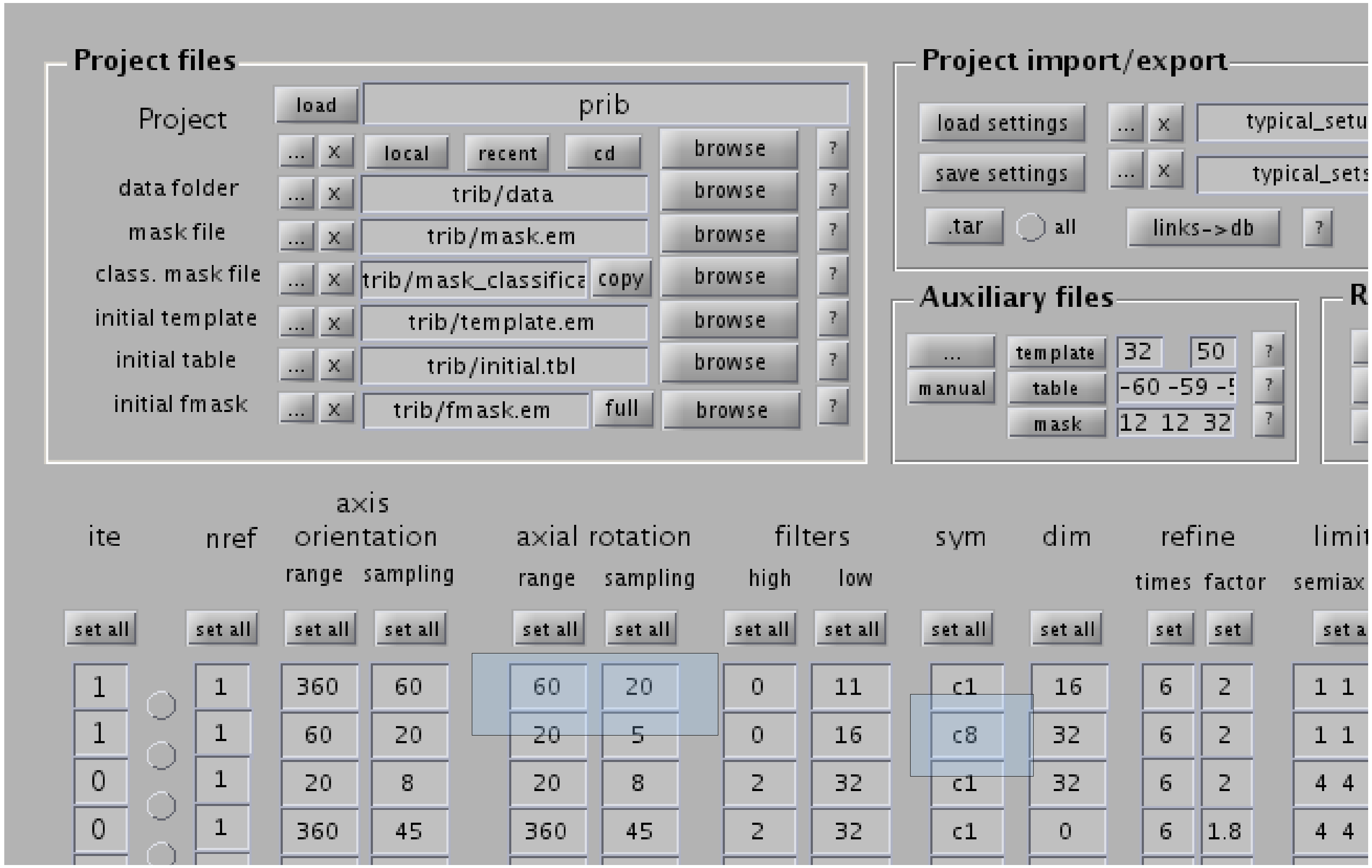
which is a lowpass of the template in the tutorial:

```
>> dslices trib/original_template.em x|y|z  -j *
```



open the generated project with `dynamo_project_manager` (or its shortfom `dpm`)

```
>> dpm prib
```



Project files

Project	load	prib
data folder		trib/data
mask file		trib/mask.em
class. mask file		trib/mask_classifica
initial template		trib/template.em
initial table		trib/initial.tbl
initial fmask		trib/fmask.em

Project import/export

load settings	typical_setu
save settings	typical_sets

Auxiliary files

template	32 50
table	-60 -59 -5
mask	12 12 32

ite	nref	axis orientation range	axis orientation sampling	axial rotation range	axial rotation sampling	filters high	filters low	sym	dim	refine times	refine factor	limit semiax
1	1	360	60	60	20	0	11	c1	16	6	2	1 1
1	1	60	20	20	5	0	16	c8	32	6	2	1 1
0	1	20	8	20	8	2	32	c1	32	6	2	4 4
0	1	360	45	360	45	2	32	c1	0	6	1.8	4 4

The command `dtutorial` creates numerical parameters that are more suitable for the default thermosome template.

Besides, in this tutorial we want to produce more iterations than just two, each in one round.

