GPseudoClust: example

This example demonstrates how to use GPseudoClust with subsampling to generate sets of posterior similarity matrices (PSMs) to be used for postprocessing (using the nonparametricSummaryPSM code or lmkk (see lmkk_summaryMatrixRepresentation)).

For this demonstration, we use a simulated data set, where we assume the "cells" where measured at three different capture times. The data set contains a total of 60 cells, 20 per capture time. For each of the subsampled MCMC chains, we subsample 10 cells per capture time.

```
addpath(genpath('../'))
captureTimes     = [ones([1,10]),repmat(2,[1,10]),repmat(3,[1,10])];
%capture times for the subsampled cells
fHandle          =  @GPseudoClust2;
```

The input file is a .csv file containing a matrix with genes along the rows and cells along the columns.

```
fileName         = 'simDataClust2.csv';
```

For 24 chains, we draw 1000 thinned (ie. 5000 unthinned) samples each from the posterior distribution. For each chain, we use a different identifier to be able to distinguish the output files.

```
uniqueIdentifiers    = 1:24;
nSamples             = 1000;
verbose          = false;
```

We set inputSeed to NaN, which uses a clock-based seeding with a chain-dependent offset, for each subsampled chain to ensure different seeds are used for the different chains.

```
inputSeed        = NaN;
```

Setting permuteData = true ensures that a random starting order is used for the cell ordering.

```
permuteData      = true;
b = 0.01;%recommended default
```

If required, cell size adjustment can be included as a preprocesing step (Anders, S. and Huber,W. (2010) . Differential expression analysis for sequence count data. Genome Biol, 11(10), R106–R106.)

```
adjustForCellSize = false;

parpool(3);%adjust to number of cores
```

```
Starting parallel pool (parpool) using the 'local' profile ...
connected to 3 workers.
```

```
tic
parfor j = 1:24
```

```
subS = [randsample(1:20,10),randsample(21:40,10),randsample(41:60,10)]
%random subsampling
feval(fHandle, fileName, uniqueIdentifiers(j), nSamples, verbose,...
    inputSeed,permuteData,captureTimes,b,adjustForCellSize,subS);
end
toc
```

```
Elapsed time is 601.803667 seconds.
```

Now we compute a PSM for each of the subsampled chains.

```
PSM = zeros(52,52,24);
for j=1:24
    PSM(:,:,j) = psm(dlmread(sprintf('simDataClust2_Results_Chain%d.csv',j),...
        ',',[499 1 999 52]));
end
save('PSMSim2NoDropout.mat','PSM');
```

Here, we compute a summary matrix representation using localised kernel k-means. For our alternative Bayesian nonparametric methods to compute summary PSMs (`DPM+PEAR', `PY+PEAR'), we use our separate R implementation (nonparametricSummaryPSM) available on github (https://github.com/magStra/nonparametricSummaryPSM, see vignette for R package concerning how to compute the summary PSMs).

```
addpath(genpath('~/SIMLR-SIMLR'));%add to path the folder where you
%downloaded SIMLR and its subfolders
addpath('~/lkkmeans')%add path to folder containing lmkk code
addpath(genpath('~/mosek'));%add path to folder containing the mosek software
load('PSMSim2NoDropout.mat');
clusterSolution = computeSummaryPSM_lmkk(PSM,2:10);
```

```
Warning: Note that we always assume there are more than one cluster.
running iteration 1...
running iteration 2...
running iteration 3...
running iteration 4...
running iteration 5...
running iteration 6...
running iteration 7...
running iteration 8...
running iteration 9...
running iteration 10...
     4
```

```
delete simDataClust2_*
```