codimensional PCA test

02/02/2011

in directory codimPCA create directory test:
mkdir test
cd test

Description of simulated models

	9453	
PRE	70% 6633	30% 70S + tRNA + EFG (2882) 30S_MDD_TS-PRE.pdb + 50S_MDD_TS-PRE_4Pawel.pdb + PPE-tRNA_MDD_TS-PRE.pdb + EFG_FUS_MDD_TS-PRE.pdb
		20% 70S + tRNA (1861) 30S_MDD_TS-PRE.pdb + 50S_MDD_TS-PRE_4Pawel.pdb + PPE-tRNA_MDD_TS-PRE.pdb
		20% 70S + EFG (1909) 30S_MDD_TS-PRE.pdb + 50S_MDD_TS-PRE_4Pawel.pdb + EFG_FUS_MDD_TS-PRE.pdb
POST	30% 2820	15% 70S + tRNAs (1455) 2WRI_alignvia50S.pdb + 2WRJ_align_4Pawel.pdb
		15% 70S (1346) 2WRI_30S.hdf + 2WRJ_align_4Pawel.pdb

Prepare simulated EM electron density maps by converting five pdb files

```
sxpdb2em.py ../data/grp0.pdb grp0.hdf  --apix=4.3 --box=75
sxpdb2em.py ../data/grp1.pdb grp1.hdf  --apix=4.3 --box=75
sxpdb2em.py ../data/grp2.pdb grp2.hdf  --apix=4.3 --box=75
sxpdb2em.py ../data/grp3.pdb grp3.hdf  --apix=4.3 --box=75
sxpdb2em.py ../data/grp4.pdb grp4.hdf  --apix=4.3 --box=75
```

Delete pixel size from the header. *chimera reads it and adjusts the scale!*

```
sxheader.py grp0.hdf --delete --params="apix_x apix_y apix_z  pixel_size"
sxheader.py grp1.hdf --delete --params="apix_x apix_y apix_z  pixel_size"
sxheader.py grp2.hdf --delete --params="apix_x apix_y apix_z  pixel_size"
sxheader.py grp3.hdf --delete --params="apix_x apix_y apix_z  pixel_size"
sxheader.py grp4.hdf --delete --params="apix_x apix_y apix_z  pixel_size"
```

To visualize test structures:
in chimera:
chimera grp*hdf &
in eman2
e2display.py grp*hdf &

PCA performed directly in 3D on multiple noise-corrupted copies of five test structures

```
mkdir  EVOL
python  ../data/genvl5.py
```

Use python installed with eman2

In EVOL
rvls.hdf - stack with 100 unfiltered test structures
frvls.hdf - stack with 100 filtered test structures (to half Nyquist)
farvls.hdf - average of filtered structures
fvrvls.hdf - variance of filtered structures

PCA and check eigenvalues

```
sxpca.py EVOL/frvls.hdf EVOL/feigs.hdf --subavg=EVOL/farvls.hdf --rad=36 --nvec=9
sxheader.py EVOL/feigs.hdf --export=EVOL/feigval.txt --params=eigval
cat  EVOL/feigval.txt
sxspliteigen.py EVOL/feigs.hdf EVOL/EIG
```

“Interpretation” of PCA results

1. Load from directory test/EVOL to chimera the average (farvls.hdf) and variance (fvrvls.hdf).
   What the variance tells us?

2. Load eigenvolumes individually:
   test/EIG000?_neg.hdf
   test/EIG000?_pos.hdf
   What is their meaning?

3. Use “Morph Map” in Volume Viewer/Tools to animate eigenvolumes (In Options, check all options).
   “Dynamics” - how to distinguish between continuous movements and distinct conformers?

Compute factorial coordinates, do K-means clustering and initial volumes.

