baredSC

Release 1.1.2

Jean-Baptiste Delisle, Lucille Lopez-Delisle

Oct 11, 2023

CONTENTS:

CHAPTER

ONE

BARED (BAYESIAN APPROACH TO RETREIVE EXPRESSION DISTRIBUTION OF) SINGLE CELL

baredSC is a tool that uses a Monte-Carlo Markov Chain to estimate a confidence interval on the probability density function (PDF) of expression of one or two genes from single-cell RNA-seq data. It uses the raw counts and the total number of UMI for each cell. The PDF is approximated by a number of 1d or 2d gaussians provided by the user. The likelihood is estimated using the asumption that the raw counts follow a Poisson distribution of parameter equal to the proportion of mRNA for the gene in the cell multiplied by the total number of UMI identified in this cell.

We encourage users which are not familiar with MCMC to start with the *Installation* page to install the baredSC package. Then, follow the tutorial for a single gene (*Run baredSC_1d with default parameters*) which is a step by step example of application using an input provided on the github repository with default parameters. The other tutorials describe influences of some parameters or use a 2d example.

Users familiar with MCMC could directly go to the *Usage* page and read the *Outputs* page.

Advanced users who wants to use baredSC as a package within python should go to the *Use baredSC within python*.

1.1 Installation

1.1.1 Requirements

It as only been tested on linux but should work on MacOS.

It requires python >= 3.7 (tested 3.7.3 and 3.9.1)

Dependencies of classical python packages:

- numpy (tested 1.16.4 and 1.19.5)
- matplotlib (tested 3.1.1 and 3.3.4)
- pandas (tested 0.25.0 and 1.2.1)
- scipy (tested 1.3.0 and 1.6.0)

Dependencies of a python package from Jean-Baptiste Delisle dedicated to mcmc:

- samsam (above 0.1.2)

1.1.2 Installation

You can install it with pip:

```
$ pip install baredSC
```

Or with conda:

```
$ conda create -n baredSC -c bioconda -c conda-forge baredsc
```

1.2 Usage

1.2.1 General usage

Here is a description of all possible parameters. The tool take around 1 minute for 1d, 2000 cells, default parameters, independently of the number of gaussian and around 30 seconds for 300 cells. For the 2d 300 cells is around 3 minutes, 2000 cells around 15 minutes. Increasing the number of samples in the MCMC or in the burning phase will increase the time.

1.2.2 baredSC_1d

Run mcmc to get the pdf for a given gene using a normal distributions. The full documentation is available at https://baredsc.readthedocs.io

```
usage: baredSC_1d [-h] (--input INPUT | --inputAnnData INPUTANNDATA)
                  --geneColName GENECOLNAME
                  [--metadata1ColName METADATA1COLNAME]
                  [--metadata1Values METADATA1VALUES]
                  [--metadata2ColName METADATA2COLNAME]
                  [--metadata2Values METADATA2VALUES]
                  [--metadata3ColName METADATA3COLNAME]
                  [--metadata3Values METADATA3VALUES] [--xmin XMIN]
                  [--xmax XMAX] [--xscale {Seurat,log}]
                  [--targetSum TARGETSUM] [--nx NX] [--osampx OSAMPX]
                  [--osampxpdf OSAMPXPDF] [--minScale MINSCALE]
                  [--nnorm NNORM] [--nsampMCMC NSAMPMCMC]
                  [--nsampBurnMCMC NSAMPBURNMCMC]
                  [--nsplitBurnMCMC NSPLITBURNMCMC] [--T0BurnMCMC T0BURNMCMC]
                  [--seed SEED] [--minNeff MINNEFF] [--force] --output OUTPUT
                  [--figure FIGURE] [--title TITLE]
                  [--removeFirstSamples REMOVEFIRSTSAMPLES]
                  [--nsampInPlot NSAMPINPLOT] [--prettyBins PRETTYBINS]
                  [--logevidence LOGEVIDENCE] [--coviscale COVISCALE]
                  [--nis NIS] [--version]
```

Named Arguments

--version show program's version number and exit

Required arguments

--input Input table (tabular separated with header) with one line per cell columns with raw counts and one column nCount_RNA with total number of UMI per cell optionally other meta data to filter.

--inputAnnData Input annData (for example from Scanpy).

--geneColName Name of the column with gene counts.

--output Ouput file basename (will be npz) with results of mcmc.

Optional arguments to select input data

--metadata1ColName Name of the column with metadata1 to filter.

--metadata1Values Comma separated values for metadata1 of cells to keep.

--metadata2ColName Name of the column with metadata2 to filter.

--metadata2Values Comma separated values for metadata2 of cells to keep.

--metadata3ColName Name of the column with metadata3 to filter.

--metadata3Values Comma separated values for metadata3 of cells to keep.

Optional arguments to run MCMC

--xmin Minimum value to consider in x axis.

Default: 0

--xmax Maximum value to consider in x axis.

Default: 2.5

--xscale Possible choices: Seurat, log

scale for the x-axis: Seurat (log(1+targetSum*X)) or log (log(X))

Default: "Seurat"

--targetSum factor when Seurat scale is used: (log(1+targetSum*X)) (default is 10^4, use 0 for the median of nRNA_Counts)

Default: 10000

--nx Number of values in x to check how your evaluated pdf is compatible with the model.

Default: 100

--osampx Oversampling factor of x values when evaluating pdf of Poisson distribution.

Default: 10

--osampxpdf Oversampling factor of x values when evaluating pdf at each step of the MCMC.

Default: 5

--minScale Minimal value of the scale of gaussians (Default is 0.1 but cannot be smaller than max of twice the bin size of pdf evaluation and half the bin size).

Default: 0.1

--nnorm Number of gaussian to fit.

Default: 2

--nsampMCMC Number of samplings (iteractions) of mcmc.

Default: 100000

--nsampBurnMCMC Number of samplings (iteractions) in the burning phase of mcmc (Default is nsampMCMC / 4).

--nsplitBurnMCMC Number of steps in the burning phase of mcmc.

Default: 10

--T0BurnMCMC Initial temperature in the burning phase of mcmc (>1).

Default: 100.0

--seed Change seed for another output.

Default: 1

--minNeff Will redo the MCMC with 10 times more samples until the number of effective samples that this value (Default is not set so will not rerun MCMC).

--force Force to redo the mcmc even if output exists.

Optional arguments to get plots and text outputs

--figure Ouput figure filename.

--title Title in figures.

--removeFirstSamples Number of samples to ignore before making the plots (default is nsampMCMC / 4).

--nsampInPlot Approximate number of samples to use in plots.

Default: 100000

--prettyBins Number of bins to use in plots (Default is nx).

Optional arguments to get logevidence

--logevidence Ouput file to put logevidence value.

--coviscale Scale factor to apply to covariance of parameters to get random parameters in logevidence evaluation.

Default: 1

--nis Size of sampling of random parameters in logevidence evaluation.

Default: 1000

1.2.3 combineMultipleModels_1d

Combine mcmc results from multiple models to get a mixture using logevidence to infer weights.

```
usage: combineMultipleModels_1d [-h]
                                (--input INPUT | --inputAnnData INPUTANNDATA)
                                --geneColName GENECOLNAME
                                [--metadata1ColName METADATA1COLNAME]
                                [--metadata1Values METADATA1VALUES]
                                [--metadata2ColName METADATA2COLNAME]
                                [--metadata2Values METADATA2VALUES]
                                [--metadata3ColName METADATA3COLNAME]
                                [--metadata3Values METADATA3VALUES] --outputs
                                OUTPUTS [OUTPUTS ...] [--xmin XMIN]
                                [--xmax XMAX] [--xscale {Seurat,log}]
                                [--targetSum TARGETSUM] [--nx NX]
                                [--osampx OSAMPX] [--osampxpdf OSAMPXPDF]
                                [--minScale MINSCALE] [--seed SEED] --figure
                                FIGURE [--title TITLE]
                                [--removeFirstSamples REMOVEFIRSTSAMPLES]
                                [--nsampInPlot NSAMPINPLOT]
                                [--prettyBins PRETTYBINS]
                                [--logevidences LOGEVIDENCES [LOGEVIDENCES ...]]
                                [--coviscale COVISCALE] [--nis NIS]
                                [--version]
```

Named Arguments

--version show program's version number and exit

Required arguments

--input Input table (tabular separated with header) with one line per cell columns with raw counts and one column nCount_RNA with total number of UMI per cell optionally other meta data to filter.

--inputAnnData Input annData (for example from Scanpy).

--geneColName Name of the column with gene counts.

--outputs Ouput files basename (will be npz) with different results of mcmc to combine.

--figure Ouput figure basename.

Optional arguments used to run MCMC

--xmin Minimum value to consider in x axis.

Default: 0

--xmax Maximum value to consider in x axis.

Default: 2.5

--xscale Possible choices: Seurat, log

scale for the x-axis: Seurat (log(1+targetSum*X)) or log (log(X))

Default: "Seurat"

--targetSum factor when Seurat scale is used: (log(1+targetSum*X)) (default is 10^4, use 0 for the median of nRNA_Counts)

Default: 10000

--nx Number of values in x to check how your evaluated pdf is compatible with the model.

Default: 100

--osampx Oversampling factor of x values when evaluating pdf of Poisson distribution.

Default: 10

--osampxpdf Oversampling factor of x values when evaluating pdf at each step of the MCMC.

Default: 5

--minScale Minimal value of the scale of gaussians (Default is 0.1 but cannot be smaller than max of twice the bin size of pdf evaluation and half the bin size).

Default: 0.1

--seed Change seed for another output.

Default: 1

Optional arguments to select input data

--metadata1ColName Name of the column with metadata1 to filter.

--metadata1Values Comma separated values for metadata1 of cells to keep.

--metadata2ColName Name of the column with metadata2 to filter.

--metadata2Values Comma separated values for metadata2 of cells to keep.

--metadata3ColName Name of the column with metadata3 to filter.

--metadata3Values Comma separated values for metadata3 of cells to keep.

Optional arguments to customize plots and text outputs

--title Title in figures.

--removeFirstSamples Number of samples to ignore before making the plots (default is nsampMCMC / 4).

--nsampInPlot Approximate number of samples to use in plots.

Default: 100000

--prettyBins Number of bins to use in plots (Default is nx).

Optional arguments to evaluate logevidence

--logevidences Ouput files of precalculated log evidence values.(if not provided will be calculated).

--coviscale Scale factor to apply to covariance of parameters to get random parameters in logevidence evaluation.

Default: 1

--nis Size of sampling of random parameters in logevidence evaluation.

Default: 1000

1.2.4 baredSC_2d

Run mcmc to get the pdf in 2D for 2 given genes using a normal distributions. The full documentation is available at https://baredsc.readthedocs.io

```
usage: baredSC_2d [-h] (--input INPUT | --inputAnnData INPUTANNDATA)
                  --geneXColName GENEXCOLNAME --geneYColName GENEYCOLNAME
                  [--metadata1ColName METADATA1COLNAME]
                  [--metadata1Values METADATA1VALUES]
                  [--metadata2ColName METADATA2COLNAME]
                  [--metadata2Values METADATA2VALUES]
                  [--metadata3ColName METADATA3COLNAME]
                  [--metadata3Values METADATA3VALUES] [--xmin XMIN]
                  [--xmax XMAX] [--nx NX] [--osampx OSAMPX]
                  [--osampxpdf OSAMPXPDF] [--minScalex MINSCALEX]
                  [--ymin YMIN] [--ymax YMAX] [--ny NY] [--osampy OSAMPY]
                  [--osampypdf OSAMPYPDF] [--minScaley MINSCALEY]
                  [--scalePrior SCALEPRIOR] [--scale {Seurat,log}]
                  [--targetSum TARGETSUM] [--nnorm NNORM]
                  [--nsampMCMC NSAMPMCMC] [--nsampBurnMCMC NSAMPBURNMCMC]
                  [--nsplitBurnMCMC NSPLITBURNMCMC] [--T0BurnMCMC T0BURNMCMC]
                  [--seed SEED] [--minNeff MINNEFF] [--force] --output OUTPUT
                  [--figure FIGURE] [--title TITLE]
                  [--splity SPLITY [SPLITY ...]]
                  [--removeFirstSamples REMOVEFIRSTSAMPLES]
                  [--nsampInPlot NSAMPINPLOT] [--prettyBinsx PRETTYBINSX]
                  [--prettyBinsy PRETTYBINSY] [--log1pColorScale]
```

(continues on next page)

(continued from previous page)

```
                   [--logevidence LOGEVIDENCE] [--coviscale COVISCALE]
                   [--nis NIS] [--version]
```

Named Arguments

--version show program's version number and exit

Required arguments

--input Input table (tabular separated with header) with one line per cell columns with raw counts and one column nCount_RNA with total number of UMI per cell optionally other meta data to filter.

--inputAnnData Input annData (for example from Scanpy).

--geneXColName Name of the column with gene counts for gene in x.

--geneYColName Name of the column with gene counts for gene in y.

--output Ouput file basename (will be npz) with results of mcmc.

Optional arguments to select input data

--metadata1ColName Name of the column with metadata1 to filter.

--metadata1Values Comma separated values for metadata1 of cells to keep.

--metadata2ColName Name of the column with metadata2 to filter.

--metadata2Values Comma separated values for metadata2 of cells to keep.

--metadata3ColName Name of the column with metadata3 to filter.

--metadata3Values Comma separated values for metadata3 of cells to keep.

Optional arguments to run MCMC

--xmin Minimum value to consider in x axis.

Default: 0

--xmax Maximum value to consider in x axis.

Default: 2.5

--nx Number of values in x to check how your evaluated pdf is compatible with the model.

Default: 50

--osampx Oversampling factor of x values when evaluating pdf of Poisson distribution.

Default: 10

--osampxpdf Oversampling factor of x values when evaluating pdf at each step of the MCMC.

Default: 4

--minScalex Minimal value of the scale of gaussians on x (Default is 0.1 but cannot be smaller than max of twice the bin size of pdf evaluation and half the bin size on x axis).

Default: 0.1

--ymin Minimum value to consider in y axis.

Default: 0

--ymax Maximum value to consider in y axis.

Default: 2.5

--ny Number of values in y to check how your evaluated pdf is compatible with the model.

Default: 50

--osampy Oversampling factor of y values when evaluating pdf of Poisson distribution.

Default: 10

--osampypdf Oversampling factor of y values when evaluating pdf at each step of the MCMC.

Default: 4

--minScaley Minimal value of the scale of gaussians on yx (Default is 0.1 but cannot be smaller than max of twice the bin size of pdf evaluation and half the bin size on y axis).

Default: 0.1

--scalePrior Scale of the truncnorm used in the prior for the correlation.

Default: 0.3

--scale Possible choices: Seurat, log

scale for the x-axis and y-axis: Seurat (log(1+targetSum*X)) or log (log(X))

Default: "Seurat"

--targetSum factor when Seurat scale is used: (log(1+targetSum*X)) (default is 10^4, use 0 for the median of nRNA_Counts)

Default: 10000

--nnorm Number of gaussian 2D to fit.

Default: 1

--nsampMCMC Number of samplings (iteractions) of mcmc.

Default: 100000

--nsampBurnMCMC Number of samplings (iteractions) in the burning phase of mcmc (Default is nsampMCMC / 4).

--nsplitBurnMCMC Number of steps in the burning phase of mcmc.

Default: 10

--T0BurnMCMC Initial temperature in the burning phase of mcmc.

Default: 100.0

--seed Change seed for another output.

Default: 1

--minNeff Will redo the MCMC with 10 times more samples until the number of effective samples that this value (Default is not set so will not rerun MCMC).

--force Force to redo the mcmc even if output exists.

Optional arguments to get plots and text outputs

--figure Ouput figure basename.

--title Title in figures.

--splity Threshold value to plot the density for genex for 2 categories in geney values.

--removeFirstSamples Number of samples to ignore before making the plots (default is nsampMCMC / 4).

--nsampInPlot Approximate number of samples to use in plots.

Default: 100000

--prettyBinsx Number of bins to use in x in plots (Default is nx).

--prettyBinsy Number of bins to use in y in plots (Default is ny).

--log1pColorScale Use log1p color scale instead of linear color scale.

Default: False

Optional arguments to get logevidence

--logevidence Ouput file to put logevidence value.

--coviscale Scale factor to appy to covariance of parameters to get random parameters in logevidence evaluation.

Default: 1

--nis Size of sampling of random parameters in logevidence evaluation.

Default: 1000

1.2.5 combineMultipleModels_2d

Combine mcmc 2D results from multiple models to get a mixture using logevidence to infer weights.

```
usage: combineMultipleModels_2d [-h]
                                (--input INPUT | --inputAnnData INPUTANNDATA)
                                --geneXColName GENEXCOLNAME --geneYColName
                                GENEYCOLNAME
                                [--metadata1ColName METADATA1COLNAME]
                                [--metadata1Values METADATA1VALUES]
                                [--metadata2ColName METADATA2COLNAME]
                                [--metadata2Values METADATA2VALUES]
                                [--metadata3ColName METADATA3COLNAME]
                                [--metadata3Values METADATA3VALUES] --outputs
                                OUTPUTS [OUTPUTS ...] [--xmin XMIN]
                                [--xmax XMAX] [--nx NX] [--osampx OSAMPX]
```

(continues on next page)

(continued from previous page)

```
                                [--osampxpdf OSAMPXPDF]
                                [--minScalex MINSCALEX] [--ymin YMIN]
                                [--ymax YMAX] [--ny NY] [--osampy OSAMPY]
                                [--osampypdf OSAMPYPDF]
                                [--minScaley MINSCALEY] [--scale {Seurat,log}]
                                [--scalePrior SCALEPRIOR]
                                [--targetSum TARGETSUM] [--seed SEED] --figure
                                FIGURE [--title TITLE]
                                [--splity SPLITY [SPLITY ...]]
                                [--removeFirstSamples REMOVEFIRSTSAMPLES]
                                [--nsampInPlot NSAMPINPLOT]
                                [--prettyBins PRETTYBINS]
                                [--prettyBinsx PRETTYBINSX]
                                [--prettyBinsy PRETTYBINSY]
                                [--log1pColorScale] [--getPVal]
                                [--logevidences LOGEVIDENCES [LOGEVIDENCES ...]]
                                [--coviscale COVISCALE] [--nis NIS]
                                [--version]
```

Named Arguments

--version show program’s version number and exit

Required arguments

--input Input table (tabular separated with header) with one line per cell columns with raw counts and one column nCount_RNA with total number of UMI per cell optionally other meta data to filter.

--inputAnnData Input annData (for example from Scanpy).

--geneXColName Name of the column with gene counts for gene in x.

--geneYColName Name of the column with gene counts for gene in y.

--outputs Ouput files basename (will be npz) with different results of mcmc to combine.

--figure Ouput figure basename.

Optional arguments used to run MCMC

--xmin Minimum value to consider in x axis.

Default: 0

--xmax Maximum value to consider in x axis.

Default: 2.5

--nx Number of values in x to check how your evaluated pdf is compatible with the model.

Default: 50

--osampx

Oversampling factor of x values when evaluating pdf of Poisson distribution.

Default: 10

--osampxpdf

Oversampling factor of x values when evaluating pdf at each step of the MCMC.

Default: 4

--minScalex

Minimal value of the scale of gaussians on x (Default is 0.1 but cannot be smaller than max of twice the bin size of pdf evaluation and half the bin size on x axis).

Default: 0.1

--ymin

Minimum value to consider in y axis.

Default: 0

--ymax

Maximum value to consider in y axis.

Default: 2.5

--ny

Number of values in y to check how your evaluated pdf is compatible with the model.

Default: 50

--osampy

Oversampling factor of y values when evaluating pdf of Poisson distribution.

Default: 10

--osampypdf

Oversampling factor of y values when evaluating pdf at each step of the MCMC.

Default: 4

--minScaley

Minimal value of the scale of gaussians on yx (Default is 0.1 but cannot be smaller than max of twice the bin size of pdf evaluation and half the bin size on y axis).

Default: 0.1

--scale

Possible choices: Seurat, log

scale for the x-axis and y-axis: Seurat (log(1+targetSum*X)) or log (log(X))

Default: “Seurat”

--scalePrior

Scale of the truncnorm used in the prior for the correlation.

Default: 0.3

--targetSum

factor when Seurat scale is used: (log(1+targetSum*X)) (default is 10^4, use 0 for the median of nRNA_Counts)

Default: 10000

--seed

Change seed for another output.

Default: 1

Optional arguments to select input data

--metadata1ColName Name of the column with metadata1 to filter.

--metadata1Values Comma separated values for metadata1 of cells to keep.

--metadata2ColName Name of the column with metadata2 to filter.

--metadata2Values Comma separated values for metadata2 of cells to keep.

--metadata3ColName Name of the column with metadata3 to filter.

--metadata3Values Comma separated values for metadata3 of cells to keep.

Optional arguments to customize plots and text outputs

--title Title in figures.