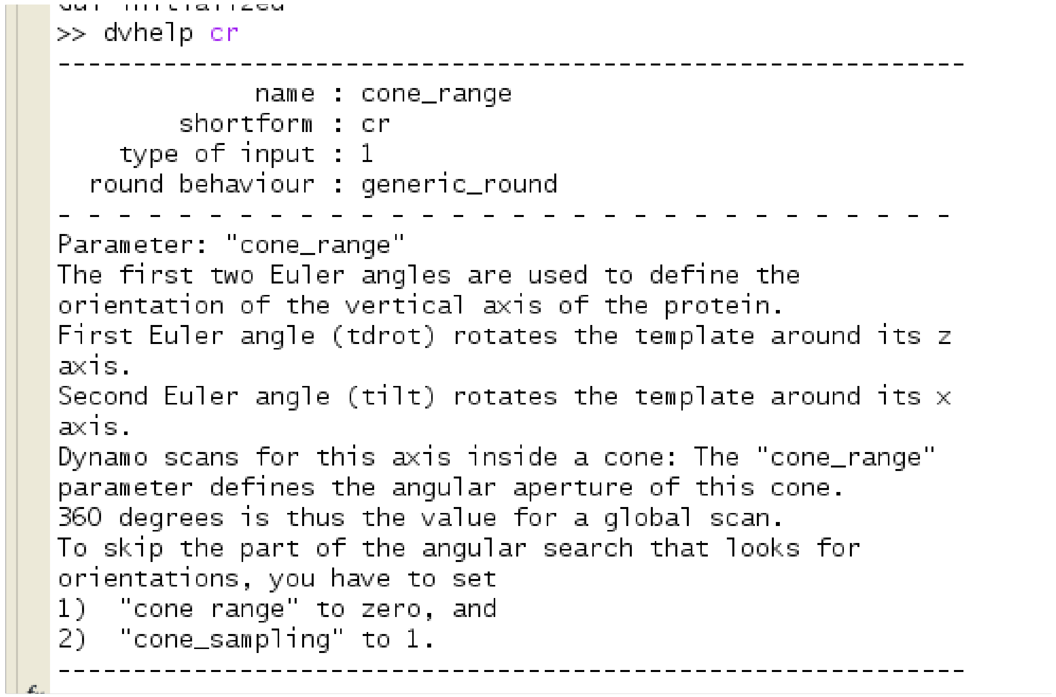
Obviously you can make the changes in the GUI (that's what it is for!).

But notice the alternate way of passing parameters into a project by the command line

```
>> dvput prib disk -inround 1 ite 2 -cr 360 -cs 60 -ir 360 -is 60;
>> dvput prib disk -inround 2 ite 4 -cr 60 -cs 20 -ir 60 -is 20 -rf 3;
>> dvput prib disk -inround 3 ite 4 -cr 20 -cs 8 -ir 20 -is 8 -rf 6;
```

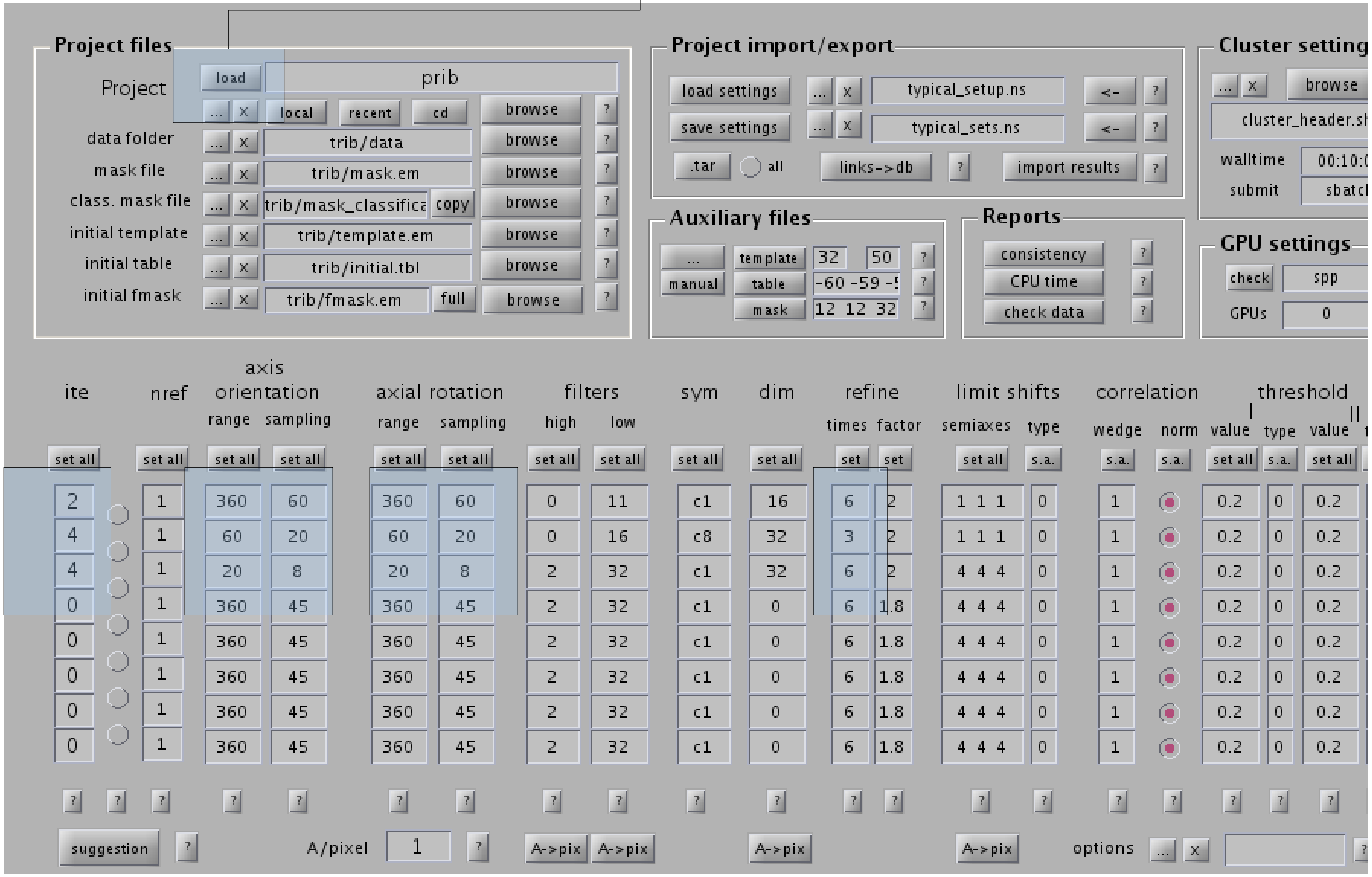
* The syntax of `dvput` is explained in its documentation (`ddoc dvput`)