Compute factorial coordinates

```
sxfactcoords.py  EVOL/frvls.hdf EVOL/farvls.hdf  EVOL/feigs.hdf  EVOL/ffactcoords
    --rad=36   --neigvol=8 &
```

Prepare a “mask” files that will select first two factorial coordinates.

```
sparx
    > a = model_blank(9)
    > a[0] = 1
    > a[1] = 1
    > print_image(a)
    > a.write_image("m2_9.hdf")
    > Quit
```

Determine number of groups:

```
sxk_means_groups.py  EVOL/ffactcoords.txt outgf  m2_9.hdf --K1=2 --K2=6 --trials=20 --maxit=1000&
```

Run K-means for five groups:

```
sxk_means.py EVOL/ffactcoords.txt woutgf9 m2_9.hdf --K=5 --trials=500 --maxit=1000 --crit='all'
    --init_method='d2w' &
```

Conclusions
PCA done directly on structures

1. *PCA on 3D structures is simple and straightforward*. Note in the test structures were aligned, noise was low, and there was no missing information - PCA on tomographic structures much more difficult and not as successful.

2. *Variance is an indication of variability* but it is difficult to interpret in functional terms.

3. *Eigenvolumes provide wealth of information*, they decouple variance into independent components. It is often possible to interpret them in functional terms. However, one has to be careful as the analysis is ultimately based on correlations between regions of the structure and presence of correlation is not necessarily prove of causal effects.

4. *Eigenvolumes make it possible to compute factorial coordinates*. These offer huge reduction of dimensionality and thus make it possible to apply clustering techniques (K-means) and quickly obtain both the number of plausible conformers and assignments of individual structures to classes. Note (i) it is far from clear how many eigenvolumes one should use and the result will depend on the choice, (ii) we might have as well applied K-means directly to 3D structures.

5. *In practical application PCA+clustering is not as simple as it would appear from this example! Many steps are subjective and validation techniques are lacking.*

codimensional PCA test

02/02/2011

in directory codimPCA create directory test:
mkdir test
cd test

codimensional PCA

We will create projections using unevenly distributed projection directions stored in ../data/a9453.txt
In directory test:

```
mkdir  CPRJ
python  ../data/genprefg.py
```

Use python installed with eman2

We have 9,453 projections stored in bdb database (subdirectory EMAN2DB). The actual number of projections generate for each structure might be different from that in the table as the process is random.
To get actual number of projections per group, type:

```
grep 0 CPRJ/assign.txt | wc
grep 1 CPRJ/assign.txt | wc
grep 2 CPRJ/assign.txt | wc
grep 3 CPRJ/assign.txt | wc
grep 4 CPRJ/assign.txt | wc
```

codimensional PCA - resampling

Write all projection images to a disk buffer (in a Fourier format)

```
sxgenbuf.py bdb:proj  buffer --npad=2
```

Determine angular step 0<***delta***<90 and ratio of projection images to leave out 0<***d***<1 experimentally. One would want to have possibly large number of strata (i.e., small delta), but large number of projections per stratum and large number of projections in resampled structure. These goals are contradictory, so a compromise has to be found. The hint is to keep in mind that the larger the delta, the lower the resolution of the results.

```
rm -rf JACK;  sxresample.py bdb:proj JACK buffer --delta=10.0 --d=0.2 --nvol=1 --nbufvol=1
```

Using ***delta*** and ***d*** determined in previous step, perform the actual resampling. This is a time-consuming step, so normally is ran using MPI parallel mode, in which case subdirectory JACK will contain as many stack files with resampled structures as there are CPUs used.
For the sake of efficiency, the program works simultaneously on multiple volumes (***nbufvol***). Given memory **M** Gb available to a CPU and size of structure **n** voxels, approximately **nbufvol** = **M**/(6*(**npad***n**)**3)/1.e6.
seedbase determines the random sequence used for resampling (makes it possible to repeat calculations and obtain the same result). In some cases one may want to compute additional resampled volumes, in which case seedbase should be changed.
This command will also print ***multiplicative factor for the variance*** (see the main presentation for the equation for the coefficient that ties expected value of the HYPERSTRATIS variance S^2 is related to the distribution variance σ^2). The variance computed from resampled volumes has to be multiplied by this number if proper scaling is desired.

```
sxresample.py bdb:proj JACK buffer --delta=10.0 --d=0.2 --nvol=11200 --nbufvol=100
    --seedbase=17
```

```
mpirun -np 4 sxresample.py bdb:proj JACK buffer --delta=10.0 --d=0.2 --nvol=11200 --nbufvol=25
    --seedbase=17  --MPI
```

Output: in subdirectory JACK stacks bsvol000*.hdf that contain resampled structures.

codimensional PCA - eigenanalysis

for EM data (not test), use next page instead.