--splity Threshold value to plot the density for genex for 2 categories in geney values.

--removeFirstSamples Number of samples to ignore before making the plots (default is nsampMCMC / 4).

--nsampInPlot Approximate number of samples to use in plots.

Default: 100000

--prettyBins Number of bins to use in plots (Default is nx).

--prettyBinsx Number of bins to use in x in plots (Default is nx).

--prettyBinsy Number of bins to use in y in plots (Default is ny).

--log1pColorScale Use log1p color scale instead of linear color scale.

Default: False

--getPVal Use less samples to get an estimation of the p-value.

Default: False

Optional arguments to evaluate logevidence

--logevidences Ouput files of precalculated log evidence values.(if not provided will be calculated).

--coviscale Scale factor to apply to covariance of parameters to get random parameters in logevidence evaluation.

Default: 1

--nis Size of sampling of random parameters in logevidence evaluation.

Default: 1000

1.3 Outputs

We will describe here each of the output file.

1.3.1 MCMC output

The only output by defaut is a numpy compressed `.npz` file. This output contains the result of the MCMC:

- samples: the value of the parameter at each step of the MCMC
  - In the 1d case, the parameters are mu0, scale0, (amp1, mu1, scale1, . . .). The first amplitude can be deduced as 1 minus the sum of all other amplitudes.
  - In the 2d case, the parameters are mux0, scalex0, muy0, scaley0, corr0, (amp1, mux1, scalex1, muy1, scaley1, corr1, ..). The first amplitude can be deduced as 1 minus the sum of all other amplitudes.
- diagnostics: a dictionary with the diagnostics at each step of the MCMC, among them:
  - logprob: the log probability at each step of the MCMC
  - mu: the final estimate of the mean of each parameter
  - cov: the final estimate of the covariance matrix of parameters
- some of the input values

When the tool is run while the output exists it will use it instead of rerunning it which is useful to get more plots.

1.3.2 Plots and txt outputs

When `--figure name.extension` is given then some QC and results are given.

QC

name_convergence.extension

This plot shows the autocorrelation between samples for all parameters (the solid line shows the median, the shaded area shows the min and max). If the MCMC converged, you should see a value of ACF close to 0 since a small value of T.

name_neff.txt

From the autocorrelation displayed above, we can evaluate the number of independent samples, also called effective sample size. The value is printed and stored in this text file.

name_p.extension

This plot shows the value of each parameter and the log probability (y axis, one panel per parameter) for all samples (x-axis). When the MCMC did not converged, it can be helpful to see if it can be explained by the fact that the first samples considered where not around the final solution. In this case, it can be useful to rerun the plots using an increase value of `--removeFirstSamples` (by default it is 1/4 of the number of samples), or increase the number of samples.

name_corner.extension

This plot shows the distribution of value of each parameter in relationship the one with the other. It can help to see which parameters are correlated. Also, when the MCMC did not converged, it can help to identify if 2 or more solutions were explored.

Results

name.extension

This is the figure with the results.

- When the 1d version is used, it displays the mean pdf in solid red line, the median in black dashed lines (/!backslash the integral of the median is not equal to 1) with the confidence interval of 1 sigma (68%), 2 sigma (95%) and 3 sigma (99.7%) as well as in green, the kernel density estimate of the input values, the detected expression (`log(1 + targetSum * raw / total UMI)`).
- When the 2d version is used, it displays the pdf as a heatmap as well as a projection on the x and y axis. On the projection, the confidence interval 68% is indicated as a shaded area as well as the mean with a solid red line and the median with a dashed black line. On the top right corner, the correlation is indicated with the confidence interval 68% as well as a confidence interval on the one-sided p-value (the probability that the correlation is the opposite sign of the mean, one sigma confidence interval).

name_individuals.extension

- When the 1d version is used, it displays the pdf of 100 samples.
- When the 2d version is used, it displays the projection of the pdf of 100 samples.

name_p.txt

This is a tabulated delimited table with the 16 percentile (low), median, 84 percentile (high) value of each parameter.

name_pdf.txt (1d only)

For each value of x, the 16 percentile (low), mean, 84 percentile (high) and median, is given in a tabulated delimited file.

name_with_posterior.extension (1d only)

Same as name.extension except that a new orange line is plotted showing the posterior density evaluated as the average of the posterior density of each cell.

name_posterior_individuals.extension (1d only)

Showing posterior density probability of 50 random cells.

name_posterior_per_cell.txt (1d only)

For each cell of the input, providing the posterior average and standard deviation of the density probability.

name_posterior_andco.extension (1d only)

Showing the mean pdf, the median pdf, the density from raw counts normalized, the average of the posterior density from all cells, the density and a histogram using only the average value of the posterior distribution of each cell and the posterior density approximating the pdf of each cell by a Gaussian using values in the "posterior_per_cell.txt" file.

name_means.txt.gz (1d only)

Each line correspond to the value of the mean expression evaluated at each sample of the MCMC.

name_median.extension (2d only)

Same as name.extension except that the median instead of the mean is used.

name_corr.txt (2d only)

The mean, median, 16 percentile, 84 percentile, p-value and error on the p-value for the correlation (see above).

name_pdf2d.txt (2d only)

The mean pdf and the x and y values stored in a tabulated delimited file in a matrix format. Different x values correspond to different columns while different y values correspond to different rows.

name_pdf2d_flat.txt (2d only)

The x, y, 16 percentile (low), mean, 84 percentile (high) and median of pdf in a tabulated delimited file.

Results when `--splity` is provided in 2d

When `--splity` is provided the pdf above and below this threshold on the y axis are summed up, resulting in 2 pdf along the x axis.

name_splitX.extension

This plot shows the 2 pdfs. The ratio between the area represent the ratio of cells above and below the threshold of the gene y. The pdf for cells below the threshold is in red (with the shaded area for the 68% confidence interval) and the pdf for cells above the threshold is in green. In black is the pdf of all cells projected on the x axis (sum of the 2).

name_splitX_renorm.extension

Same plot as above except that the pdf were renormalized so the area of each pdf is equal to 1. Also the median is added in dashed black lines.

name_splitX.txt

This is a tabulated delimited table with the x values, the 16 percentile (low), mean, 84 percentile (high) values of each pdf (below and above the threshold) before normalization.

1.3.3 Evidence

When `--logevidence` is set. The log evidence is calculated and stored in this file. This can be used to compare different models, here different number of gaussians.

1.4 Tutorial on simulated data

1.4.1 Run baredSC_1d with default parameters

We will describe here an example step by step on simulated data where we will use default parameters and carefully check the QC.

Inputs

We took total UMI counts from a real dataset of NIH3T3. We generated a example where 2 genes have the same distribution (2 gaussians, one of mean 0.375, scale 0.125 and another one of mean 1 and scale 0.1). Half of cells goes in each gaussian. The gene is called "0.5_0_0_0.5_x". The input table can be downloaded from here.

Run

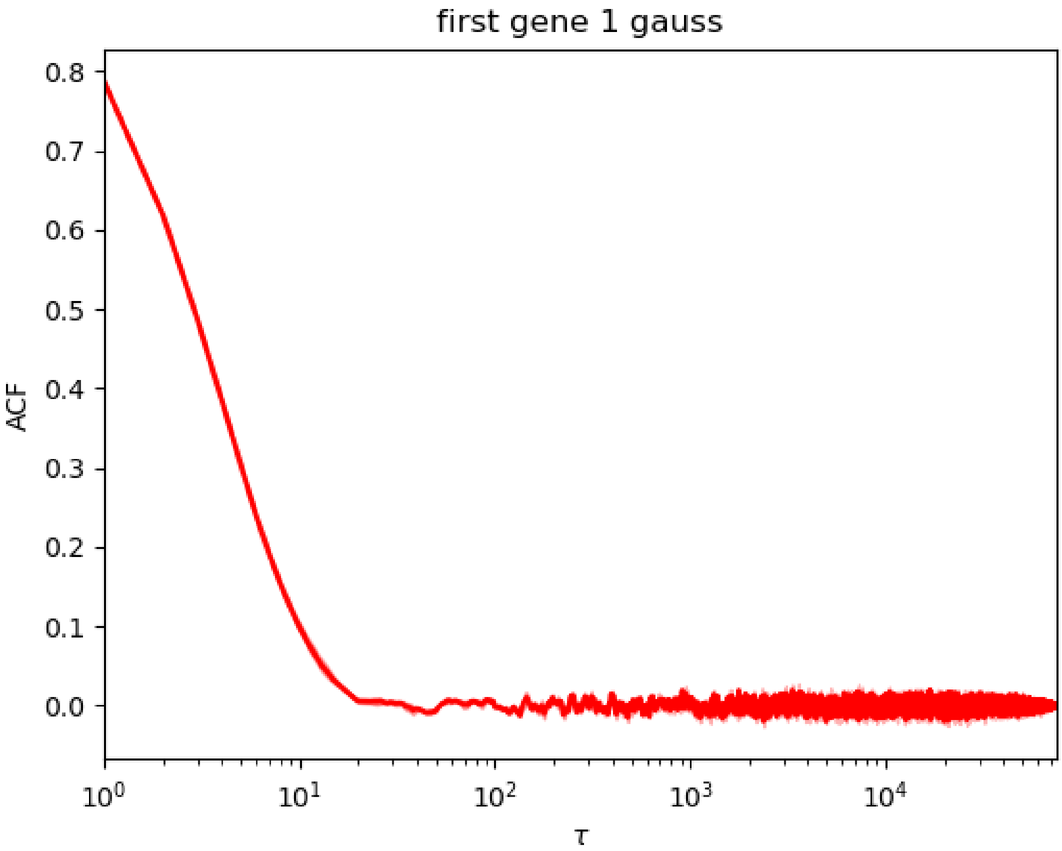
Run 1 gaussian

Let say we don't know the number of gaussian, we try one. We keep the default parameters and we set `--figure` to get visual outputs:

```
$ baredSC_1d \
    --input example/nih3t3_generated_2d_2.txt \
    --geneColName 0.5_0_0_0.5_x \
    --output example/first_example_1d_1gauss \
    --nnorm 1 \
    --figure example/first_example_1d_1gauss.png \
    --title "first gene 1 gauss" \
    --logevidence example/first_example_1d_1gauss_logevid.txt
```

Check QC

We first check the convergence:

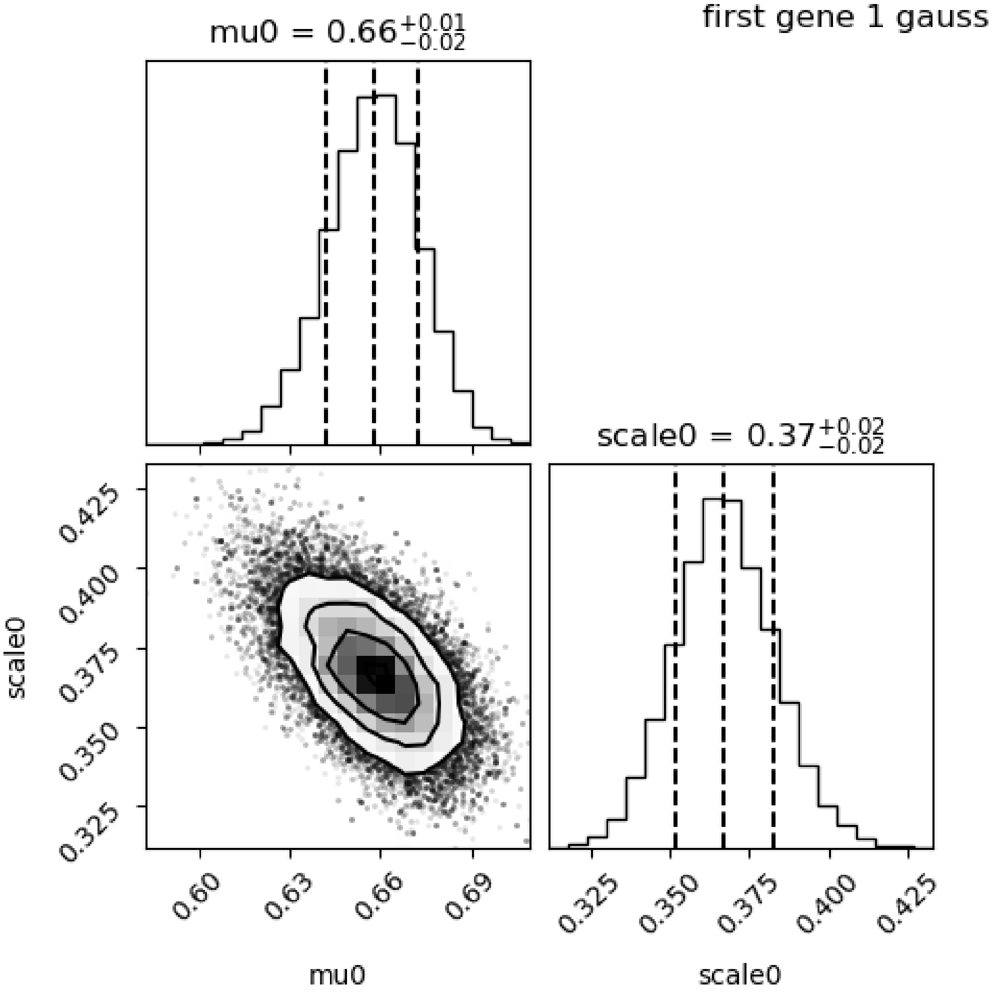


This plot show the autocorrelation as a function of number of samples. The earlier the curves goes close to 0, the more it converged.

Here, this is perfect.

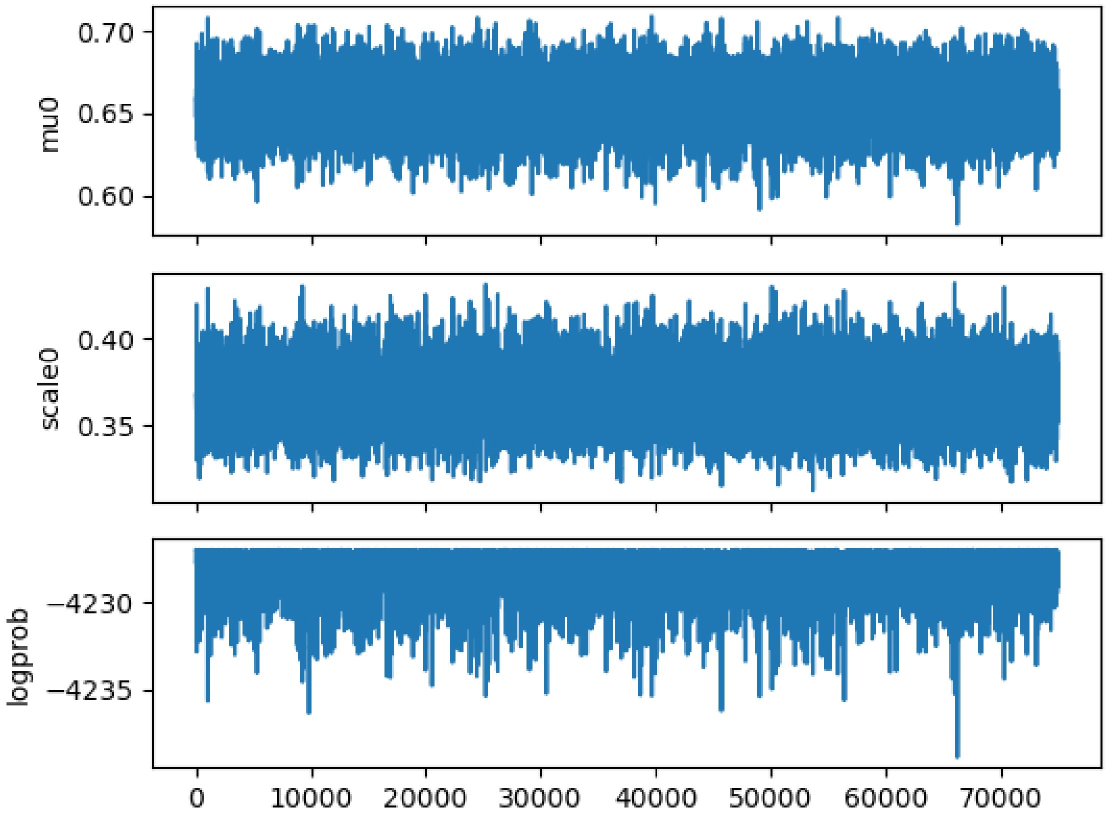
As printed during the run (or reading the file *neff.txt), the Neff is around 8000 which is enough to estimate the confidence interval.

We have a look at the corner plot:



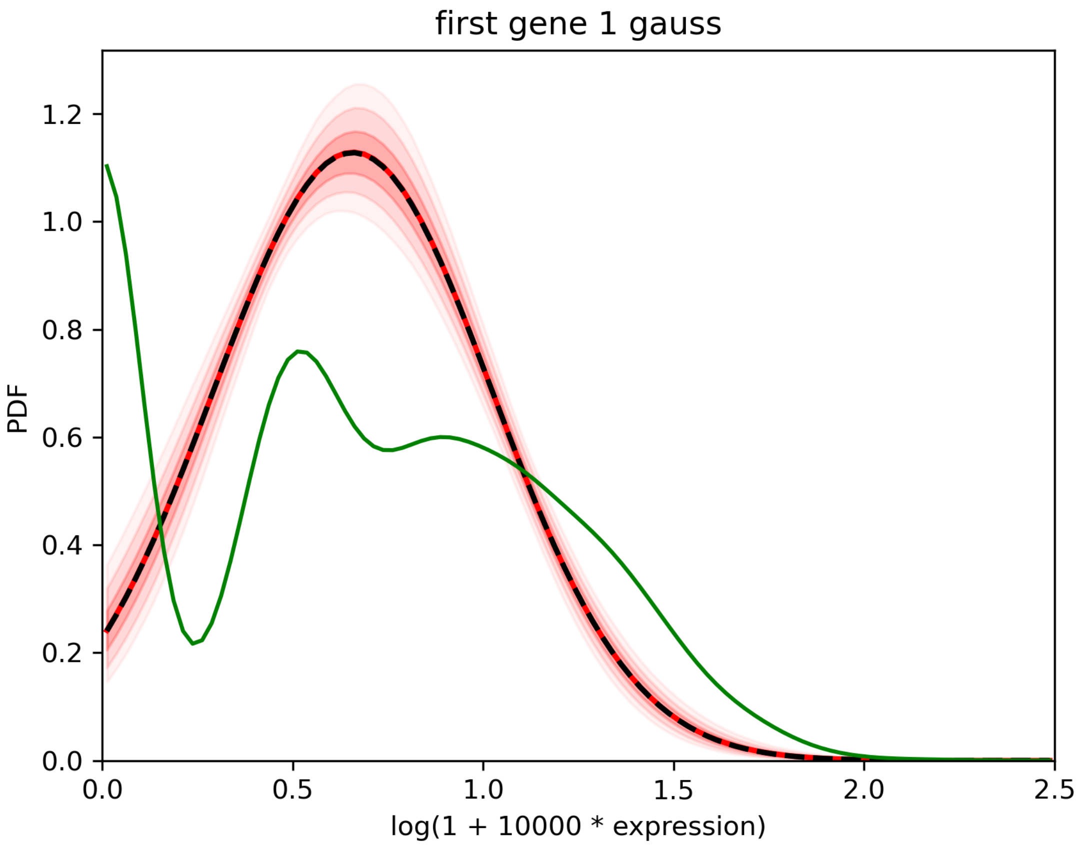
We see that the distribution of each parameter is close to gaussian, this is perfect.

We have a look at the parameter plot:



It nicely shows that the 2 parameters (mu0 and scale0) oscilate around the mean position while the log probability oscilate with a maximum value.

Look at the results



The pdf is well constrained (the shaded areas indicating the 68%, 95% and 99% interval are thin). The mean in red is really close to the median in dashed black line. However, using the green curve which is the density of the detected expression, we suspect that there are 2 gaussians.