* Each parameter is explained with the `dvhelp` command:
  - without arguments lists all project parameters
  - with a parmeter name as argument, it will look for specific help on that parameter:



```
>> dvhelp cr
-------------------------------------------------------------
             name : cone_range
        shortform : cr
    type of input : 1
  round behaviour : generic_round
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Parameter: "cone_range"
The first two Euler angles are used to define the
orientation of the vertical axis of the protein.
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 the angular aperture of this cone.
360 degrees is thus the value for a global scan.
To skip the part of the angular search that looks for
orientations, you have to set
1)  "cone range" to zero, and
2)  "cone_sampling" to 1.
-------------------------------------------------------------
```

Now, if you load the project against from the GUI `dynamo_project_manager`:

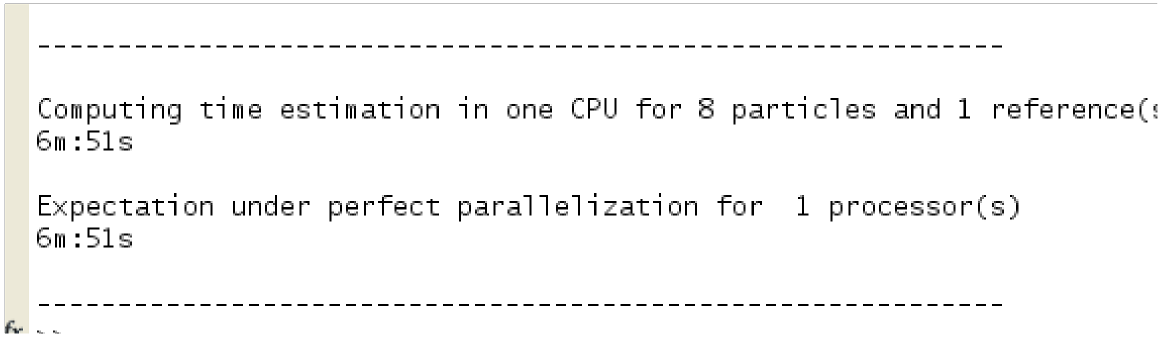


Project	prib
data folder	trib/data
mask file	trib/mask.em
class. mask file	trib/mask_classifica
initial template	trib/template.em
initial table	trib/initial.tbl
initial fmask	trib/fmask.em
load settings	typical_setup.ns
save settings	typical_sets.ns
template	32 50
table	-60 -59 -5
mask	12 12 32
cluster header	cluster_header.sh
walltime	00:10:0
submit	sbatc
GPU settings	spp
GPUs	0
A/pixel	1

ite	nref	axis orientation range	axis orientation sampling	axial rotation range	axial rotation sampling	filters high	filters low	sym	dim	refine times	refine factor	limit shifts semiaxes	limit shifts type	correlation wedge	correlation norm	threshold value	threshold type	threshold value
2	1	360	60	360	60	0	11	c1	16	6	2	1 1 1	0	1	●	0.2	0	0.2
4	1	60	20	60	20	0	16	c8	32	3	2	1 1 1	0	1	●	0.2	0	0.2
4	1	20	8	20	8	2	32	c1	32	6	2	4 4 4	0	1	●	0.2	0	0.2
0	1	360	45	360	45	2	32	c1	0	6	1.8	4 4 4	0	1	●	0.2	0	0.2
0	1	360	45	360	45	2	32	c1	0	6	1.8	4 4 4	0	1	●	0.2	0	0.2
0	1	360	45	360	45	2	32	c1	0	6	1.8	4 4 4	0	1	●	0.2	0	0.2
0	1	360	45	360	45	2	32	c1	0	6	1.8	4 4 4	0	1	●	0.2	0	0.2
0	1	360	45	360	45	2	32	c1	0	6	1.8	4 4 4	0	1	●	0.2	0	0.2

... the GUI updates, as the numerical scheme for the iterations will have changed

We can make sure that the project does not need a lot of computation time:
>>dvtiming prib



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

Computing time estimation in one CPU for 8 particles and 1 reference(s
6m:51s

Expectation under perfect parallelization for  1 processor(s)
6m:51s

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

... so we can unfold and run the project in the usual way:

```
>> dvunfold prib
>> prib
```