Compute adaptive mask using low-pass filtered resampled volume.
Important: the overall results strongly depend on the lowpass filtration parameters in the command below and all subsequent commands. If the test is repeated and data generated again, it is likely that the filter would have to be adjusted. cutoff_abs=0.27:fall_off=0.2 works often.
The filter parameters have to have the same exact values in all commands that follow (fl is cutoff_abs, fall_off is aa).

```
e2proc3d.py JACK/bsvol0000.hdf junk.hdf --process=filter.lowpass.tanh:cutoff_abs=0.3:fall_off=0.1
--last=0
sxbmask.py  junk.hdf  CPRJ/bmsk.hdf  CPRJ/smsk.hdf
rm junk.hdf
```

Compute average structure and its variance, fl, aa - cut-off and width of tangent low-pass filter and eigenvolumes (MPI. However, this program is parallelized over stacks that contain resampled volumes, so use at most the number of CPUs equal to the number of available stacks.).

```
sxvar.py  `find JACK -name "b*.hdf" ` rack02 --radccc=36 --pca --pcamask=CPRJ/bmsk.hdf  --
pcanvec=6  --fl=0.30 --aa=0.1
```

codimensional PCA - eigenanalysis

optional, recommended for EM data

Compute average structure and its variance, fl, aa - cut-off and width of tangent low-pass filter (use MPI. However, this program is parallelized over stacks that contain resampled volumes, so use at most the number of CPUs equal to the number of available stacks.)

```
sxvar.py  `find JACK -name "b*.hdf" ` rack01  --radccc=34 --repair=None --fl=0.30 --aa=0.1
```

Compute adaptive mask and repair volume to equalize the variance.

```
cd rack01
sxbmask.py  avg.hdf  bmsk.hdf  smsk.hdf  --repair=repair.hdf  --variance=var.hdf
cd ..
```

Compute ‘repaired’ variance and eigenvolumes (MPI).

```
sxvar.py  `find JACK -name "b*.hdf" ` rack02 --repair=rack01/repair.hdf  --radccc=36 --pca
--pcamask=rack01/bmsk.hdf  --pcanvec=6  --fl=0.30 --aa=0.1
```

Optionally in order to determine number of eigenvolumes perform PCA on correlation matrix and compare the resulting eigenvolumes with the previous set. Check using ccc how many reasonably agree (should be abs(ccc)~1).

```
cd rack01
sparx
> square_root(get_im(“var.hdf”)).write_image(“stdv.hdf”,0)
Quit
cd ..
sxvar.py  `find JACK -name "b*.hdf" ` rack03 --repair=rack01/stdv.hdf  --radccc=36 --MPI --
pca --pcamask=rack01/bmsk.hdf  --pcanvec=9  --fl=0.30 --aa=0.1
```

codimensional PCA - eigenanalysis interpretation of results

Compute average structure and its variance, fl, aa - cut-off and width of tangent low-pass filter (use MPI!)

```
sxheader.py rack02/eigvol.hdf --export=CPRJ/eigval.txt --params=eigval
cat  CPRJ/eigval.txt
sxspliteigen.py rack02/eigvol.hdf  CPRJ/EIG
```

1. Compare variance map obtained directly from the structures to that obtained from resampling (in rack02). What is the difference?