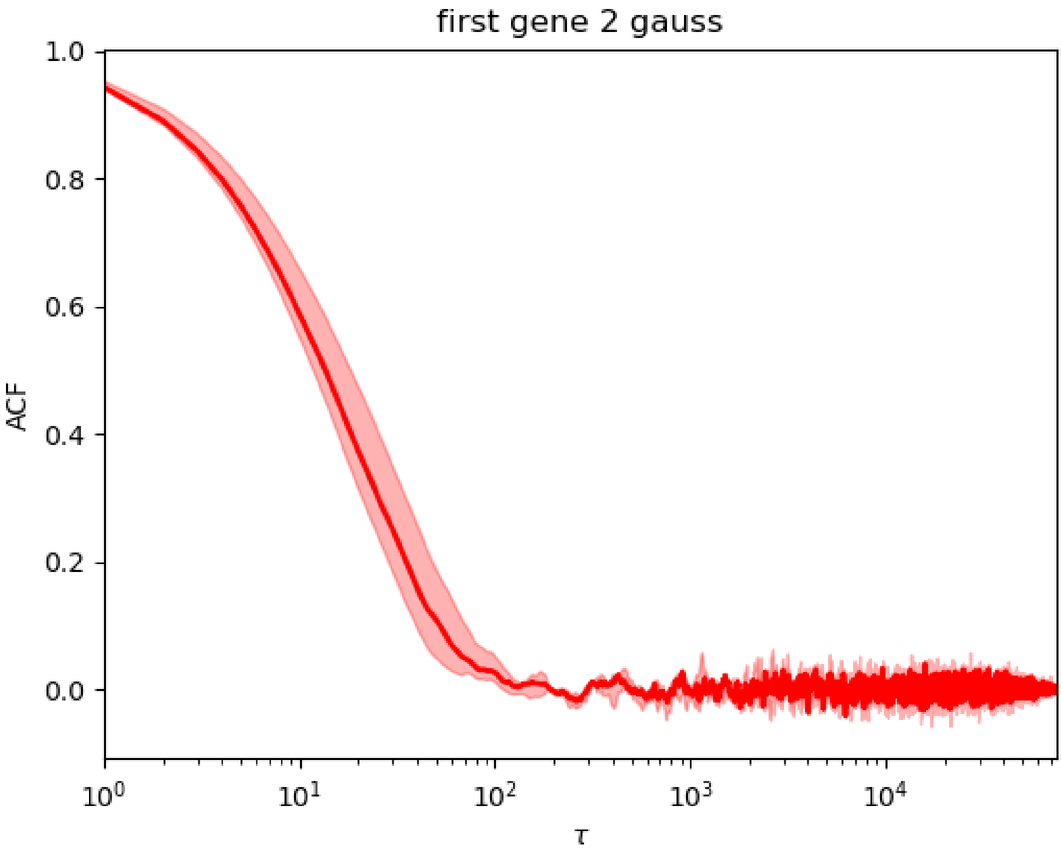
Run 2 gaussians

Now let's try 2 gaussians

```
$ baredSC_1d \
    --input example/nih3t3_generated_2d_2.txt \
    --geneColName 0.5_0_0_0.5_x \
    --output example/first_example_1d_2gauss \
    --nnorm 2 \
    --figure example/first_example_1d_2gauss.png \
    --title "first gene 2 gauss" \
    --logevidence example/first_example_1d_2gauss_logevid.txt
```

Check QC

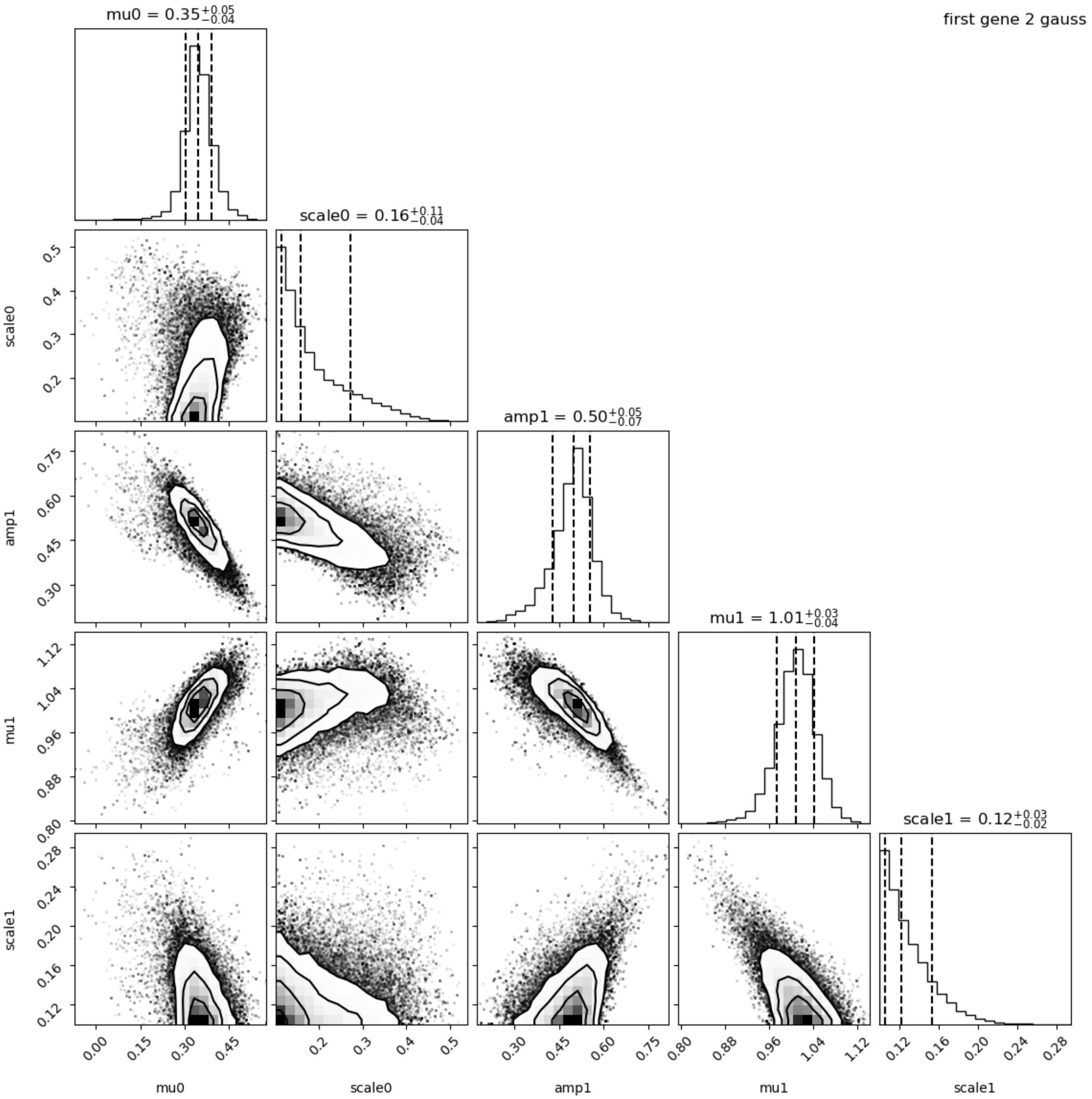
We first check the convergence:



This is perfect.

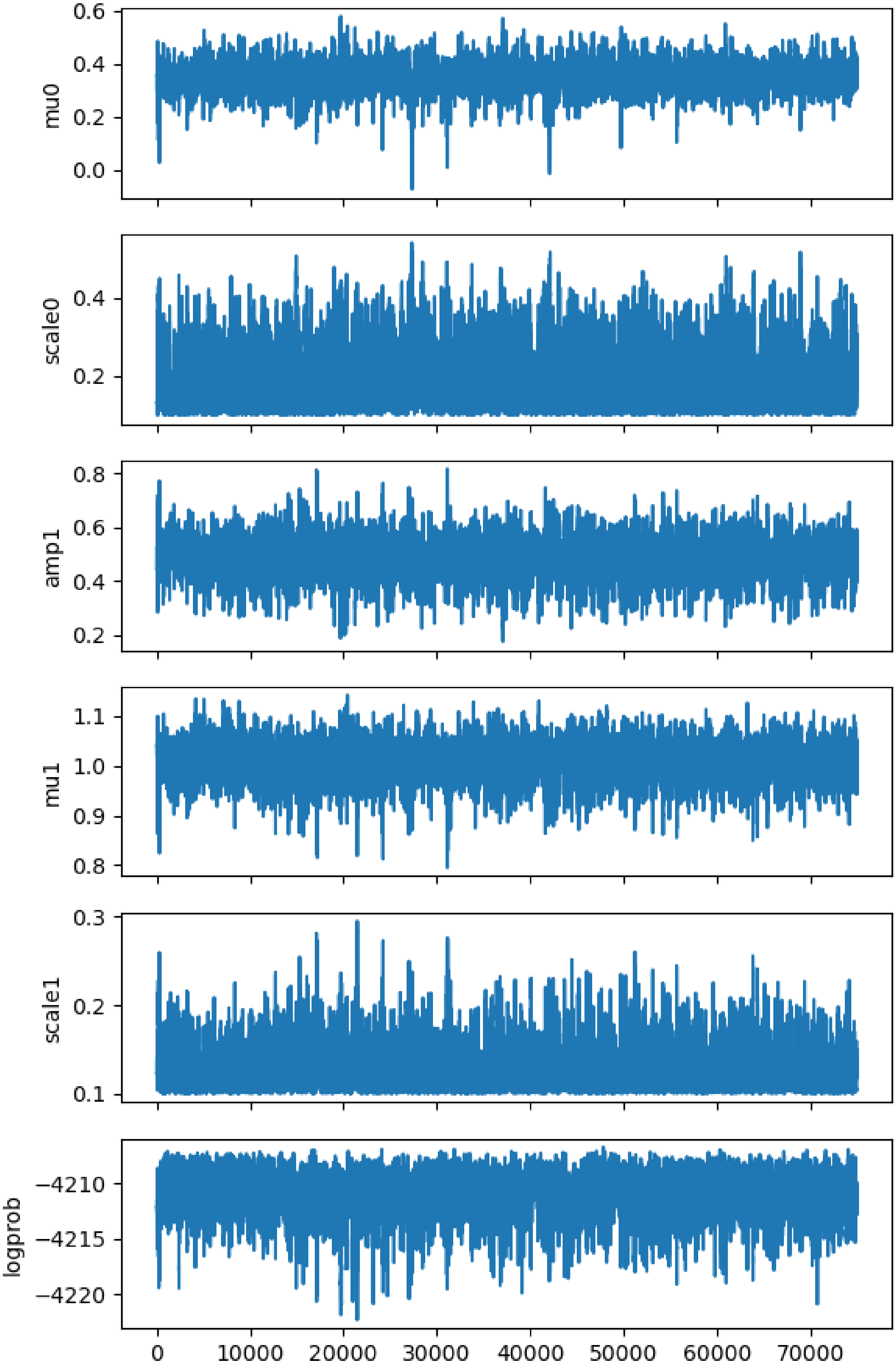
As printed during the run (or reading the file *neff.txt), the Neff is around 1300, perfect.

We have a look at the corner plot:



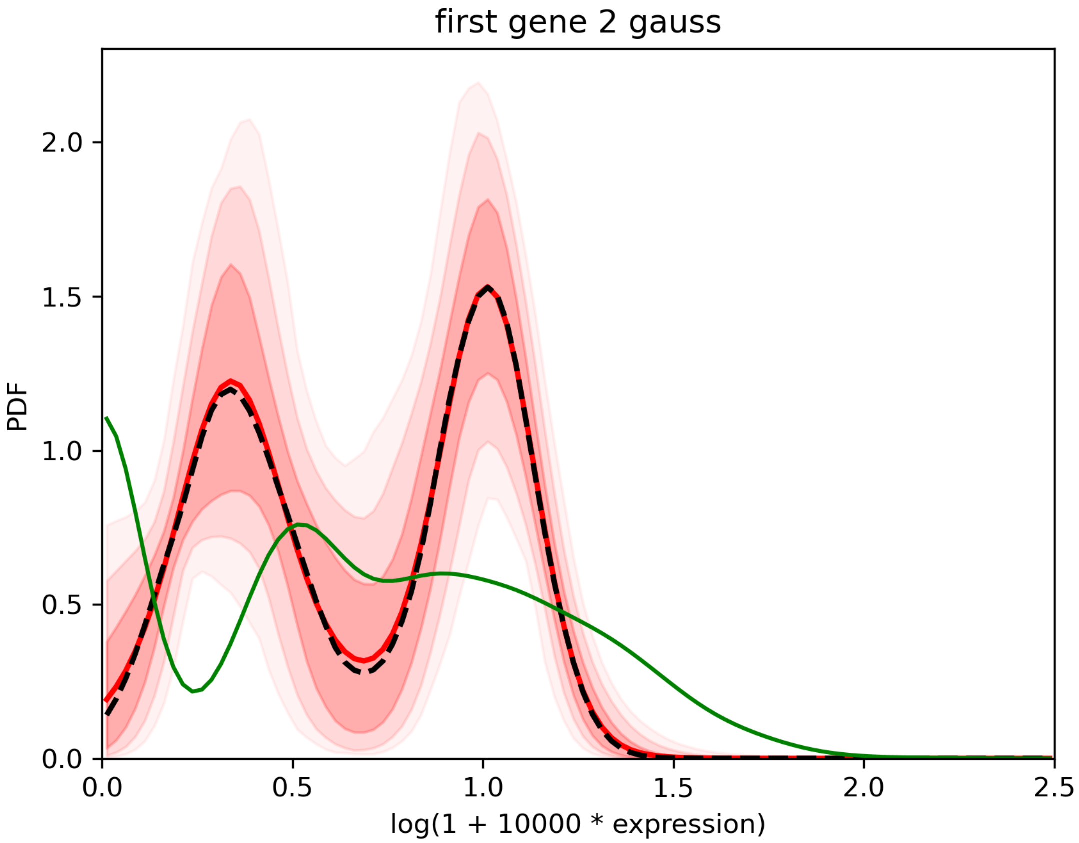
The means and amplitude are like a gaussian. The scale distribution is asymetric because by default, the minimum scale is set to 0.1 which is close to our values here. Some parameters are correlated: the mean of the first Gaussian with the mean of the second Gaussian. Some are anti-correlated: the mean of the second Gaussian with its amplitude. But this is not problematic, just an information we can get from this plot.

We have a look at the parameter plot:


0.6
0.4
0.2
0.0
mu0
0.4
0.2
scale0
0.8
0.6
0.4
0.2
amp1
1.1
1.0
0.9
0.8
mu1
0.3
0.2
0.1
scale1
−4210
−4215
−4220
logprob
0
10000
20000
30000
40000
50000
60000
70000

It nicely shows that the 5 parameters oscilate around the mean position and the log probability is quite constant.

Look at the results



The confidence interval is larger than in the first case but still good.

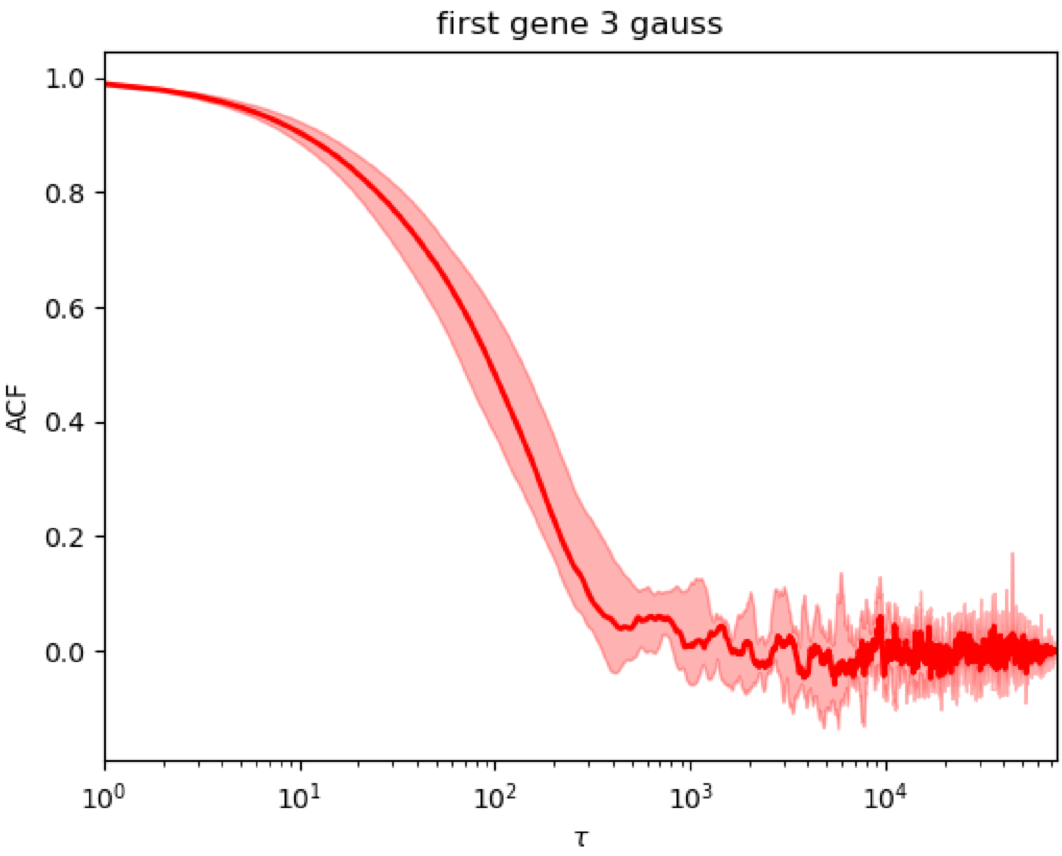
Run 3 gaussians

Now let’s try 3 gaussians

```
$ baredSC_1d \
    --input example/nih3t3_generated_2d_2.txt \
    --geneColName 0.5_0_0_0.5_x \
    --output example/first_example_1d_3gauss \
    --nnorm 3 \
    --figure example/first_example_1d_3gauss.png \
    --title "first gene 3 gauss" \
    --logevidence example/first_example_1d_3gauss_logevid.txt
```

Check QC

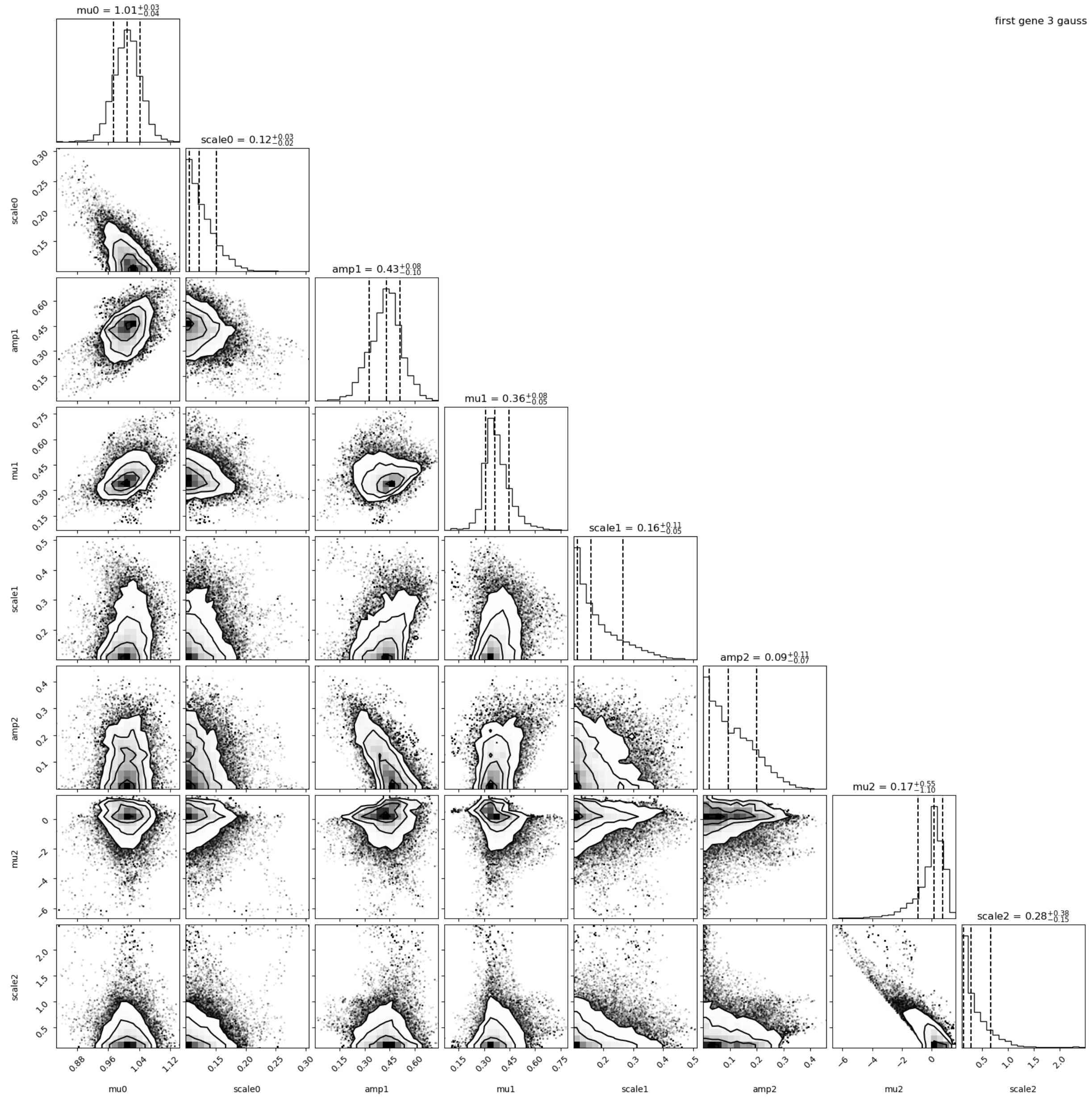
We first check the convergence:



It is much worse than the first ones. The auto-correlation decreases later and does not stay a flat line at 0 but oscillate.