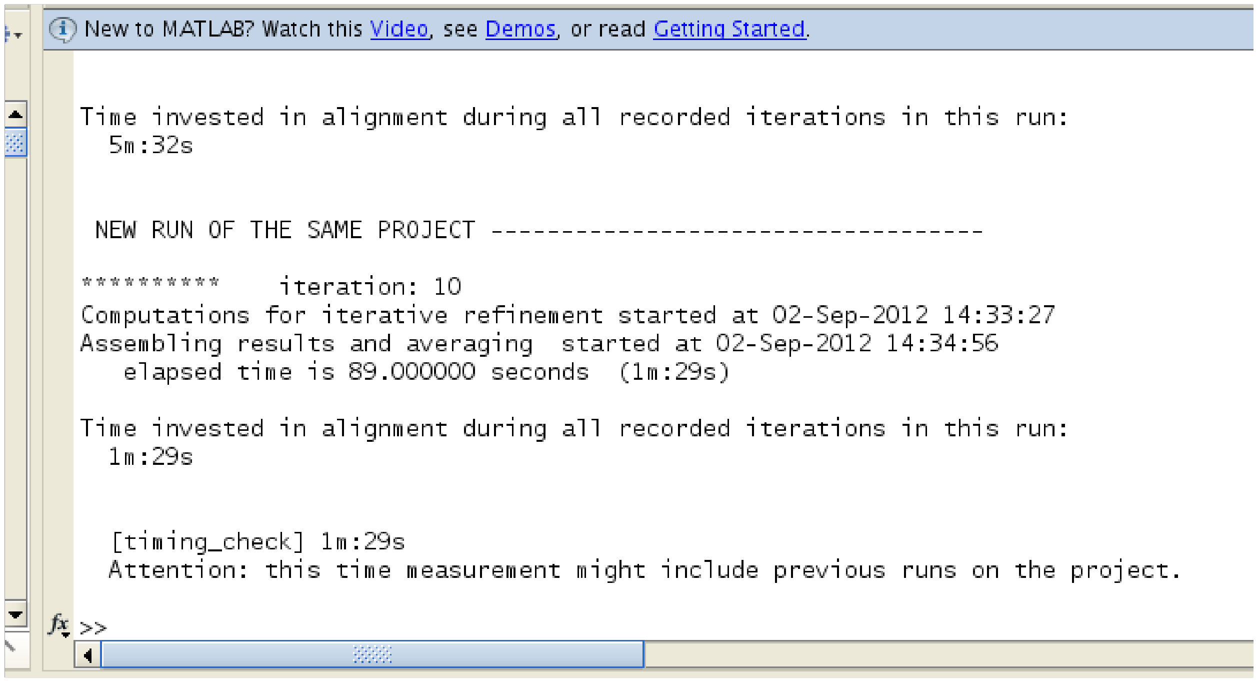
when running Dynamo from a shell you need to execute the produced execution script (with extension .exe,.bat) or submit it to a queuing system(with extension .sh)

... and after completion we can check the actually used computation time:

```
>> dvtiming_check prib
```

which (in this case) turns out to be quite accurate.

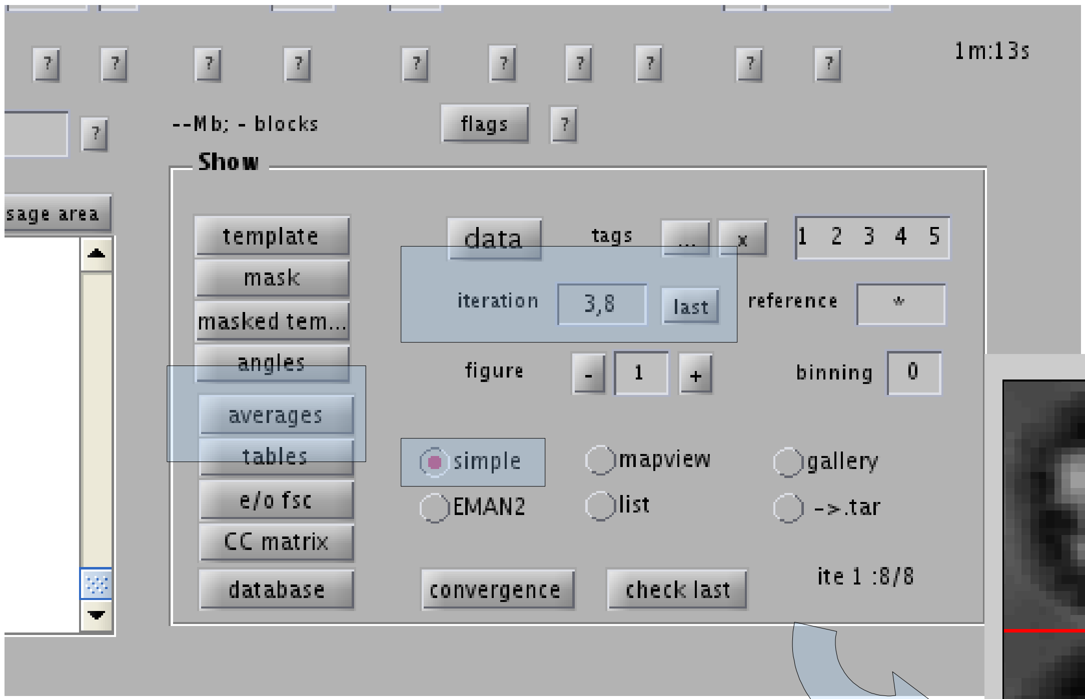
With multicore and MPI runs things won't be so accurate!



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

```
Time invested in alignment during all recorded iterations in this run:
  5m:32s


 NEW RUN OF THE SAME PROJECT -----------------------------------

**********    iteration: 10
Computations for iterative refinement started at 02-Sep-2012 14:33:27
Assembling results and averaging  started at 02-Sep-2012 14:34:56
   elapsed time is 89.000000 seconds  (1m:29s)

Time invested in alignment during all recorded iterations in this run:
  1m:29s


  [timing_check] 1m:29s
  Attention: this time measurement might include previous runs on the project.
>>
```

You probably know how to retrieve the results from the database using the GUI:



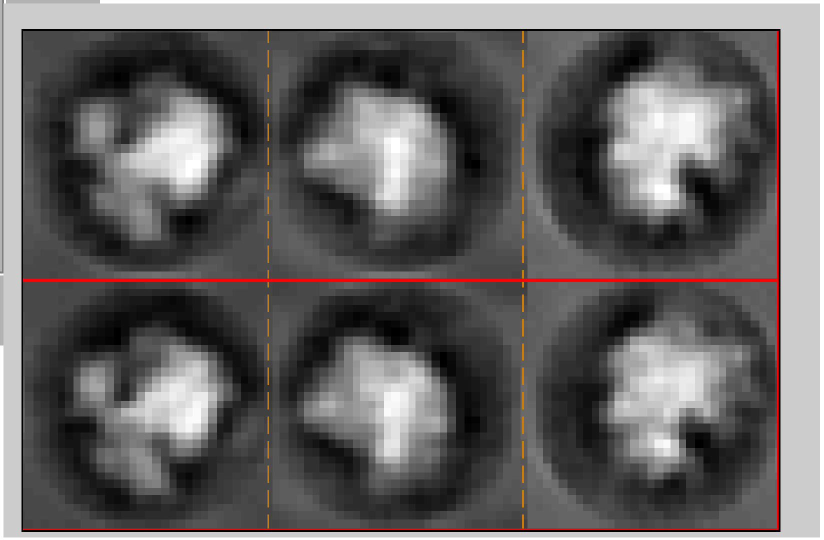
iteration 3,8 last

reference *

figure 1

binning 0

[✓ simple]



Pressing here for this parameter combination would create a simple depiction: the projections along x,y and of the averages attained in iterations 3 and 8 but the files can also be located, accessed and operated upon with the database:

1- pick the project

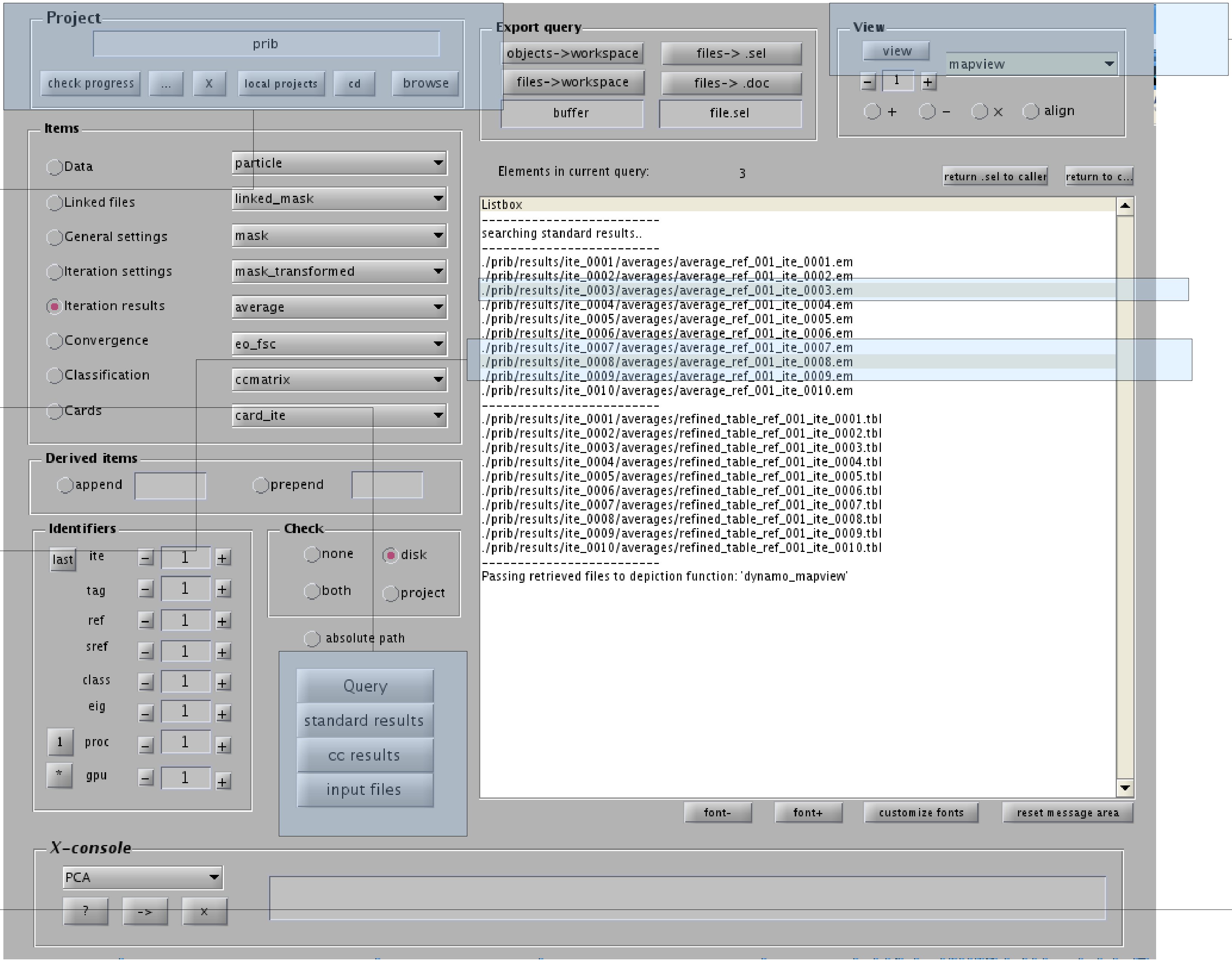
2- query for standard results

3- select items of interest

4 operate on them:

4a with the Dynamo linker, or

4b with local tools



Project: prib

Items: [✓ Iteration results] average

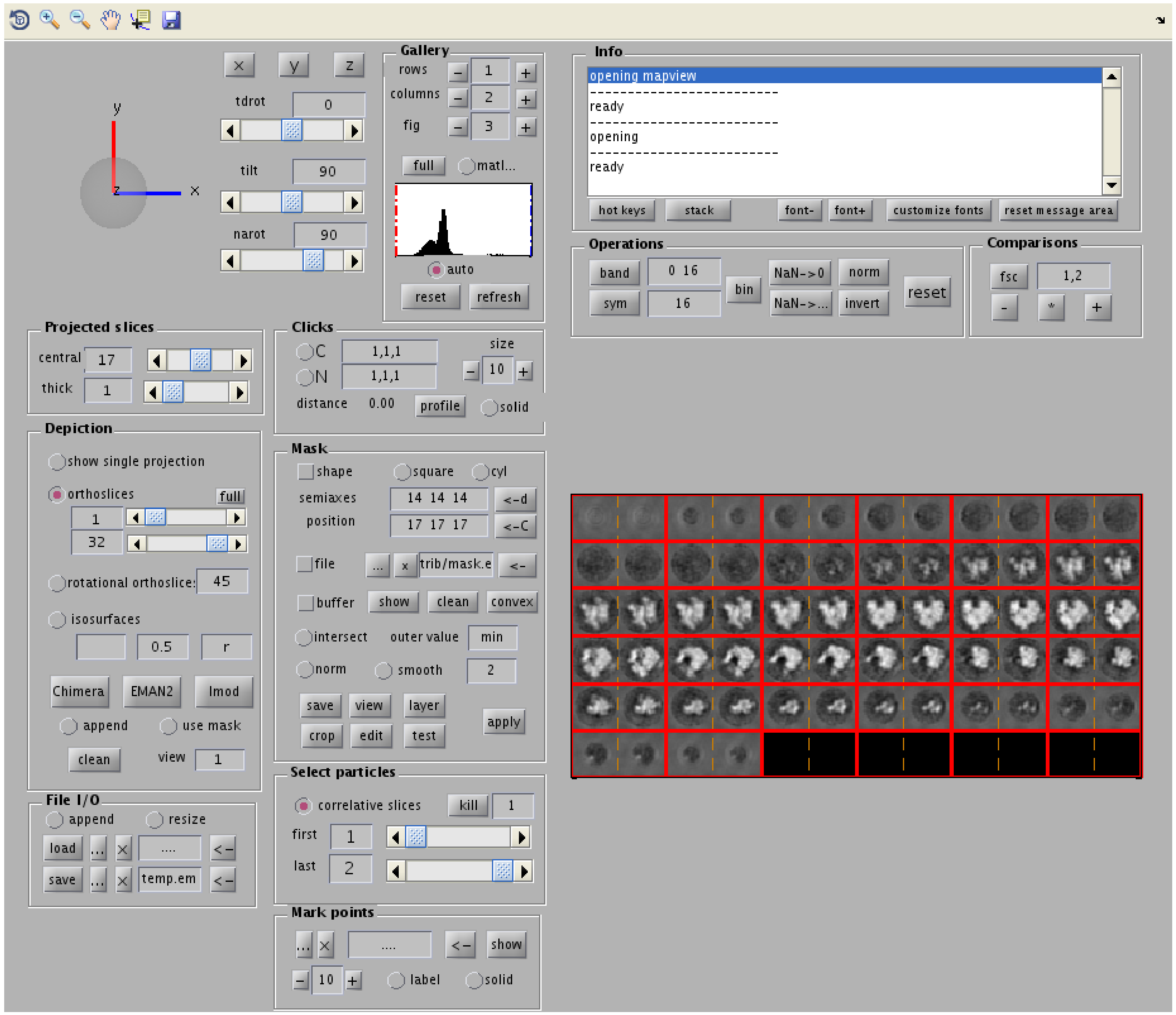
Identifiers: ite 1, tag 1, ref 1, sref 1, class 1, eig 1, proc 1, gpu 1

Check: none [✓ disk] both project

Elements in current query: 3

```
Listbox
--------------------------
searching standard results..
--------------------------
./prib/results/ite_0001/averages/average_ref_001_ite_0001.em
./prib/results/ite_0002/averages/average_ref_001_ite_0002.em
./prib/results/ite_0003/averages/average_ref_001_ite_0003.em
./prib/results/ite_0004/averages/average_ref_001_ite_0004.em
./prib/results/ite_0005/averages/average_ref_001_ite_0005.em
./prib/results/ite_0006/averages/average_ref_001_ite_0006.em
./prib/results/ite_0007/averages/average_ref_001_ite_0007.em
./prib/results/ite_0008/averages/average_ref_001_ite_0008.em
./prib/results/ite_0009/averages/average_ref_001_ite_0009.em
./prib/results/ite_0010/averages/average_ref_001_ite_0010.em
--------------------------
./prib/results/ite_0001/averages/refined_table_ref_001_ite_0001.tbl
./prib/results/ite_0002/averages/refined_table_ref_001_ite_0002.tbl
./prib/results/ite_0003/averages/refined_table_ref_001_ite_0003.tbl
./prib/results/ite_0004/averages/refined_table_ref_001_ite_0004.tbl
./prib/results/ite_0005/averages/refined_table_ref_001_ite_0005.tbl
./prib/results/ite_0006/averages/refined_table_ref_001_ite_0006.tbl
./prib/results/ite_0007/averages/refined_table_ref_001_ite_0007.tbl
./prib/results/ite_0008/averages/refined_table_ref_001_ite_0008.tbl
./prib/results/ite_0009/averages/refined_table_ref_001_ite_0009.tbl
./prib/results/ite_0010/averages/refined_table_ref_001_ite_0010.tbl
--------------------------
Passing retrieved files to depiction function: 'dynamo_mapview'
```

... and also with the command line tool for database browsing ddb:
>> ddb prib:a:ite=[3,8] -m