2. Examine distribution of eigenvalues - first two are clearly dominating.

3. Compare eigenvolumes with those obtained from the direct analysis of 3D structures - first two agree very well.

4. Load average volume and first two eigenvolumes two chimera and use option Morph Map to infer their meaning.

codimensional PCA - clustering

Compute factorial coordinates (use MPI!)

```
sxfactcoords.py bdb:proj rack02/avg.hdf rack02/eigvol.hdf fact02  --rad=32 --fl=0.3 --aa=0.1
--neigvol=3
```

Determine plausible number of groups and do the clustering (MPI). (Prepare mask file that will select first two factorial coordinates).

```
sparx
    > a = model_blank(4)
    > a[0] = 1
    > a[1] = 1
    > print_image(a)
    > a.write_image("m2_4.hdf")
    > Quit
```

```
sxk_means_groups.py fact02.txt moutgf m2_4.hdf --K1=2 --K2=10 --trials=10 --maxit=1000
```

```
sxk_means.py fact02.txt g05 m2_4.hdf  --K=5 --trials=10 --maxit=1000 --rand_seed=10
    --crit='all' --init_method='rnd'
```

Use programs fassi.py and compgrp.py to check how well clustering performed.

Compute structures for all clusters (MPI).

```
sxrecons3d_n.py bdb:proj  CPRJ/vgrp1.hdf  --npad=2  --list=g05/kmeans_grp_001.txt &
sxrecons3d_n.py bdb:proj  CPRJ/vgrp2.hdf  --npad=2  --list=g05/kmeans_grp_002.txt &
sxrecons3d_n.py bdb:proj  CPRJ/vgrp3.hdf  --npad=2  --list=g05/kmeans_grp_003.txt &
sxrecons3d_n.py bdb:proj  CPRJ/vgrp4.hdf  --npad=2  --list=g05/kmeans_grp_004.txt &
sxrecons3d_n.py bdb:proj  CPRJ/vgrp5.hdf  --npad=2  --list=g05/kmeans_grp_005.txt &
```

codimensional PCA - multireference alignment

Copy all five initial structures into one stack.

```
sxcpy.py  CPRJ/vgrp*  CPRJ/initrev.hdf
e2proc3d.py CPRJ/initrev.hdf CPRJ/initref.hdf --
process=filter.lowpass.tanh::cutoff_abs=0.4:fall_off=0.1
```

Multireference alignment (use MPI!, only MPI version works). In fact, with the settings below we will only do reassignments (3D K-means) - no change of orientation parameters.

```
mpirun -np 16 sxmref_ali3d.py bdb:proj CPRJ/initref.hdf masi --ou=32 --maxit=20
      --function="[../data,spruceup,spruce_upmadj]"  --nassign=1 --nrefine=0
      --npad=2 --stoprnct=0.0 --MPI &
```

Recover assignments of projection images to groups.

```
sxheader.py bdb:proj --export=masi/assimasi.txt --params=group
```

***Exercise**: confirm that multireference alignment brought group assignment to 100% agreement with the original one.*

Conclusions
codimensional PCA

1. *PCA on resampled 3D structures is simple and straightforward, even though it is laborious*. It requires minimization of alignment errors and good normalization of the data.

2. *Variance is poorly resolved and distorted, cannot be used for detailed interpretation.*

3. *Eigenvolumes provide wealth of information*, they decouple variance into independent components and unlike the variance, are more detailed. It is often possible to interpret them in functional terms. However, one has to be careful as the analysis is ultimately based on correlations between regions of the structure and presence of correlation is not necessarily prove of causal effects.

4. *Eigenvolumes make it possible to compute factorial coordinates*. These offer huge reduction of dimensionality and thus make it possible to apply clustering techniques (K-means) and quickly obtain both the number of plausible conformers and assignments of individual structures to classes. Note it is far from clear how many eigenvolumes one should use and the result will depend on the choice. *The method is mainly limited by the resolution - the noise level in*

5. *In practical application codimensional PCA is not as simple as it would appear from this example! Many steps are subjective and validation techniques are lacking. However, it is the only currently available method that provides a very efficient way to analyze conformational variability in single particle EM data sets and the only that allows to rapidly and reliably to establish the number and structure of plausible conformers.*