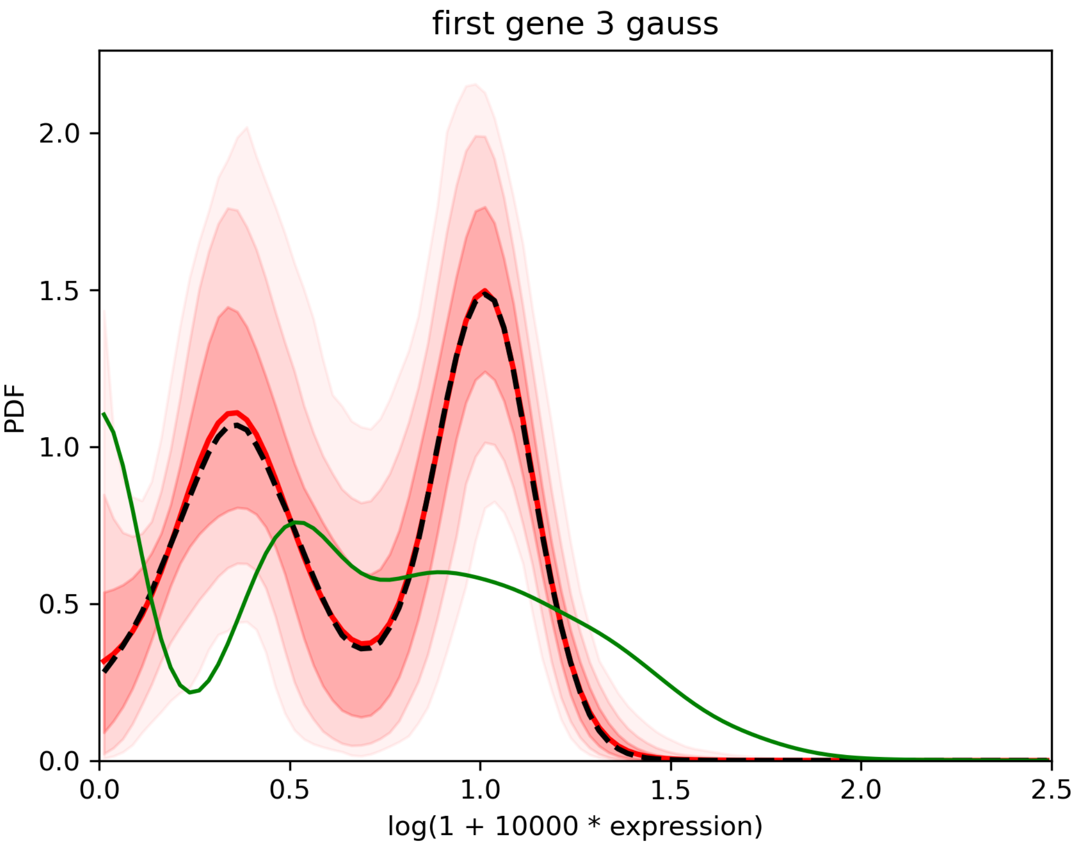
As printed during the run, the Neff is around 191. This is better to get more indenpendent samples. We can rerun with another value of the seed but it is safer to rerun with increased number of samples.

We still have a look at the corner plot:



The first two Gaussians are close to what was expected. The third Gaussian is a Gaussian with a reduced mean (0.17 in average). We see that this last Gaussian is not very well constrained (large error bar on each of its parameters).

Look at the results

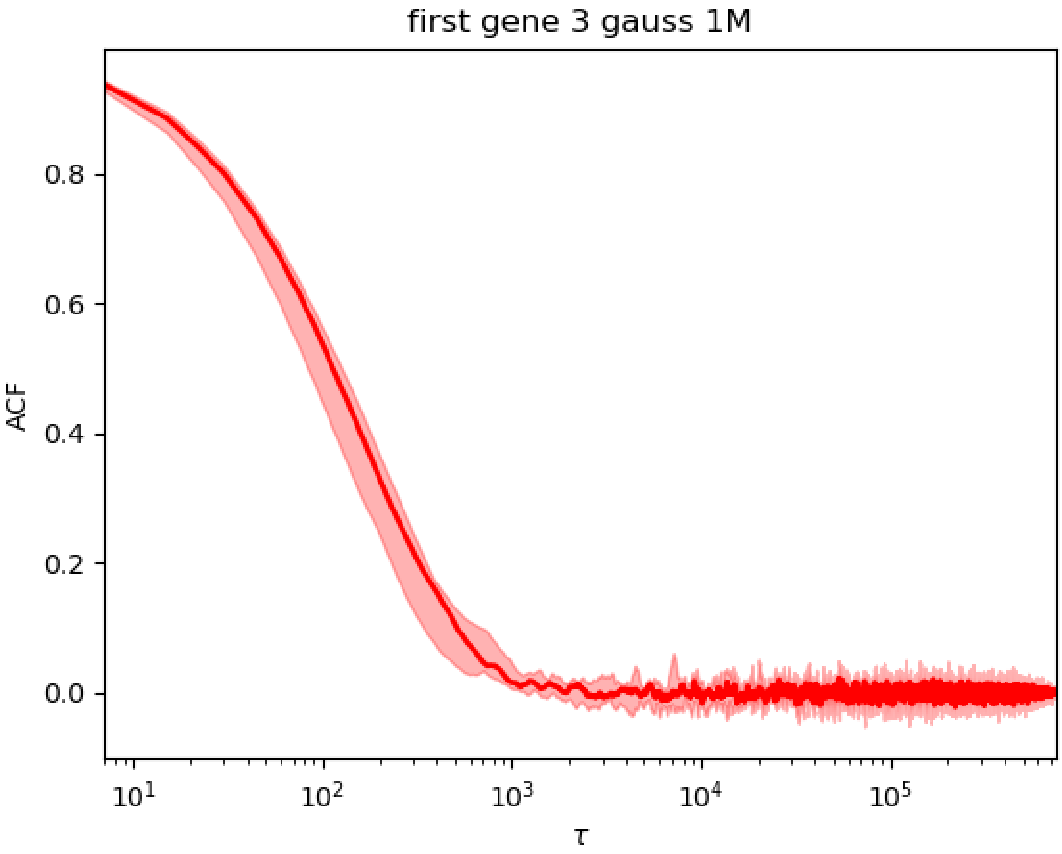


The results are very close to the one with 2 Gaussians.

Rerun 3 gauss with more samples

```
$ baredSC_1d \
    --input example/nih3t3_generated_2d_2.txt \
    --geneColName 0.5_0_0_0.5_x \
    --output example/first_example_1d_3gauss_1M \
    --nnorm 3 --nsampMCMC 1000000 \
    --figure example/first_example_1d_3gauss_1M.png \
    --title "first gene 3 gauss 1M" \
    --logevidence example/first_example_1d_3gauss_1M_logevid.txt
```

It converged:



Automatic rerun when Neff is too small

While some models converge even with a small number of samples, some other needs a lot of sample to reach acceptable covergence. A way to automatically rerun the MCMC when the effectif number of samples is too low is to use the `--minNeff`. The MCMC will be rerun with 10 times more sample until it reaches the value. This can potentially take forever as some model may never converge. But can be useful in other cases. Even with this option, we highly encourage the users to manually check the QC.

Compare models

In order to compare models, we will use the values of logevidence.

model	log evidence
1gauss	-4233.7
2gauss	-4221.8
3gauss	-4223.0

We can see that the model with the highest log evidence is the model with 2 gaussians. However, we see that the model with 3 gaussians is very close. When you compare models, what is important is the difference between the log evidence, not its absolute value.

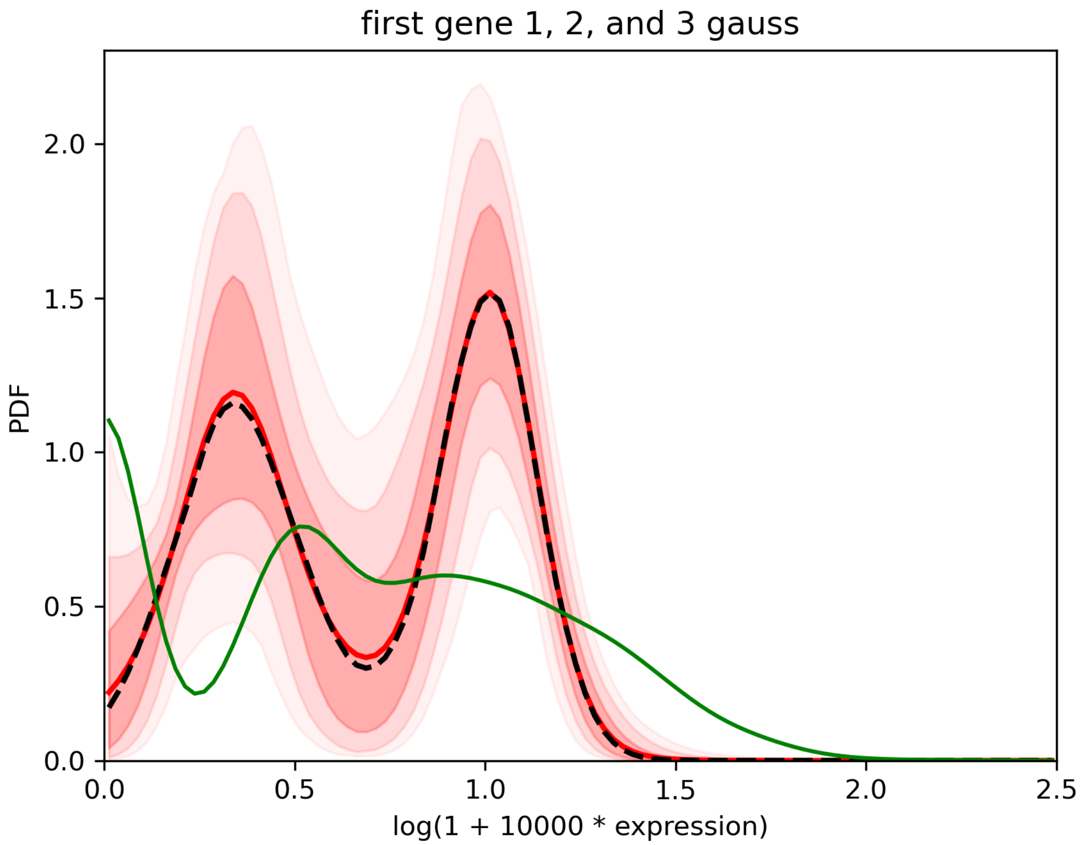
We can either choose the best model or decide to combine them:

Combine models

Another way to use these models is to use samples from all models but using the log evidence to put weight on the number of sample to use from each model.

```
$ combineMultipleModels_1d \
    --input example/nih3t3_generated_2d_2.txt \
    --geneColName 0.5_0_0_0.5_x \
    --outputs example/first_example_1d_1gauss \
    example/first_example_1d_2gauss \
    example/first_example_1d_3gauss_1M \
    --figure example/first_example_1d_1-3gauss.png \
    --title "first gene 1, 2, and 3 gauss"
```

In the standard output you will see that it only integrates samples from the 2gauss and 3gauss. Here is the result:



1.4.2 Run baredSC_2d with default parameters

Inputs

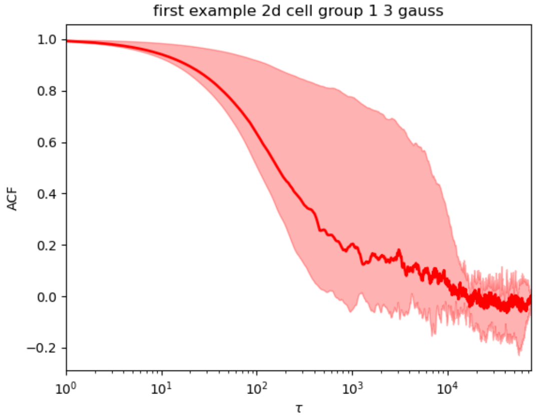
We took total UMI counts from a real dataset of NIH3T3. We generated a example where 2 genes have the same distribution (2 gaussians, one of mean 0.375, scale 0.125 and another one of mean 1 and scale 0.1). For each gene, half of cells goes in each gaussian. The genes are called "0.5_0_0_0.5_x" and "0.5_0_0_0.5_y". The input table can be downloaded from here.

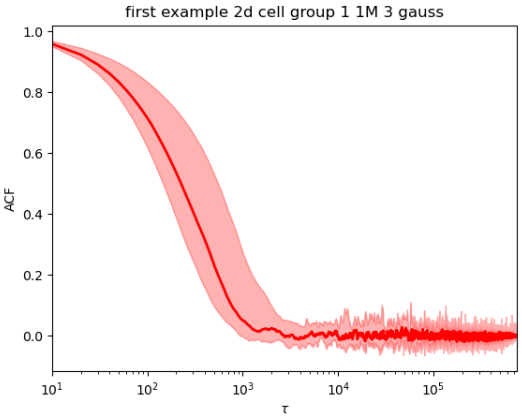
2d

As for the 1d, you need to run it with different number of gaussian 2d to find the best model or to mix them. The 2d tool is much slower than the 1d. The time depends on the `--nx`, `--ny`, `--osampxpdf`, `--osampypdf` parameters and the number of cells. To make the example quicker we will run only on the 300 random cells (group1).

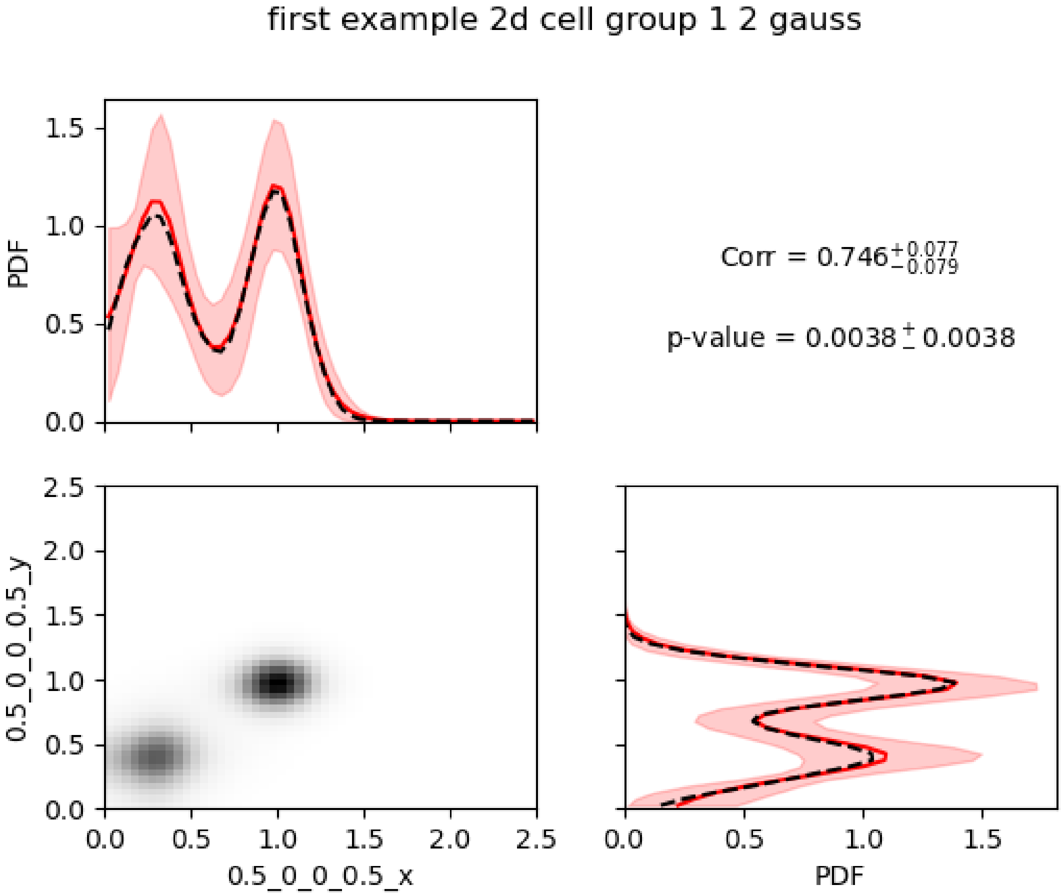
```
$ for nnorm in 1 2 3; do
    baredSC_2d \
      --input example/nih3t3_generated_2d_2.txt \
      --geneXColName 0.5_0_0_0.5_x \
      --geneYColName 0.5_0_0_0.5_y \
      --metadata1ColName group \
      --metadata1Values group1 \
      --output example/first_example_2d_cellgroup1_${nnorm}gauss \
      --nnorm ${nnorm} \
      --figure example/first_example_2d_cellgroup1_${nnorm}gauss.png \
      --title "first example 2d cell group 1 ${nnorm} gauss" \
      --logevidence example/first_example_cellgroup1_2d_${nnorm}gauss_logevid.txt
  done
```

The QC are done the same way as the 1d. The models with 1 and 2 gaussians are converging. The model with 3 gaussian is not converging (picture below left). When we rerun it with 1 million samples, it now converges (picture below right).





The best model (using the log evidence) is the 2 gaussians model.



We see a very high correlation highly significant. Here, we would like to warn the users that the correlation calculated here is a Pearson correlation, so it reflects how much the data are close to a line with positive or negative slope.

In order to appreciate the confidence interval it can be useful to split the 2d pdf in 2 parts: one above a threshold for y and one below the same threshold. This is for this purpose that we can use `--splity`. For the demo we will try different values:

```
$ baredSC_2d \
```

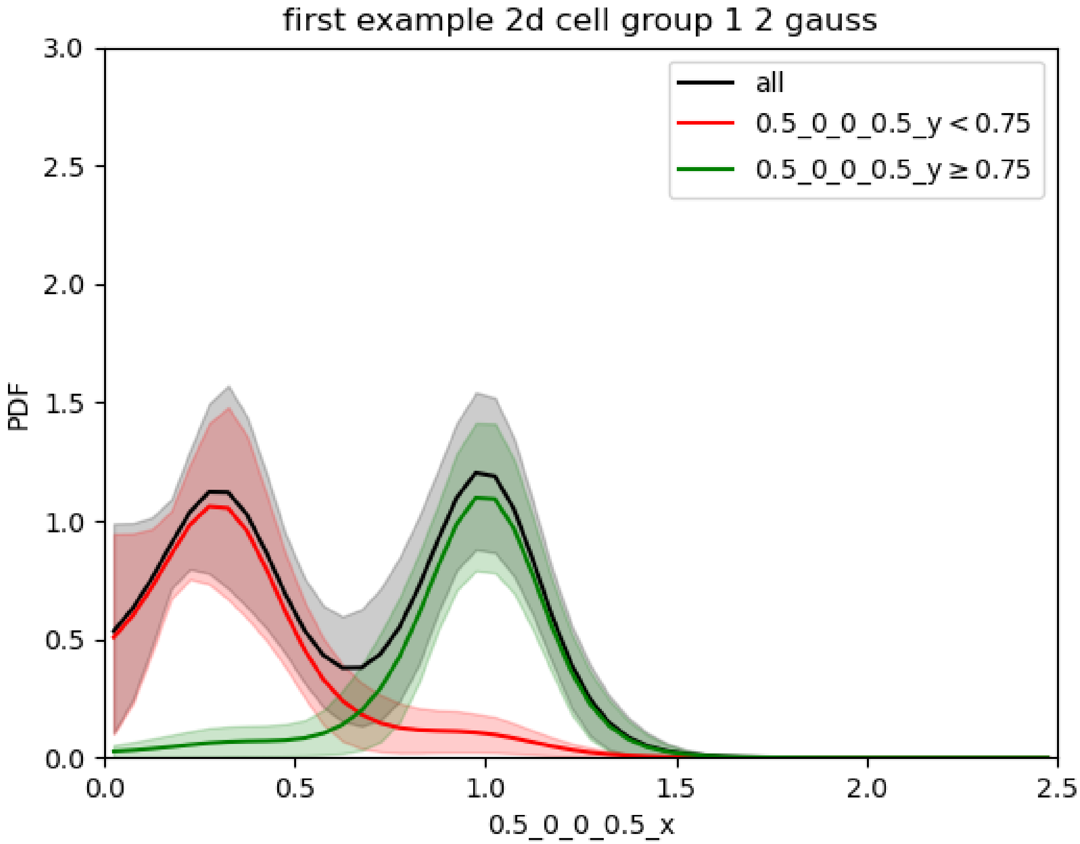
(continues on next page)