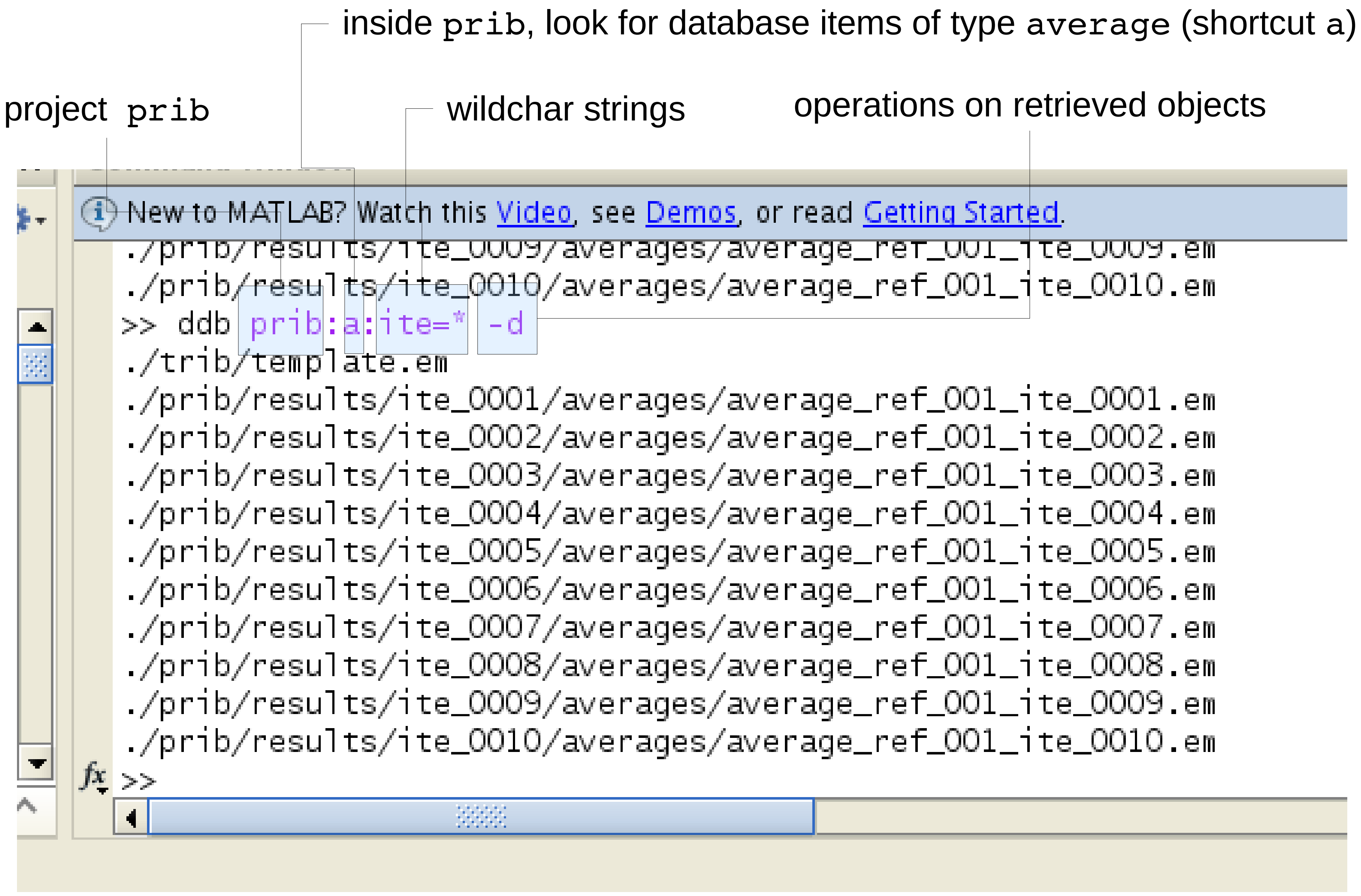


```
opening mapview
-------------------------
ready
-------------------------
opening
-------------------------
ready
```

tdrot 0
tilt 90
narot 90
rows 1
columns 2
fig 3
band 0 16
sym 16
fsc 1,2
central 17
thick 1
C 1,1,1
N 1,1,1
size 10
distance 0.00
[✓ orthoslices] 1 32
rotational orthoslice 45
isosurfaces 0.5
view 1
semiaxes 14 14 14
position 17 17 17
file trib/mask.e
outer value min
smooth 2
save temp.em
[✓ correlative slices] kill 1
first 1
last 2
Mark points 10

The ddb tool lets you access different elements in a project or set of projects:

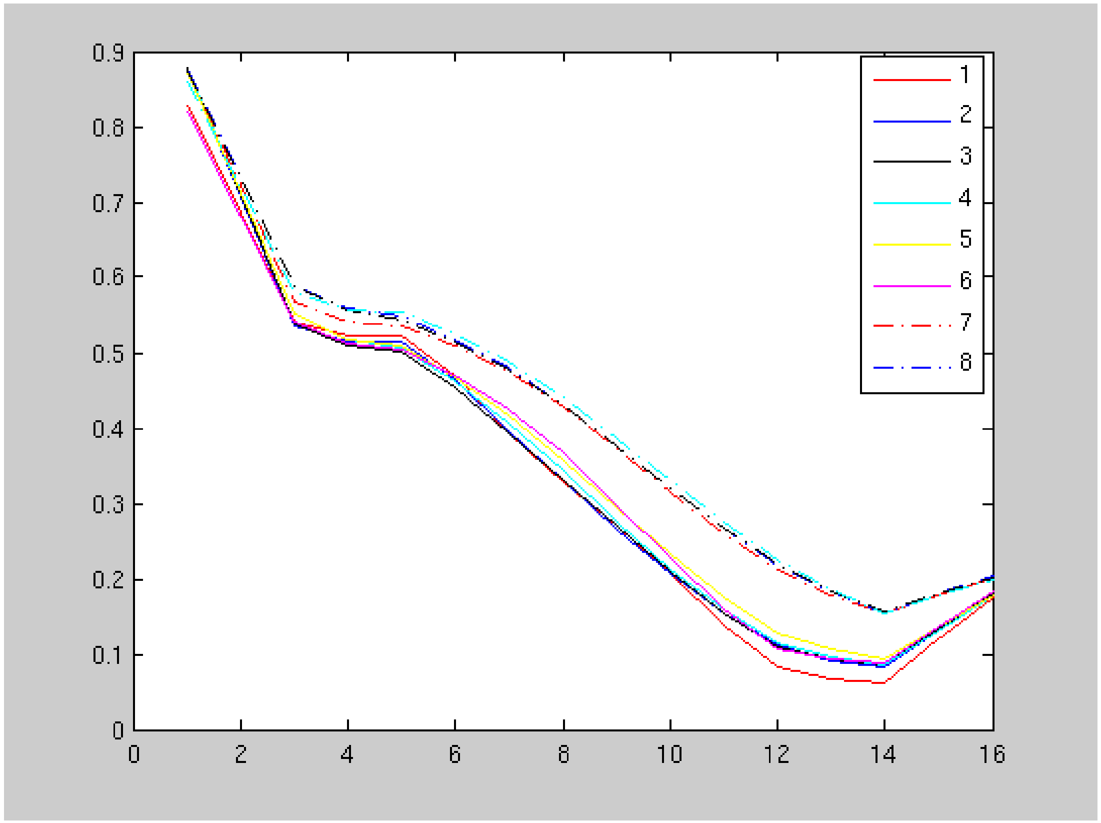
For instance:


inside `prib`, look for database items of type `average` (shortcut `a`)
project `prib`
wildchar strings
operations on retrieved objects
New to MATLAB? Watch this Video, see Demos, or read Getting Started.

```
./prib/results/ite_0009/averages/average_ref_001_ite_0009.em
./prib/results/ite_0010/averages/average_ref_001_ite_0010.em
>> ddb prib:a:ite=* -d
./trib/template.em
./prib/results/ite_0001/averages/average_ref_001_ite_0001.em
./prib/results/ite_0002/averages/average_ref_001_ite_0002.em
./prib/results/ite_0003/averages/average_ref_001_ite_0003.em
./prib/results/ite_0004/averages/average_ref_001_ite_0004.em
./prib/results/ite_0005/averages/average_ref_001_ite_0005.em
./prib/results/ite_0006/averages/average_ref_001_ite_0006.em
./prib/results/ite_0007/averages/average_ref_001_ite_0007.em
./prib/results/ite_0008/averages/average_ref_001_ite_0008.em
./prib/results/ite_0009/averages/average_ref_001_ite_0009.em
./prib/results/ite_0010/averages/average_ref_001_ite_0010.em
>>
```

The syntax is general for anything
that has to do with a project:

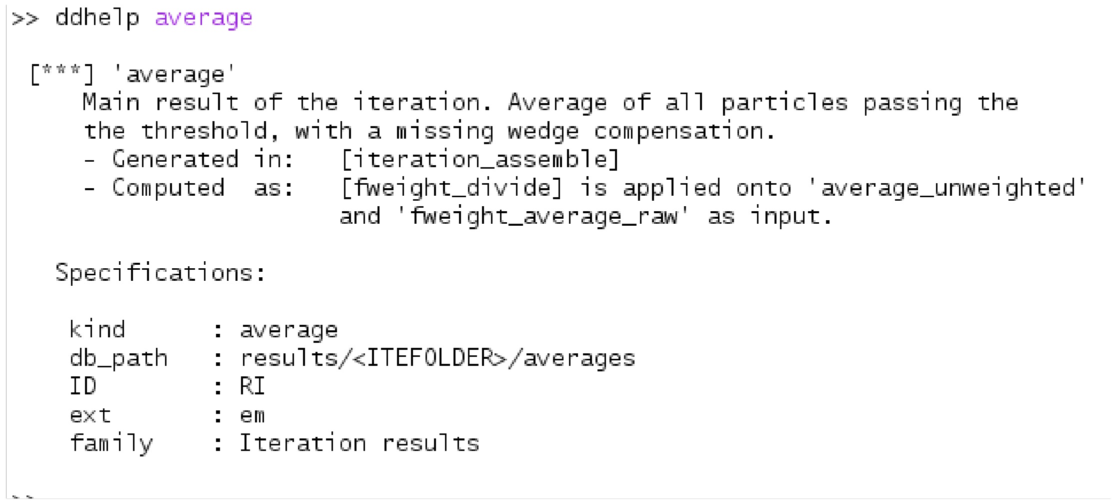
```
>>  prib:eo_fsc:ite=[1:10] -p
```



The syntax and list of “database items” that can be retrieved with `ddb` is in its documentation (`ddoc ddb`)

Closer information on the database items can be invoked with `dbhelp`

`and the tutorial on plugins`



```
>> ddhelp average

 [***] 'average'
     Main result of the iteration. Average of all particles passing the
     the threshold, with a missing wedge compensation.
     - Generated in:   [iteration_assemble]
     - Computed  as:   [fweight_divide] is applied onto 'average_unweighted'
                       and 'fweight_average_raw' as input.

   Specifications:

    kind       : average
    db_path    : results/<ITEFOLDER>/averages
    ID         : RI
    ext        : em
    family     : Iteration results
```