(continued from previous page)

```
    --input example/nih3t3_generated_2d_2.txt \
    --geneXColName 0.5_0_0_0.5_x \
    --geneYColName 0.5_0_0_0.5_y \
    --metadata1ColName group \
    --metadata1Values group1 \
    --output example/first_example_2d_cellgroup1_2gauss \
    --nnorm 2 \
    --figure example/first_example_2d_cellgroup1_2gauss.png \
    --title "first example 2d cell group 1 2 gauss" \
    --splity 0.35 0.75
```

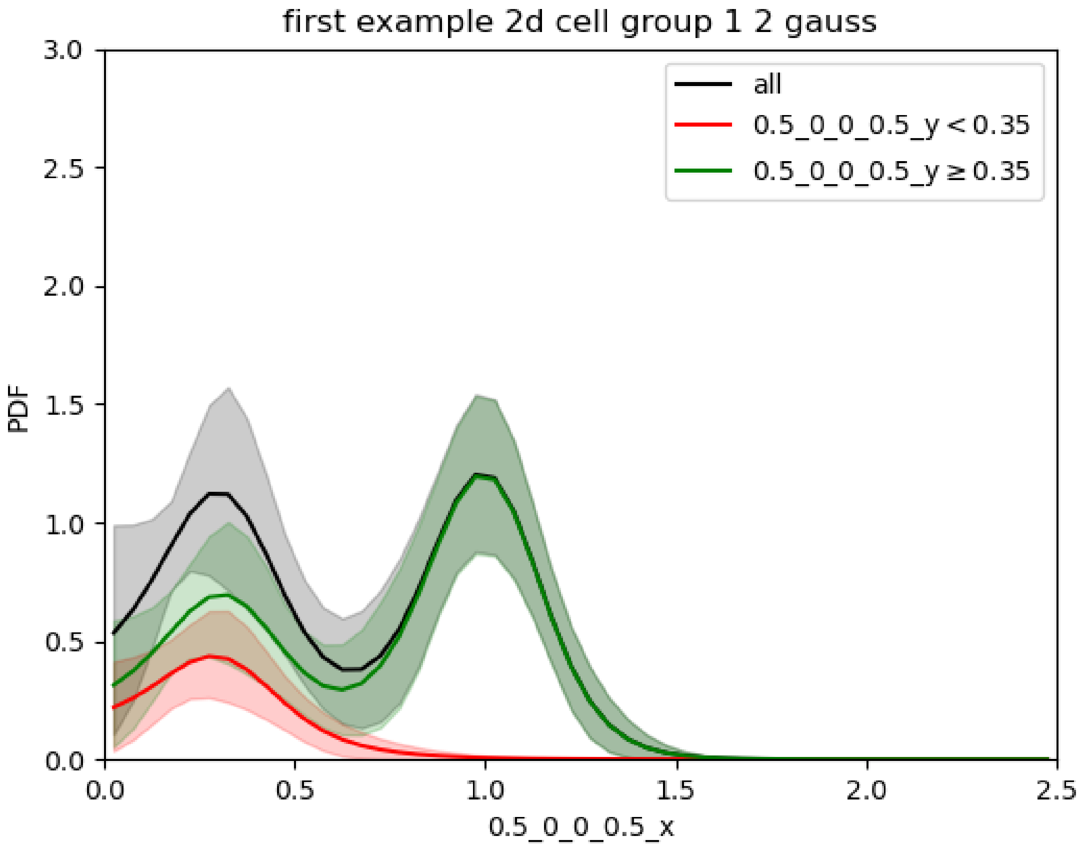
As the MCMC was run previously, it will use the `.npz` output to generate the figures, thus this operation is really quick.

When we split at 0.75 (between the 2 gaussian):

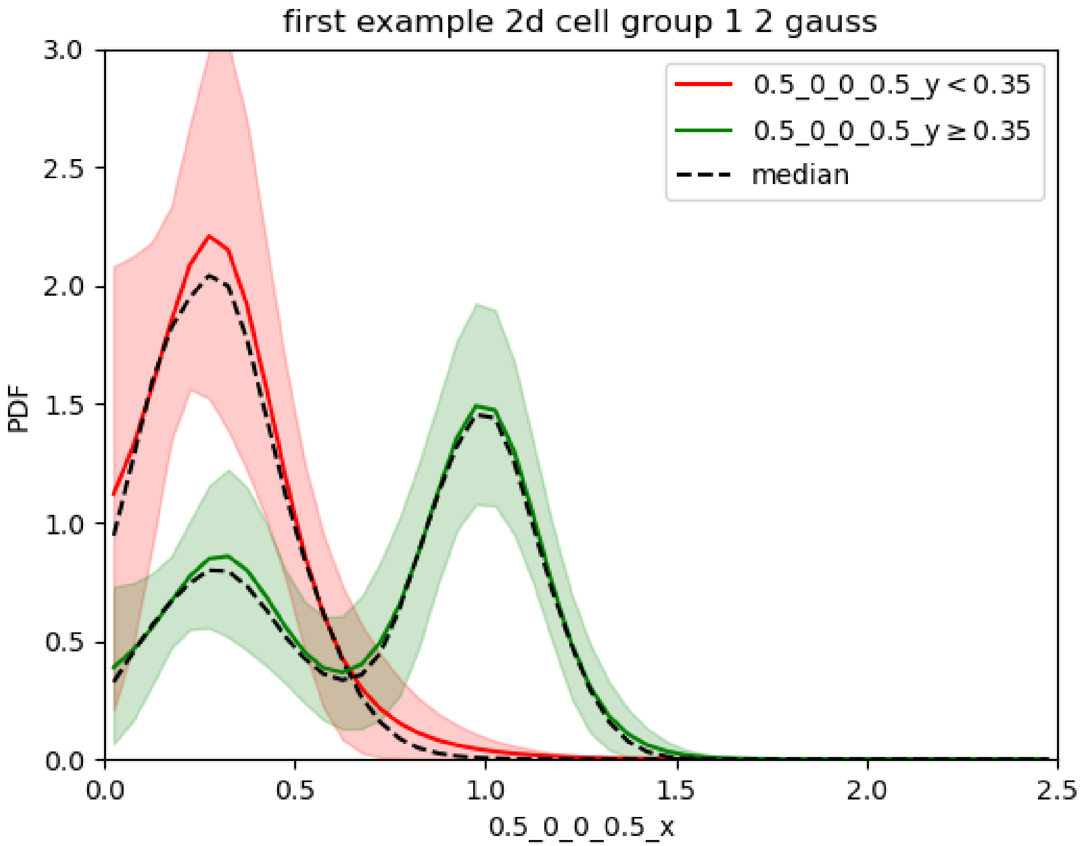


We find each of the 2 gaussians in 1d and the confidence interval is quite small.

When we split in the low gaussian (0.35):



We see that the green curve is made of 2 gaussian. The sum of both the green and red curves is the black one. This can make the comparison difficult. So the output `renorm.extension` is sometimes better.



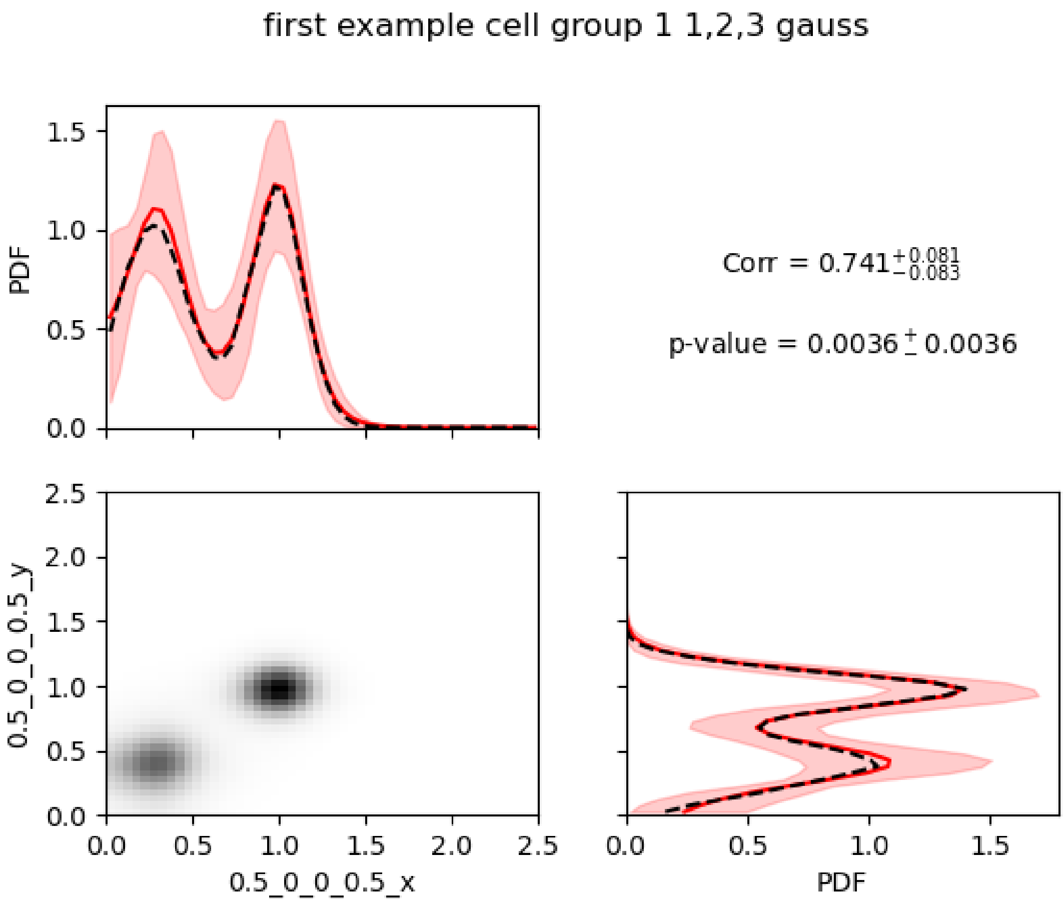
Now we clearly see that in the cells with low expression of gene y all cells are low for gene x while for cells with relatively high expression of gene y, gene x is bimodal with a greater proportion in the second gaussian.

Similarly to the 1d, the option `--minNeff` is also implemented.

You can combine multiple models with `combineMultipleModels_2d`. By default, no p-value will be evaluated for the correlation but you can use less samples to get a p-value with `--getPVal`.

```
$ combineMultipleModels_2d \
    --input example/nih3t3_generated_2d_2.txt \
    --geneXColName 0.5_0_0_0.5_x \
    --geneYColName 0.5_0_0_0.5_y \
    --metadata1ColName group \
    --metadata1Values group1 \
    --outputs example/first_example_2d_cellgroup1_1gauss \
    example/first_example_2d_cellgroup1_2gauss \
    example/first_example_2d_cellgroup1_1M_3gauss \
    --figure example/first_example_2d_cellgroup1_1-3gauss.png \
    --getPVal \
    --title "first example cell group 1 1,2,3 gauss"
```

The lines printed indicates that it uses only 282 independent samples (1 from the 1 Gaussian model, 264 from the 2 Gaussians model and 15 from the 3 Gaussians model).



1.4.3 Customize your baredSC plots

When `--figure` is used. baredSC generates plots and output text tables. To use baredSC results in your publication, you may want to customize your plots. Here is an example of python script you can use to customize your plots.

baredSC_1d

We assume you ran the *Run baredSC_1d with default parameters*.

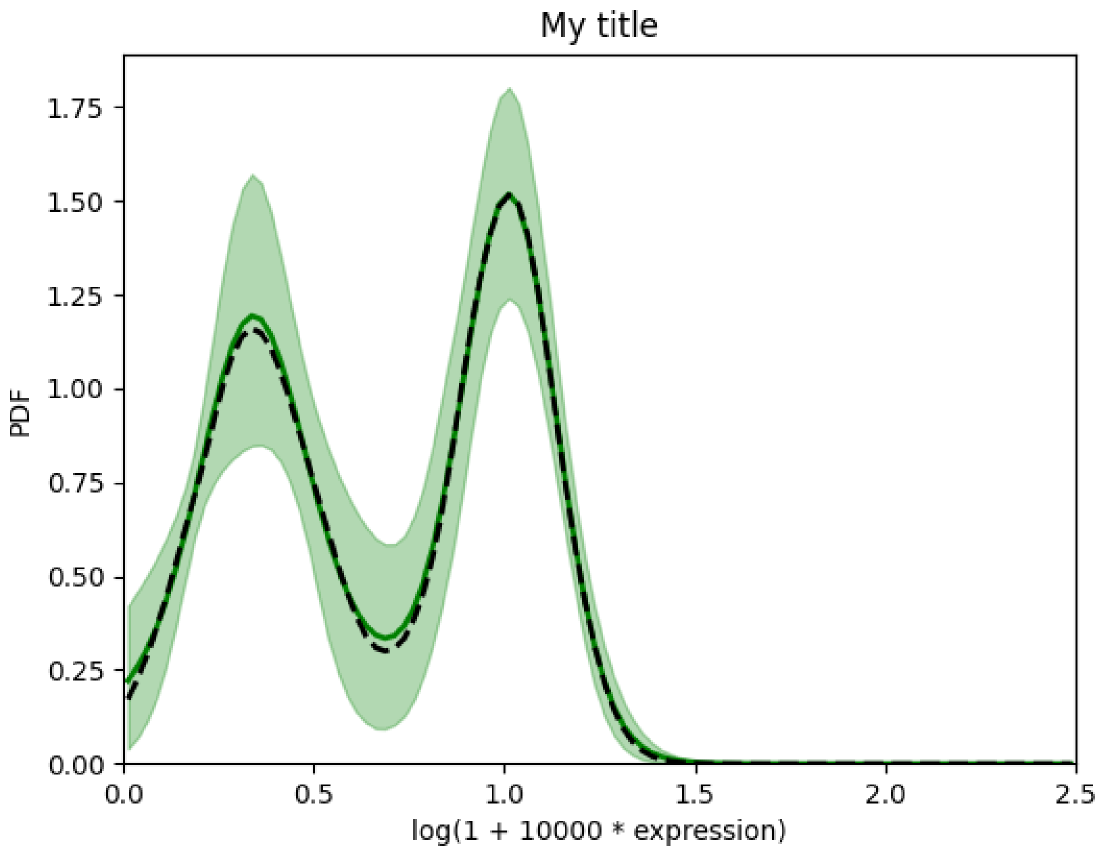
We will use matplotlib to display the pdf with the error bar:

```
In [1]: import numpy as np
   ...: import pandas as pd
   ...: import matplotlib.pyplot as plt
   ...: # Get the pdf
   ...: pdf = pd.read_csv("../example/first_example_1d_1-3gauss_pdf.txt",
   ...:                   sep="\t")
```

(continues on next page)

(continued from previous page)

```
   ...: x = pdf['x'].to_numpy()
   ...: # We assume x is equally spaced
   ...: dx = x[1] - x[0]
   ...: xmin = x[0] - dx / 2
   ...: xmax = x[-1] + dx / 2
   ...: # Plot PDF with 1 sigma
   ...: plt.figure()
   ...: plt.fill_between(x, pdf['low'].to_numpy(), pdf['high'].to_numpy(),
   ...:                  color='g', alpha=0.3, rasterized=True)
   ...: # Mean
   ...: plt.plot(x, pdf['mean'].to_numpy(), 'g', lw=2, rasterized=True)
   ...: plt.plot(x, pdf['median'].to_numpy(), 'k--', lw=2, rasterized=True)
   ...: plt.xlim(xmin, xmax)
   ...: plt.ylim(0, )
   ...: plt.xlabel('log(1 + 10000 * expression)')
   ...: plt.ylabel('PDF')
   ...: plt.title('My title')
   ...:
Out[1]: Text(0.5, 1.0, 'My title')
```

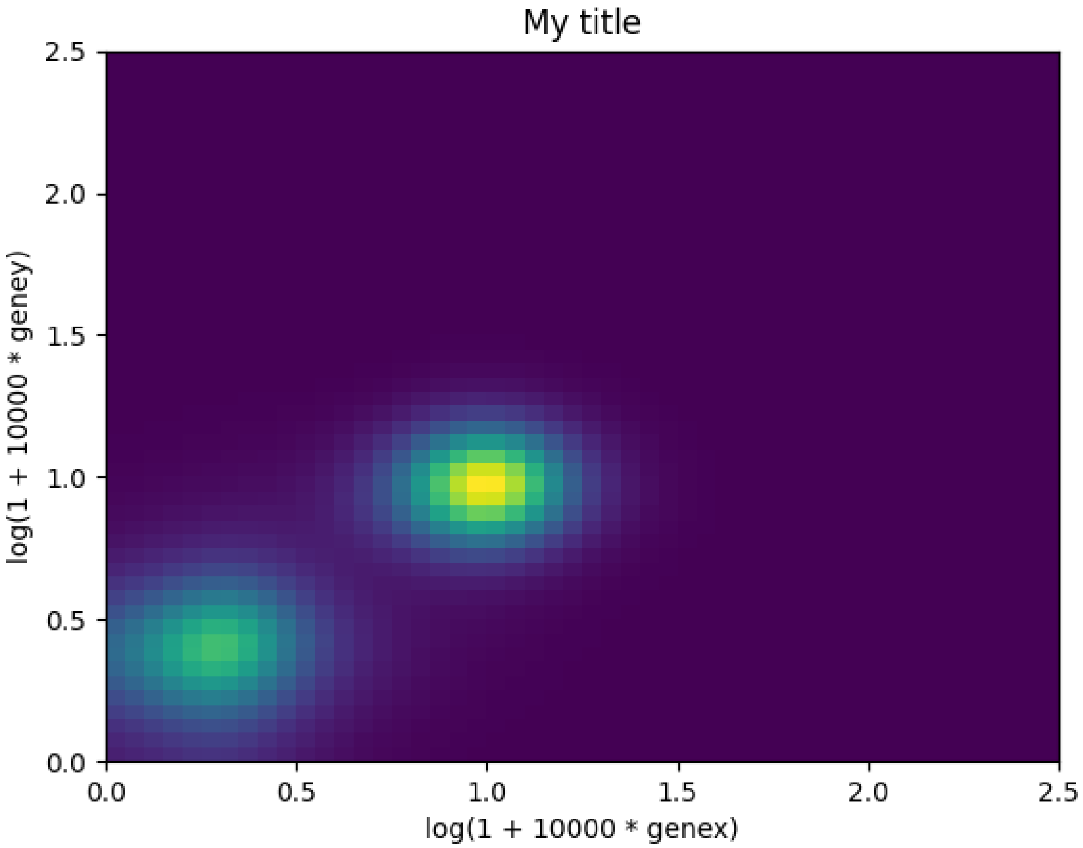


baredSC_2d

We assume you ran the *Run baredSC_2d with default parameters*.

We will use matplotlib to display the mean pdf in 2d:

```
In [2]: import numpy as np
   ...: import matplotlib.pyplot as plt
   ...: # Get the pdf
   ...: pdf = np.loadtxt("../example/first_example_2d_cellgroup1_1-3gauss_pdf2d.txt",
   ...:                  delimiter="\t", dtype='S30')
   ...: x = pdf[0, 1:].astype('float')
   ...: y = pdf[1:, 0].astype('float')
   ...: # We assume x and y is equally spaced
   ...: dx = x[1] - x[0]
   ...: xmin = x[0] - dx / 2
   ...: xmax = x[-1] + dx / 2
   ...: x_borders = np.linspace(xmin, xmax, len(x) + 1)
   ...: dy = y[1] - y[0]
   ...: ymin = y[0] - dy / 2
   ...: ymax = y[-1] + dy / 2
   ...: y_borders = np.linspace(ymin, ymax, len(y) + 1)
   ...: plt.figure()
   ...: plt.pcolormesh(x_borders, y_borders, pdf[1:,1:].astype('float'),
   ...:                shading='flat', rasterized=True, cmap='viridis')
   ...: plt.xlabel('log(1 + 10000 * genex)')
   ...: plt.ylabel('log(1 + 10000 * geney)')
   ...: plt.title('My title')
   ...: 
Out[2]: Text(0.5, 1.0, 'My title')
```



1.4.4 Compare means from baredSC results

baredSC outputs can be used to evaluate the fold-change of expression between 2 conditions.

Inputs

We use the same inputs as in *previous input descriptions*.

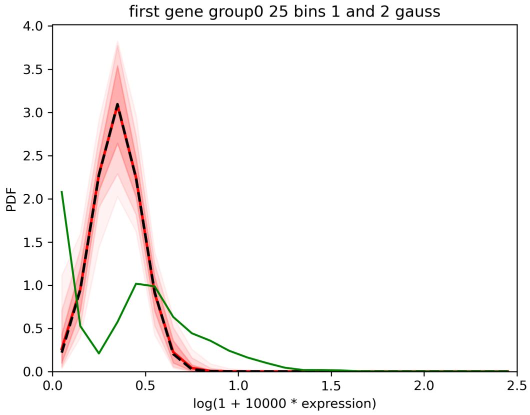
Run baredSC on each subpopulation

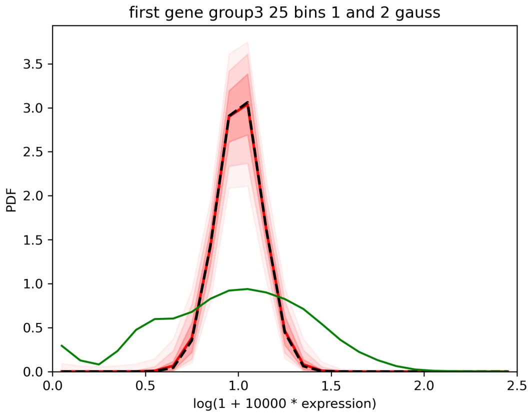
Let's focus on cells of group 0.0 and group 3.0 (they correspond to each of the 2 gaussians found previously). To increase the speed we will use less bins (`--nx 25`):

```
$ for group in 0 3; do
    for nnorm in 1 2; do
      baredSC_1d \
        --input example/nih3t3_generated_2d_2.txt \
        --metadata1ColName 0.5_0_0_0.5_group \
        --metadata1Values ${group}.0 \
        --geneColName 0.5_0_0_0.5_x \
        --output example/first_example_1d_group${group}_${nnorm}gauss_25_neff200 \
        --nnorm ${nnorm} --nx 25 --minNeff 200 \
        --figure example/first_example_1d_group${group}_${nnorm}gauss_25_neff200.png \
        --title "first gene ${nnorm} gauss group${group} 25 bins neff200" \
        --logevidence example/first_example_1d_group${group}_${nnorm}gauss_25_neff200_
↪logevid.txt
    done
    combineMultipleModels_1d \
      --input example/nih3t3_generated_2d_2.txt \
      --metadata1ColName 0.5_0_0_0.5_group \
      --metadata1Values ${group}.0 \
      --geneColName 0.5_0_0_0.5_x --nx 25 \
      --outputs example/first_example_1d_group${group}_1gauss_25_neff200 \
      example/first_example_1d_group${group}_2gauss_25_neff200 \
      --figure example/first_example_1d_group${group}_1-2gauss_25.png \
      --title "first gene group$group 25 bins 1 and 2 gauss"
  done
```

In one case, 100 000 samples were not sufficient.

We check the QC. We can now compare the results:



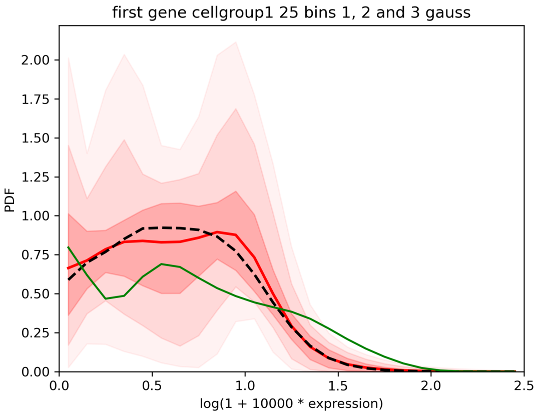


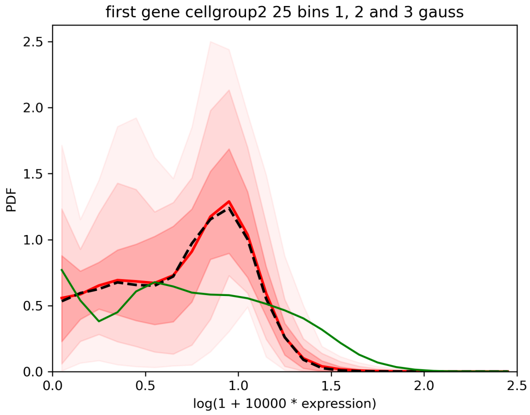
We can see that the tool fits relatively nicely the gaussians which were in inputs.

If we use another metadata which is just 300 and 500 random cells:

```
$ for group in 1 2; do
    for nnorm in 1 2 3; do
      baredSC_1d \
        --input example/nih3t3_generated_2d_2.txt \
        --metadata1ColName group \
        --metadata1Values group${group} \
        --geneColName 0.5_0_0_0.5_x \
        --output example/first_example_1d_cellgroup${group}_${nnorm}gauss_25_neff200 \
        --nnorm ${nnorm} --nx 25 --minNeff 200 \
        --figure example/first_example_1d_cellgroup${group}_${nnorm}gauss_25_neff200.png \
        --title "first gene ${nnorm} gauss cellgroup${group}" \
        --logevidence example/first_example_1d_cellgroup${group}_${nnorm}gauss_25_neff200_logevid.txt
    done
    combineMultipleModels_1d \
      --input example/nih3t3_generated_2d_2.txt \
      --metadata1ColName group \
      --metadata1Values group${group} \
      --geneColName 0.5_0_0_0.5_x --nx 25 \
      --outputs example/first_example_1d_cellgroup${group}_1gauss_25_neff200 \
      example/first_example_1d_cellgroup${group}_2gauss_25_neff200 \
      example/first_example_1d_cellgroup${group}_3gauss_25_neff200 \
      --figure example/first_example_1d_cellgroup${group}_1-3gauss_25.png \
      --title "first gene cellgroup$group 25 bins 1, 2 and 3 gauss"
  done
```

We see that the results are different in both groups but in both cases the confidence interval is quite large:





One of the output is *_means.txt.gz*. Each line correspond to the value of the mean expression evaluated at each sample of the MCMC. It can be used to estimate:

- A confidence interval on the mean value (in the used axis log(1 + 10^4 * expression))
- A confidence interval on the fold change (delogged).

```
In [1]: import numpy as np
   ...: def delog(x):
   ...:     return(1e-4 * (np.exp(x) - 1))
   ...: my_quantiles = [0.5 - 0.6827 / 2, 0.5, 0.5 + 0.6827 / 2]
   ...: # Get data
   ...: means1 = np.genfromtxt('../example/first_example_1d_group0_1-2gauss_25_means.txt.
↪gz')
   ...: print(f'Values of mean in group0: {means1}')
   ...: means2 = np.genfromtxt('../example/first_example_1d_group3_1-2gauss_25_means.txt.
↪gz')
   ...: print(f'Values of mean in group3: {means2}')
   ...: # Shuffle means2
   ...: np.random.shuffle(means2)
   ...: print(f'Mean log expression in group0: {np.quantile(means1, my_quantiles)}')
   ...: print(f'Mean log expression in group3: {np.quantile(means2, my_quantiles)}')
   ...: fc = [delog(x1) / delog(x2) for x1, x2 in zip(means1, means2)]
   ...: print(f'Estimation of fold-change: {np.quantile(fc,  my_quantiles)}')
   ...:
```

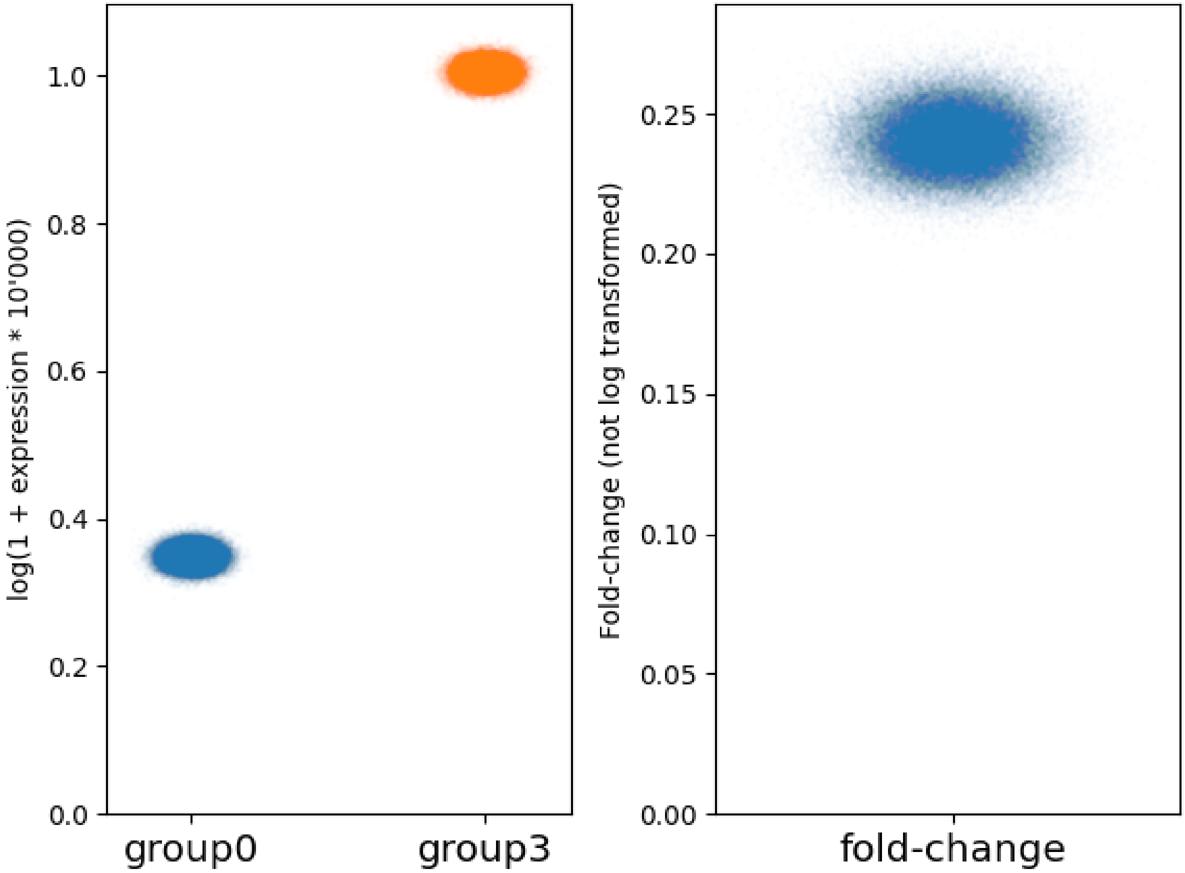
(continues on next page)

(continued from previous page)

```
Values of mean in group0: [0.35776993 0.35776993 0.35776993 ... 0.34672964 0.34672964 0.
↪38877004]
Values of mean in group3: [1.00746002 1.02085388 1.02085388 ... 1.00369734 1.01363153 1.
↪02256951]
Mean log expression in group0: [0.33838793 0.34953488 0.36060358]
Mean log expression in group3: [0.99479228 1.00628287 1.01751424]
Estimation of fold-change: [0.23110761 0.24112418 0.25130651]
```

We can use matplotlib to display the results graphically:

```
In [2]: import matplotlib.pyplot as plt
   ...: x1, x2 = np.random.normal(1, 0.1, len(means1)), np.random.normal(3, 0.1,␣
↪len(means2))
   ...: fig, axs = plt.subplots(1, 2)
   ...: axs[0].scatter(x1, means1, s=1, alpha=0.01)
   ...: axs[0].scatter(x2, means2, s=1, alpha=0.01)
   ...: axs[0].set_ylim(0, )
   ...: axs[0].set_ylabel('log(1 + expression * 10\'000)')
   ...: axs[0].set_xticks([1, 3])
   ...: axs[0].set_xticklabels(['group0', 'group3'], fontsize='x-large')
   ...: axs[1].scatter(x1[:len(fc)], fc, s=1, alpha=0.01)
   ...: axs[1].set_xticks([1])
   ...: axs[1].set_xticklabels(['fold-change'], fontsize='x-large')
   ...: axs[1].set_ylim(0, )
   ...: axs[1].set_ylabel('Fold-change (not log transformed)')
   ...: fig.tight_layout()
   ...:
```



On the first example with group0 and group3 where each is a different gaussian, we see a mean log expression around the expected 0.375 value in group 0, 1 in group 3. We see that the fold-change is around 25% (this is the real fold-change, not log transformed).

Now if we have a look to the subsamples of cells (300 and 500 cells) and perform the same analysis:

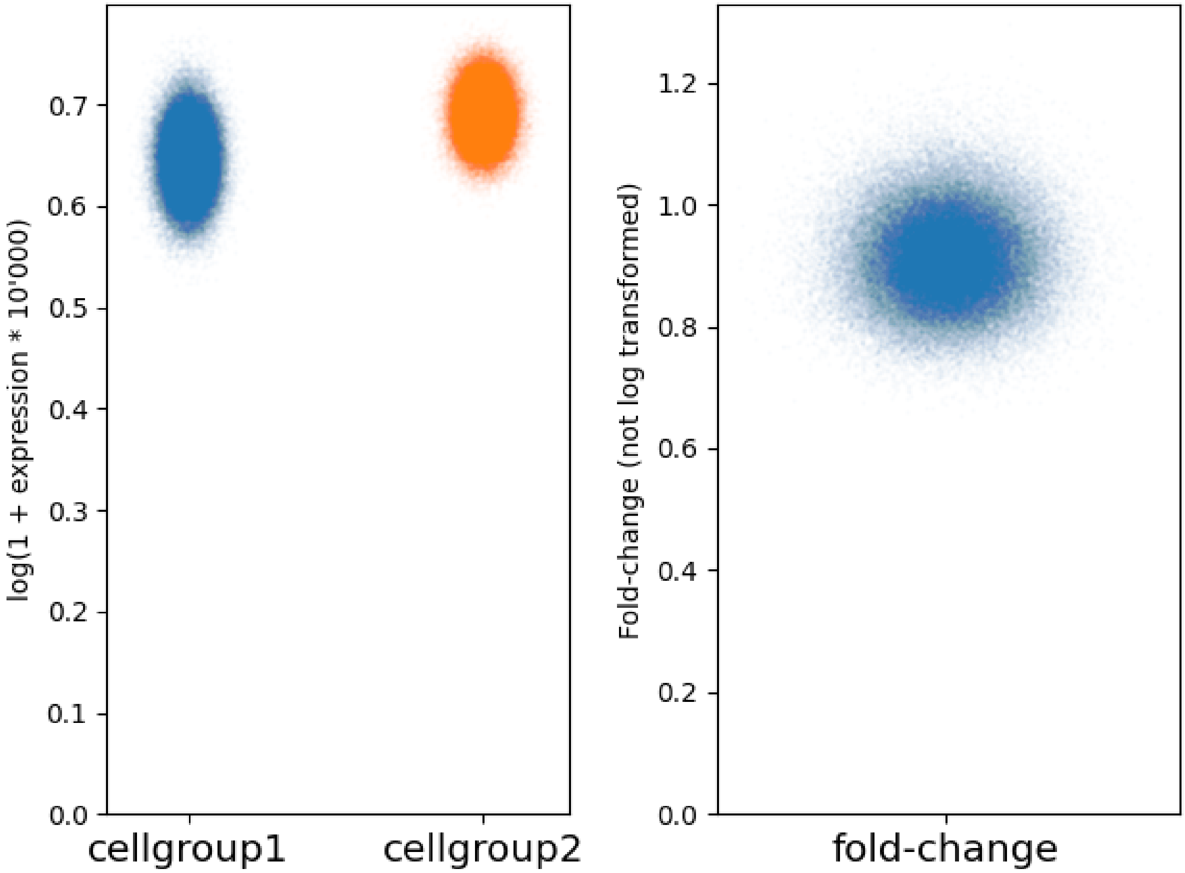
```
In [3]: # Get data
   ...: means1 = np.genfromtxt('../example/first_example_1d_cellgroup1_1-3gauss_25_means.
↪txt.gz')
   ...: print(f'Values of mean in cellgroup1: {means1}')
   ...: means2 = np.genfromtxt('../example/first_example_1d_cellgroup2_1-3gauss_25_means.
↪txt.gz')
   ...: print(f'Values of mean in cellgroup2: {means2}')
   ...: # Shuffle means2
   ...: np.random.shuffle(means2)
   ...: print(f'Mean log expression in cellgroup1: {np.quantile(means1, my_quantiles)}')
   ...: print(f'Mean log expression in cellgroup2: {np.quantile(means2, my_quantiles)}')
   ...: fc = [delog(x1) / delog(x2) for x1, x2 in zip(means1, means2)]
   ...: print(f'Estimation of fold-change: {np.quantile(fc,  my_quantiles)}')
   ...:
Values of mean in cellgroup1: [0.66035127 0.66035127 0.62547552 ... 0.62783495 0.
↪62783495 0.62783495]
Values of mean in cellgroup2: [0.68203872 0.69214061 0.71198453 ... 0.70698152 0.
↪70698152 0.70698152]
```

(continues on next page)

(continued from previous page)

```
Mean log expression in cellgroup1: [0.6161945  0.64620379 0.67630369]
Mean log expression in cellgroup2: [0.66763529 0.69110072 0.71426981]
Estimation of fold-change: [0.8423754  0.91214824 0.9865969 ]
```

```
In [4]: x1, x2 = np.random.normal(1, 0.1, len(means1)), np.random.normal(3, 0.1,␣
↪len(means2))
   ...: fig, axs = plt.subplots(1, 2)
   ...: axs[0].scatter(x1, means1, s=1, alpha=0.01)
   ...: axs[0].scatter(x2, means2, s=1, alpha=0.01)
   ...: axs[0].set_ylim(0, )
   ...: axs[0].set_ylabel('log(1 + expression * 10\'000)')
   ...: axs[0].set_xticks([1, 3])
   ...: axs[0].set_xticklabels(['cellgroup1', 'cellgroup2'], fontsize='x-large')
   ...: axs[1].scatter(x1[:len(fc)], fc, s=1, alpha=0.01)
   ...: axs[1].set_xticks([1])
   ...: axs[1].set_xticklabels(['fold-change'], fontsize='x-large')
   ...: axs[1].set_ylim(0, )
   ...: axs[1].set_ylabel('Fold-change (not log transformed)')
   ...: fig.tight_layout()
   ...:
```



Here, we can see that the fold-change is slightly below 1.

1.4.5 Change minScale

Inputs

We took total UMI counts from a real dataset of NIH3T3. We generated a example where 2 genes have the same distribution (2 gaussians, one of mean 0.375, scale 0.125 and another one of mean 1 and scale 0.1). Half of cells goes in each gaussian. The gene is called “0.5_0_0_0.5_x”. The input table can be downloaded from here.

minScale

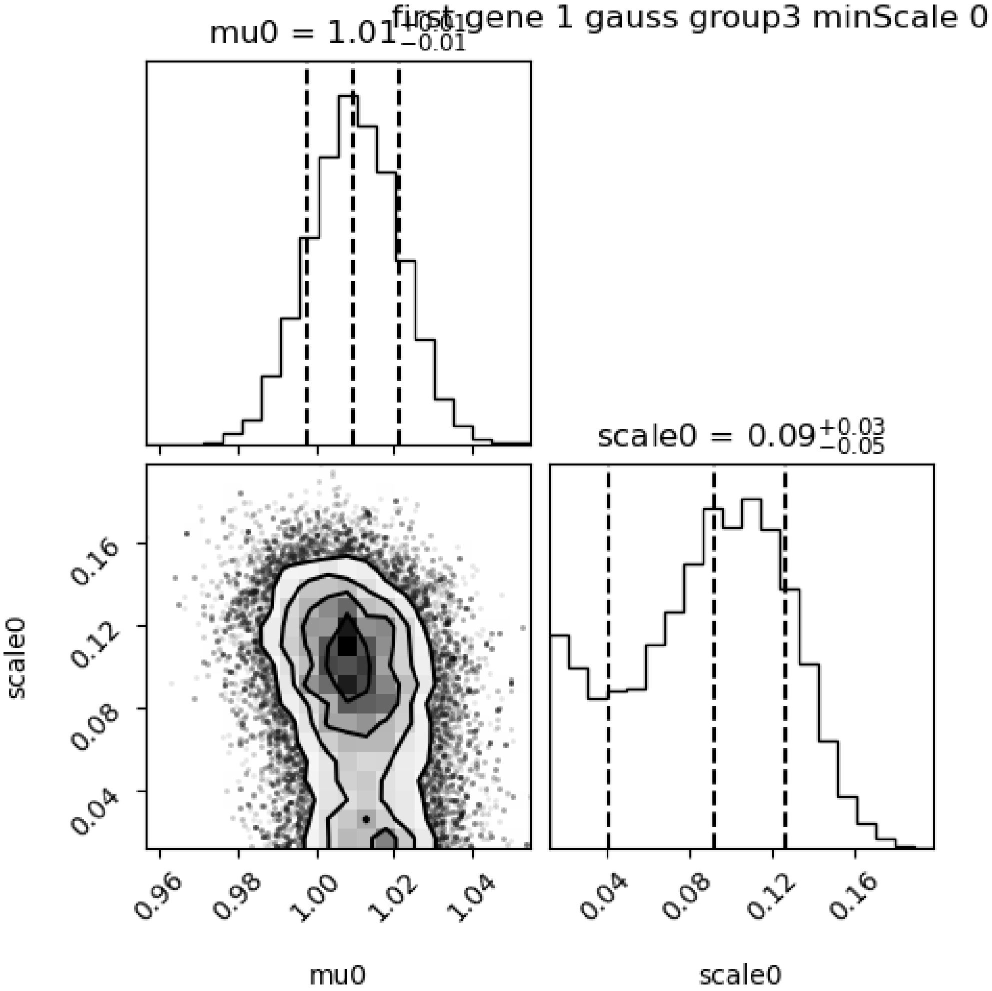
In our model, the prior on the scale of each Gaussian is that it must be above the `--minScale`. By default, this value is set to 0.1 because most of the time we don’t have the resolution to go below. Here we will decrease this value to see how it affects the results

Run baredSC on the subpopulation with a scale of 0.1

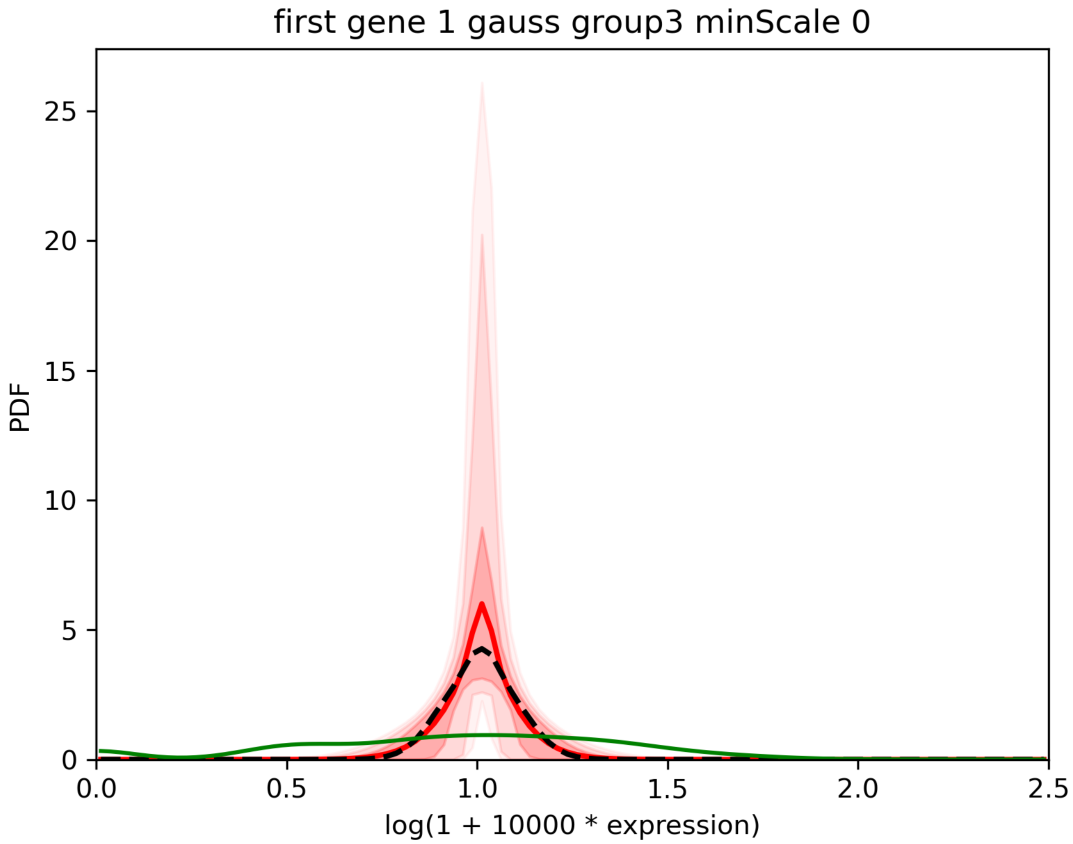
Let’s focus on cells of group 3.0 (which corresponds to the second Gaussian of mean 1 and scale 0.1). We run baredSC but we put `--minScale` to 0. A minimum scale of 0 is not accepted by baredSC because it causes some issues so setting it to 0 will put the minimum value accepted by baredSC.

```
$ nnorm=1
$ baredSC_1d \
        --input example/nih3t3_generated_2d_2.txt \
        --metadata1ColName 0.5_0_0_0.5_group \
        --metadata1Values 3.0 \
        --geneColName 0.5_0_0_0.5_x \
        --output example/first_example_1d_group3_${nnorm}gauss_ms0 \
        --nnorm ${nnorm} --minScale 0 \
        --minNeff 200 \
        --figure example/first_example_1d_group3_${nnorm}gauss_ms0.png \
        --title "first gene ${nnorm} gauss group3 minScale 0"
```

First let’s have a look to the corner plot:



We see that the median value for scale0 is 0.09 so very close to what was simulated. However, we also see some very small values of scale. As a consequence, when we look at the results:

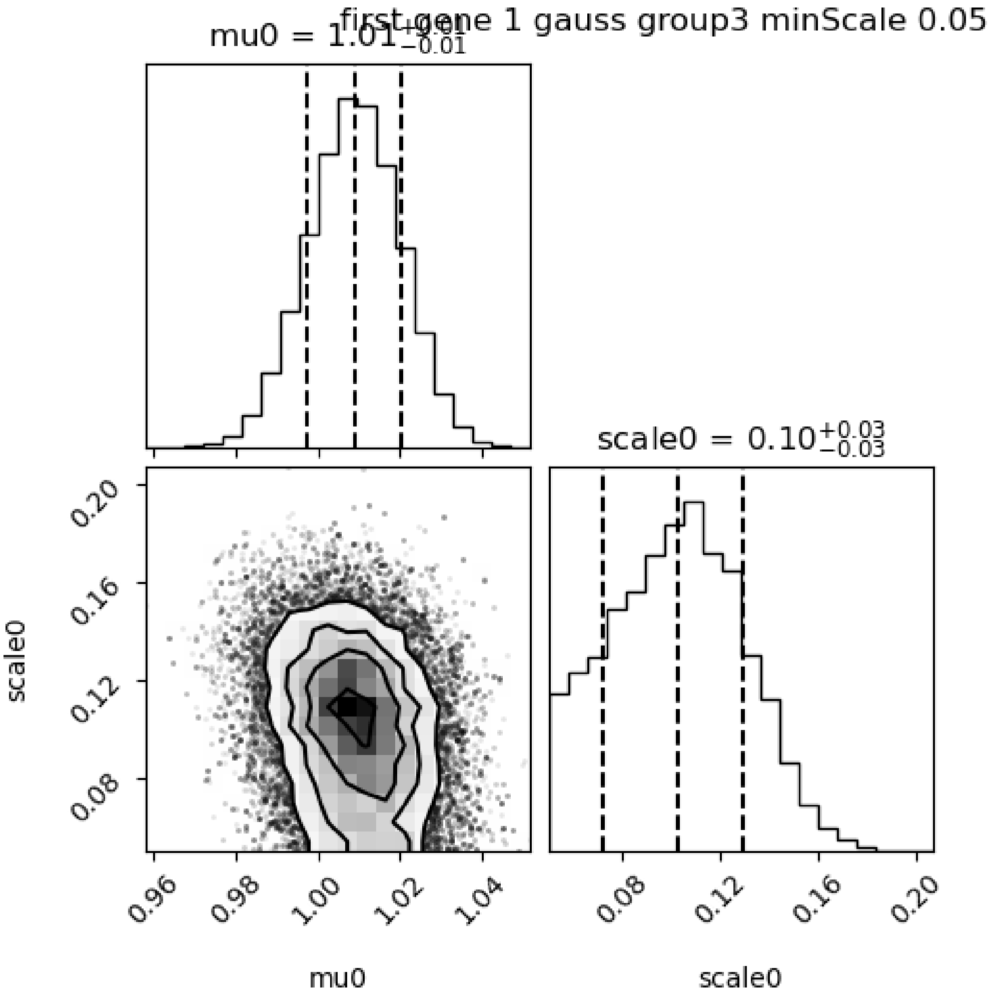


We see that the confidence interval is quite large and that the mean (red) is far from the median (dashed black).

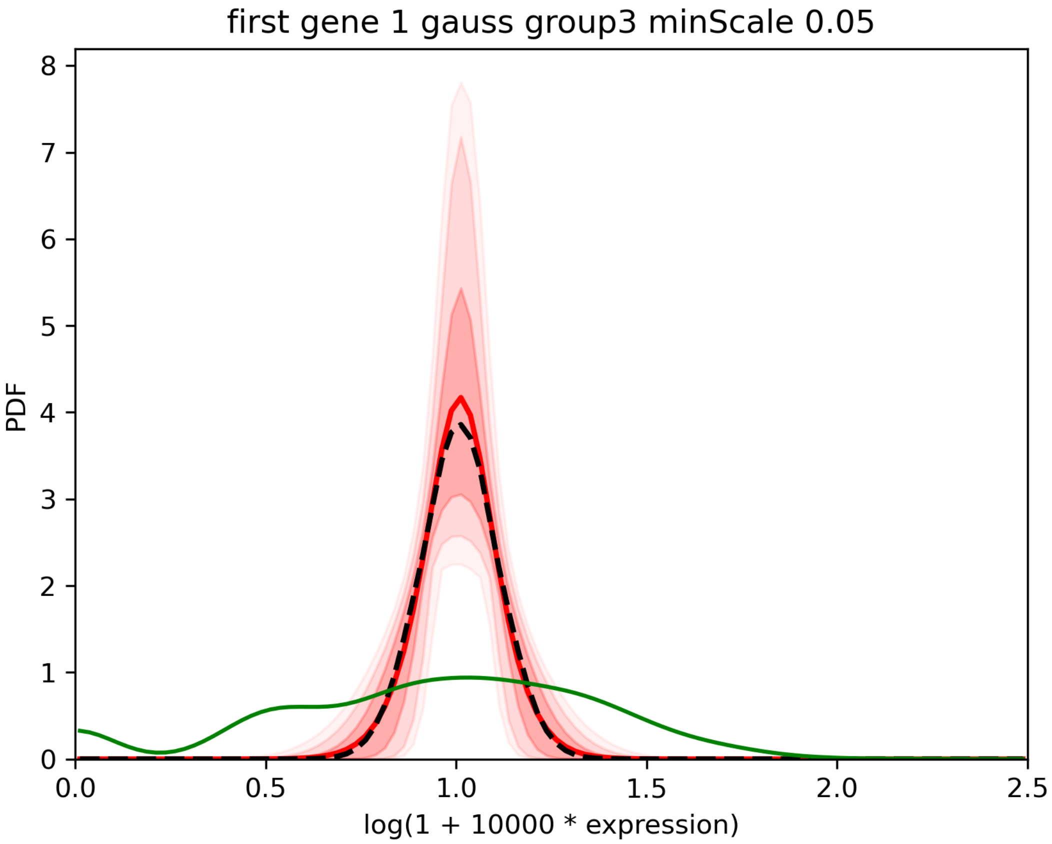
In such cases, it is reasonable to use a value for the minimum scale intermediate between the minimum value accepted by baredSC (0.0125) and the default value (0.1). For example, we can use 0.05:

```
$ nnorm=1
$ baredSC_1d \
    --input example/nih3t3_generated_2d_2.txt \
    --metadata1ColName 0.5_0_0_0.5_group \
    --metadata1Values 3.0 \
    --geneColName 0.5_0_0_0.5_x \
    --output example/first_example_1d_group3_${nnorm}gauss_ms0.05 \
    --nnorm ${nnorm} --minScale 0.05 \
    --minNeff 200 \
    --figure example/first_example_1d_group3_${nnorm}gauss_ms0.05.png \
    --title "first gene ${nnorm} gauss group3 minScale 0.05"
```

The corner plot shows that the median is still close to 0.1:



We see that the confidence interval is reduced large and that the mean (red) is closer from the median (dashed black).



1.4.6 Change the number of bins

- *Inputs*
- *Run baredSC in 2D*

baredSC can be relatively slow. A way to speed it is to decrease the number of bins.

Inputs

We took total UMI counts from a real dataset of NIH3T3. We generated a example where the PDF of the 2 genes is a 2D Gaussian. The mean on each axis and the scale on each axis is equal to 0.5 and the correlation value is also 0.5. The input table can be downloaded from here.

Run baredSC in 2D

By default baredSC_2d uses 50 bins in x and 50 bins in y. Let's run with default parameters.

```
$ nnorm=1
$ baredSC_2d \
    --input example/nih3t3_generated_second.txt \
    --geneXColName 1_0.5_0.5_0.5_0.5_0.5_x \
    --geneYColName 1_0.5_0.5_0.5_0.5_0.5_y \
    --metadata1ColName group \
    --metadata1Values group1 \
    --output example/second_example_2d_cellgroup1_${nnorm}gauss \
    --nnorm ${nnorm} \
    --figure example/second_example_2d_cellgroup1_${nnorm}gauss.png \
    --title "second example 2d cell group 1 ${nnorm} gauss"
```

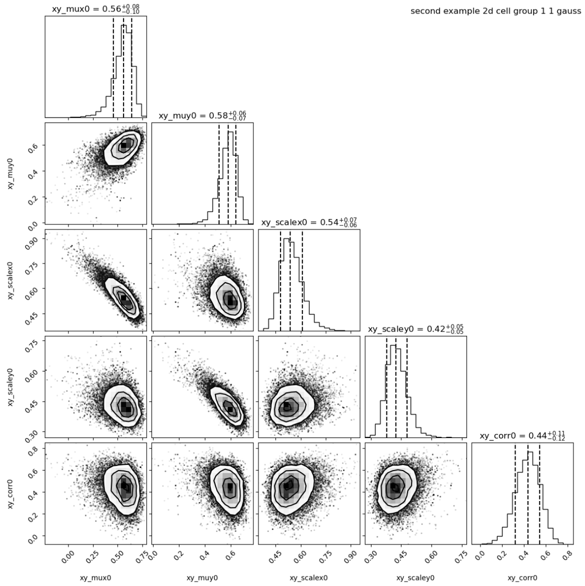
It took 11 minutes to run the MCMC and 3 minutes to compute the PDF.

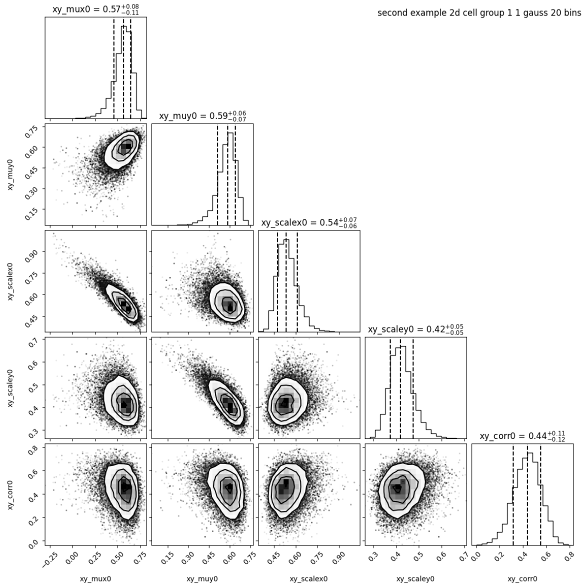
Let's try to reduce the number of bins using `--nx` and `--ny`:

```
$ nnorm=1
$ baredSC_2d \
    --input example/nih3t3_generated_second.txt \
    --geneXColName 1_0.5_0.5_0.5_0.5_0.5_x \
    --geneYColName 1_0.5_0.5_0.5_0.5_0.5_y \
    --metadata1ColName group \
    --metadata1Values group1 \
    --output example/second_example_2d_cellgroup1_${nnorm}gauss_nx20 \
    --nnorm ${nnorm} \
    --nx 20 --ny 20 \
    --figure example/second_example_2d_cellgroup1_${nnorm}gauss_nx20.png \
    --title "second example 2d cell group 1 ${nnorm} gauss 20 bins"
```

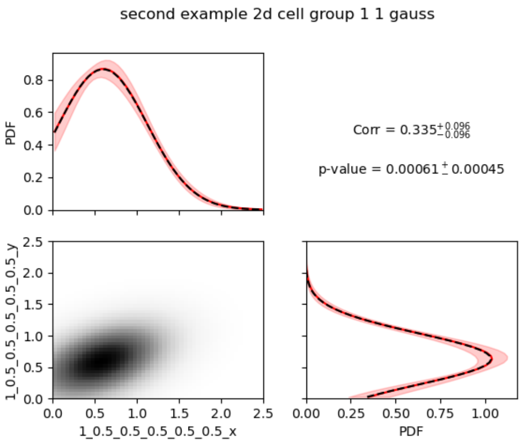
This time it took 3:32 minutes to compute MCMC and 30 seconds to get the PDF.

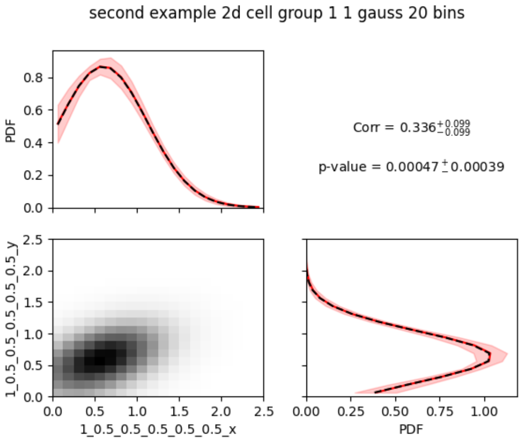
The parameters found are the same:





However, the image provided with 20 bins is much more pixelized:

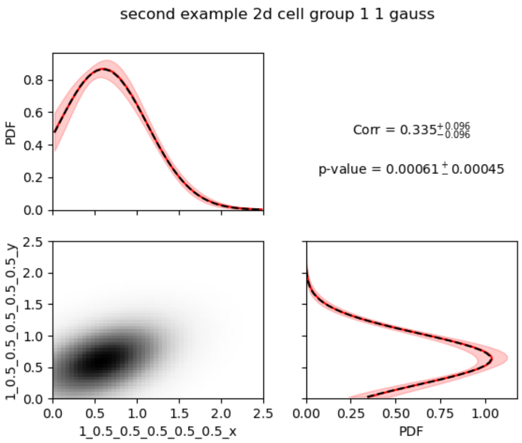


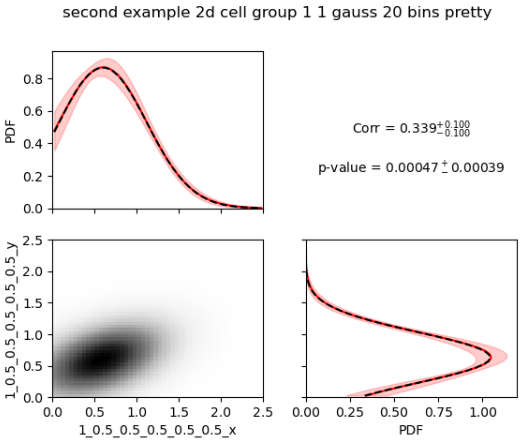


There is a way to render the plot prettier. However, you need to keep in mind that these pretty plots will not display the data as they have been used to compute the likelihood. In this example, the scale of the Gaussian is large enough that's why it gave the same results with both number of bins. We can set the number of bins to use in the plot with `--prettyBinsx` and `--prettyBinsy`.

```
$ nnorm=1
$ baredSC_2d \
    --input example/nih3t3_generated_second.txt \
    --geneXColName 1_0.5_0.5_0.5_0.5_0.5_x \
    --geneYColName 1_0.5_0.5_0.5_0.5_0.5_y \
    --metadata1ColName group \
    --metadata1Values group1 \
    --output example/second_example_2d_cellgroup1_${nnorm}gauss_nx20 \
    --nnorm ${nnorm} \
    --nx 20 --ny 20 --prettyBinsx 50 --prettyBinsy 50 \
    --figure example/second_example_2d_cellgroup1_${nnorm}gauss_nx20_pretty.png \
    --title "second example 2d cell group 1 ${nnorm} gauss 20 bins pretty"
```

It will use the `.npz` file generated with the last run to get the MCMC results.





Now they really look alike.

These options also exists in 1D.

1.4.7 Change scalePrior

baredSC_2d uses as a prior on the correlation value of each Gaussian a normal distribution. In order to reduce the number of false-positive (anti-)correlation detection. The scale of the normal distribution is set to 0.3. We show here the influence of this prior. The input table can be downloaded from here.

Inputs

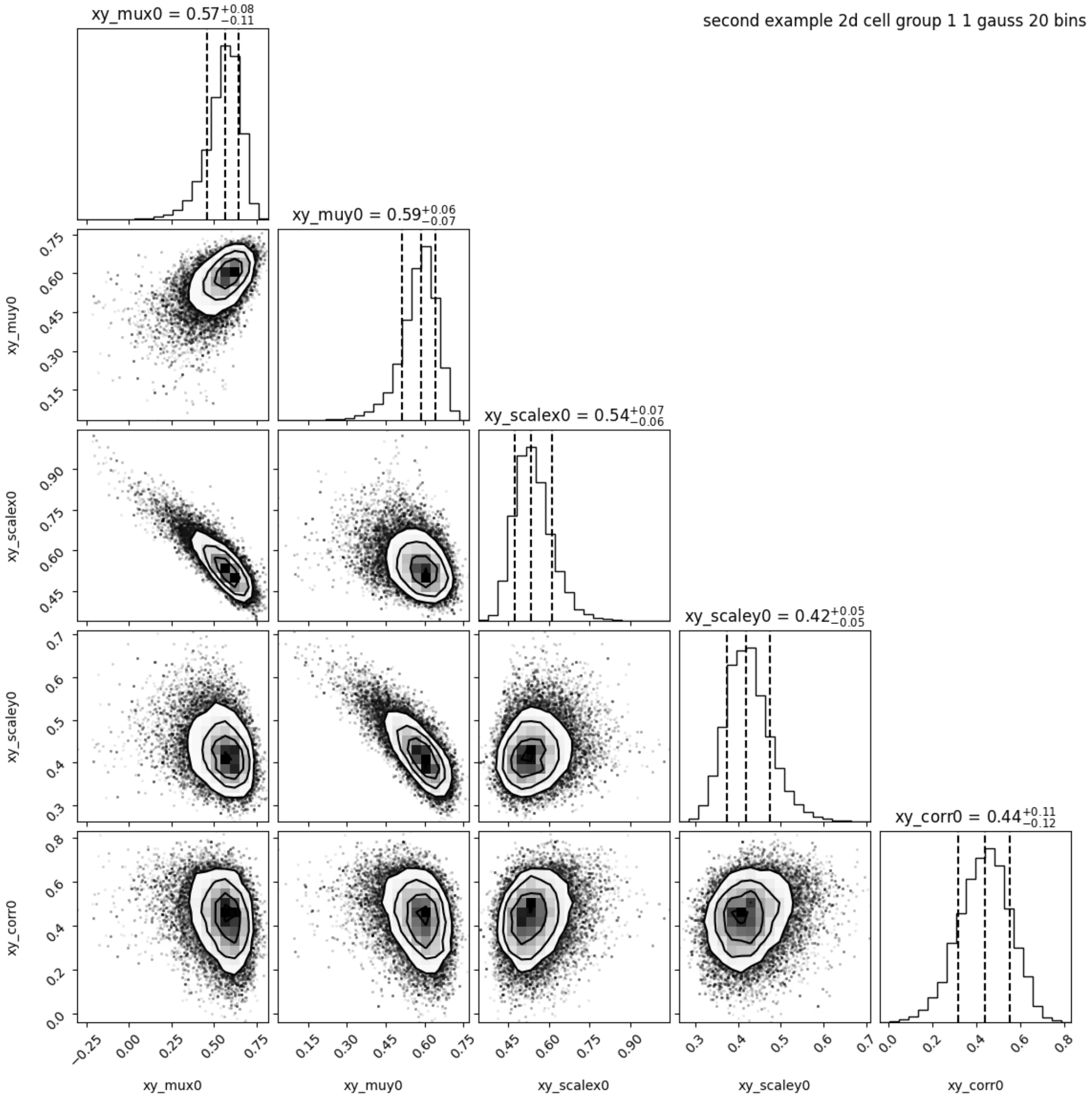
We took total UMI counts from a real dataset of NIH3T3. We generated a example where the PDF of the 2 genes is a 2D Gaussian. The mean on each axis and the scale on each axis is equal to 0.5 and the correlation value is also 0.5.

Run baredSC in 2D

By default baredSC_2d uses 50 bins in x and 50 bins in y. But to increase the speed we use only 20 bins:

```
$ nnorm=1
$ baredSC_2d \
    --input example/nih3t3_generated_second.txt \
    --geneXColName 1_0.5_0.5_0.5_0.5_0.5_x \
    --geneYColName 1_0.5_0.5_0.5_0.5_0.5_y \
    --metadata1ColName group \
    --metadata1Values group1 \
    --output example/second_example_2d_cellgroup1_${nnorm}gauss_nx20 \
    --nnorm ${nnorm} \
    --nx 20 --ny 20 \
    --figure example/second_example_2d_cellgroup1_${nnorm}gauss_nx20.png \
    --title "second example 2d cell group 1 ${nnorm} gauss 20 bins"
```

We see that the correlation found is 0.44 +/- 0.11.



Let see how this changes if we reduce the scale of the Normal distribution of the prior to 0.1

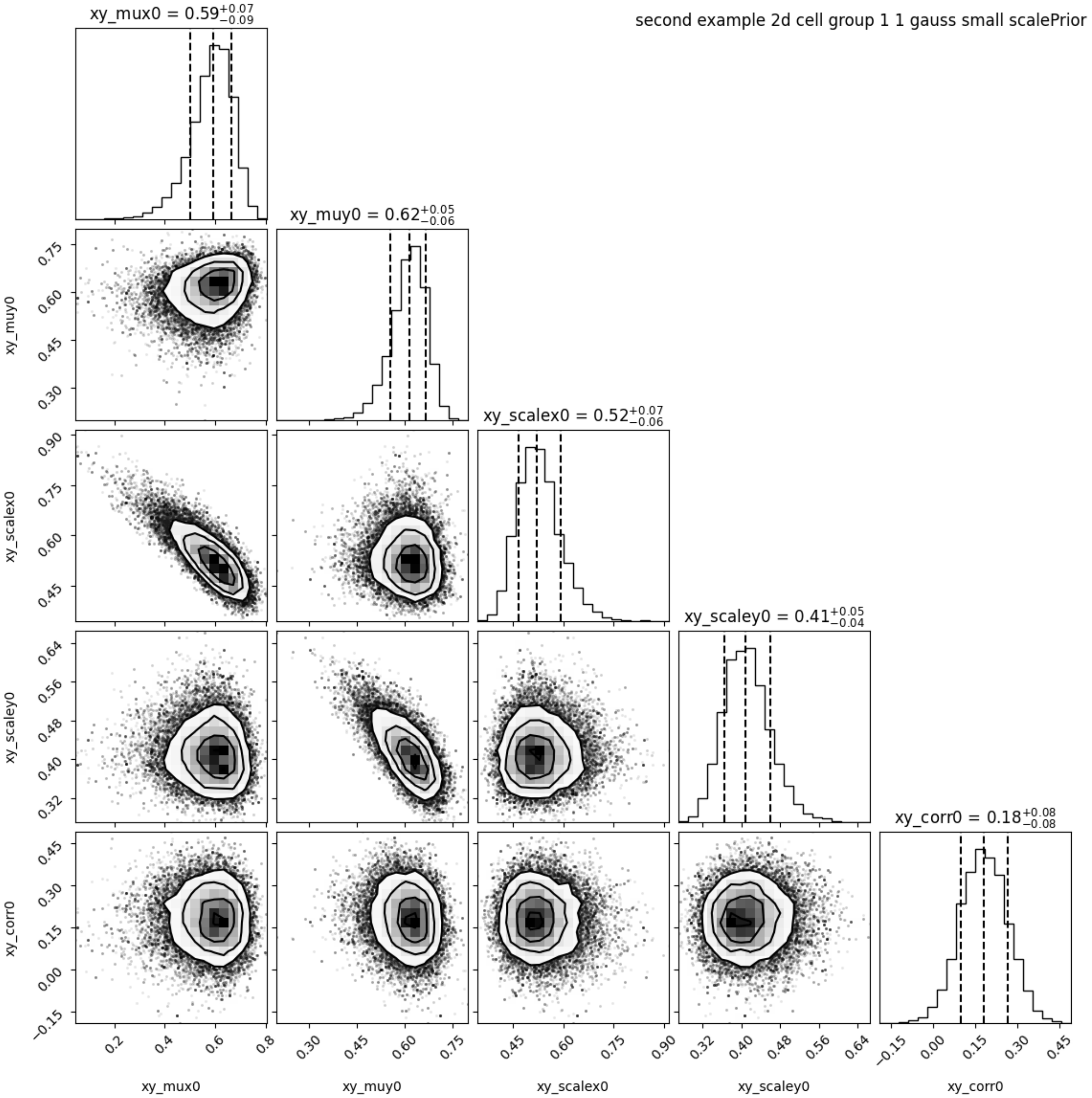
```
$ nnorm=1
$ baredSC_2d \
    --input example/nih3t3_generated_second.txt \
    --geneXColName 1_0.5_0.5_0.5_0.5_0.5_x \
    --geneYColName 1_0.5_0.5_0.5_0.5_0.5_y \
    --metadata1ColName group \
    --metadata1Values group1 \
    --output example/second_example_2d_cellgroup1_${nnorm}gauss_nx20_smallSP \
    --nnorm ${nnorm} \
    --nx 20 --ny 20 \
    --scalePrior 0.1 \
    --figure example/second_example_2d_cellgroup1_${nnorm}gauss_nx20_smallSP.png \
```

(continues on next page)

(continued from previous page)

```
    --title "second example 2d cell group 1 ${nnorm} gauss small scalePrior"
```

We see that the correlation drop to 0.18 +/- 0.08.



On the contrary, if we know that there is a correlation we can increase this value in order to remove the penalty on high correlation coefficient.

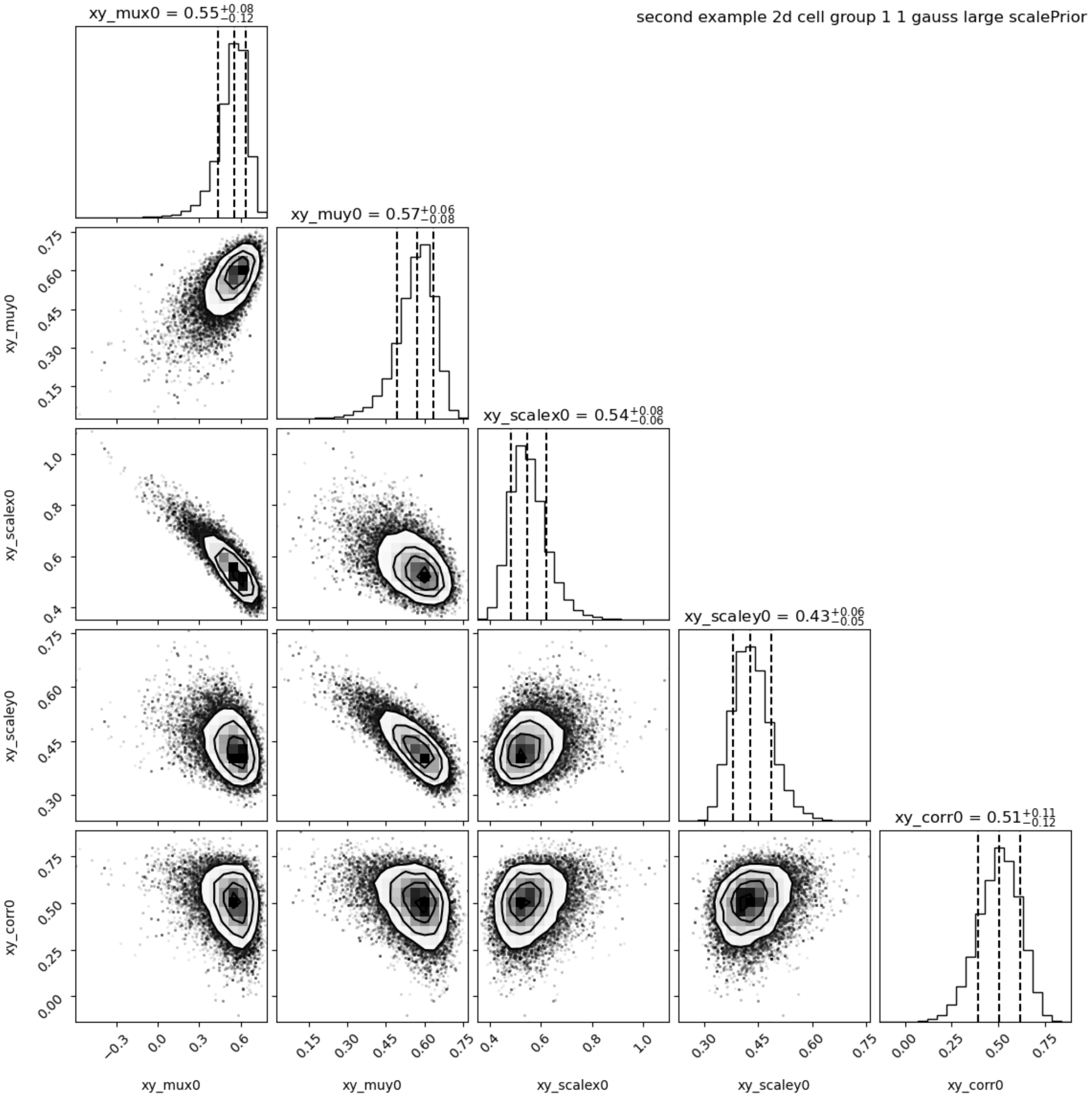
```
$ nnorm=1
$ baredSC_2d \
    --input example/nih3t3_generated_second.txt \
    --geneXColName 1_0.5_0.5_0.5_0.5_0.5_x \
    --geneYColName 1_0.5_0.5_0.5_0.5_0.5_y \
    --metadata1ColName group \
```

(continues on next page)

(continued from previous page)

```
    --metadata1Values group1 \
    --output example/second_example_2d_cellgroup1_${nnorm}gauss_nx20_largeSP \
    --nnorm ${nnorm} \
    --nx 20 --ny 20 \
    --scalePrior 3 \
    --figure example/second_example_2d_cellgroup1_${nnorm}gauss_nx20_largeSP.png \
    --title "second example 2d cell group 1 ${nnorm} gauss large scalePrior"
```

We see that the correlation is now at 0.51 +/- 0.11.



However, these settings may detect (anti-)correlations in situation where there is no, that's why we recommand the default value if you don't have any knowledge on the correlation you expect.

1.5 Use baredSC within python

This part is not a proper documentation of baredSC as a package as we are missing a lot of documentation of our methods. It just gives indications on how it can be used.

1.5.1 baredSC_1d

Use the `npz` content

baredSC is mainly used with the command line interface. By default, only the numpy compressed `.npz` file is output, but if the `--figure` argument is used, it outputs much more. All the outputs are described in the *Outputs* page.

We provide two examples of plots using the text outputs in *Compare means from baredSC results* and *Customize your baredSC plots*. However, sometimes the user may want to plot information which is not part of the text outputs. We describe here a script which will use the baredSC_1d output to plot a custom error bars.

First we get the value of parameters at each step of MCMC:

```
In [1]: import numpy as np
   ...: import baredSC.oned
   ...: import matplotlib.pyplot as plt
   ...:
   ...: output_baredSC = "../example/first_example_1d_2gauss.npz"
   ...:
   ...: # First option, use the function extract_from_npz
   ...: mu, cov, ox, oxpdf, x, logprob_values, samples = \
   ...:   baredSC.oned.extract_from_npz(output_baredSC)
   ...: # mu is the mean of each parameter,
   ...: # cov is the covariance of parameters
   ...: # ox is the oversampled x to compute the poisson noise pdf
   ...: # oxpdf is the oversampled x to compute the pdf
   ...: # x is the x used to compute the likelihood
   ...: # logprob_values is the value of log likelihood at each step fo the MCMC
   ...: # samples is the value of each parameter at each step of the MCMC
   ...:
   ...: # Second option, get the samples and x, oxpdf from the npz:
   ...: mcmc_res = np.load(output_baredSC, allow_pickle=True)
   ...: samples = mcmc_res['samples']
   ...: x = mcmc_res['x']
   ...: oxpdf = mcmc_res['oxpdf']
   ...:
   ...: # From the sample size we deduce the number of Gaussians
```

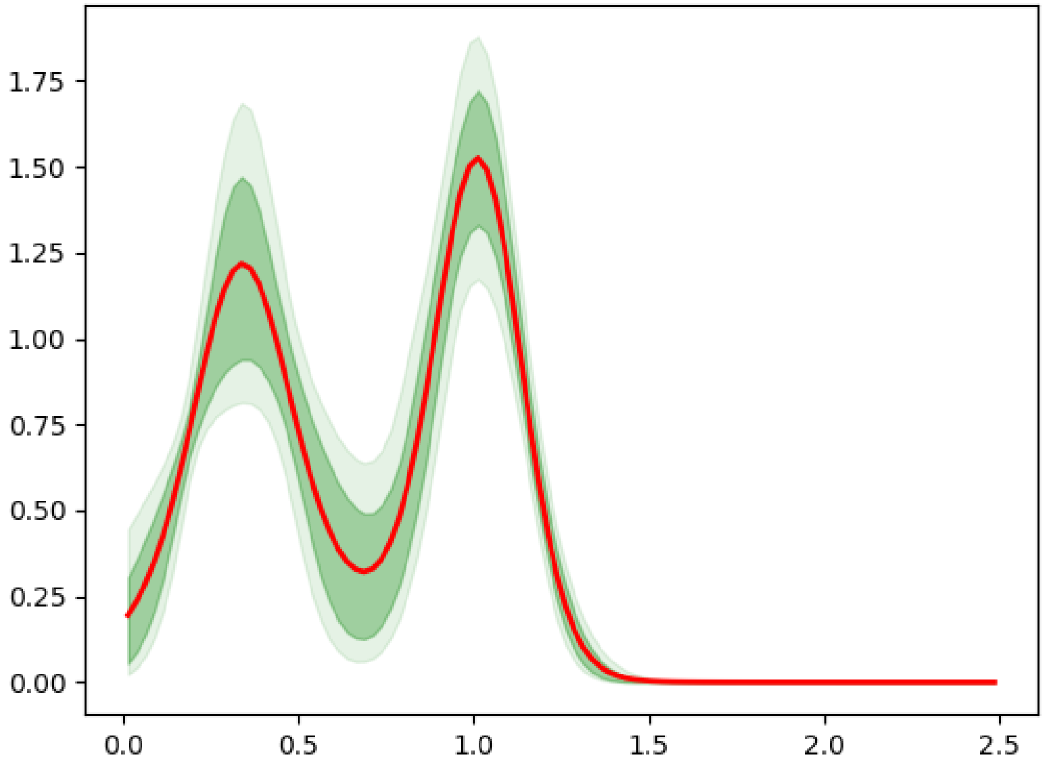
(continues on next page)

(continued from previous page)

```
   ...: nnorm = (samples.shape[1] + 1) // 3
   ...: # We display the parameter names:
   ...: p_names = [f'{pn}{i}' for i in range(nnorm) for pn in ['amp', 'mu', 'scale']][1:]
   ...: print(f'The parameters are: {p_names}')
   ...:
Reading
Read. It took 0.02 seconds.
The parameters are: ['mu0', 'scale0', 'amp1', 'mu1', 'scale1']
```

Then we compute the pdf for each step of the MCMC and we plot the pdf with the custom error bars:

```
In [2]: # We assume x is equally spaced
   ...: dx = x[1] - x[0]
   ...: nx = x.size
   ...: noxpdf = oxpdf.size
   ...: # We assume oxpdf is equally spaced
   ...: odxpdf = oxpdf[1] - oxpdf[0]
   ...:
   ...: # Compute the pdf for each sample
   ...: # This can be long
   ...: pdf = np.array([baredSC.oned.get_pdf(p, nx, noxpdf, oxpdf, odxpdf)
   ...:                 for p in samples])
   ...:
   ...: xmin = x[0] - dx / 2
   ...: xmax = x[-1] + dx / 2
   ...:
   ...: my_custom_quantiles = {'0.3': [0.25, 0.75], '0.1': [0.1, 0.9]}
   ...:
   ...: plt.figure()
   ...: for alpha in my_custom_quantiles:
   ...:   pm = np.quantile(pdf, my_custom_quantiles[alpha][0], axis=0)
   ...:   pp = np.quantile(pdf, my_custom_quantiles[alpha][1], axis=0)
   ...:   plt.fill_between(x, pm, pp, color='g', alpha=float(alpha),
   ...:   rasterized=True)
   ...: # Mean
   ...: plt.plot(x, np.mean(pdf, axis=0), 'r', lw=2, rasterized=True)
   ...:
Out[2]: [<matplotlib.lines.Line2D at 0x7ff1de76ddd0>]
```



Run baredSC_1d

You can also run the MCMC from python directly.

However, it requires formating of the input:

```
In [3]: import numpy as np
   ...: import pandas as pd
   ...: from scipy.stats import lognorm, truncnorm, poisson
   ...: from baredSC.baredSC_1d import gauss_mcmc
   ...:
   ...: # I generate 200 cells with normal expression at 1.5 with scale of 0.2
   ...: # In the Seurat scale (log(1 + 10^4 X))
   ...: n_cells = 200
   ...: cur_loc = 1.5
   ...: cur_scale = 0.2
   ...: N = lognorm.rvs(s=0.3, scale=16000, size=n_cells, random_state=1).astype(int)
   ...: expression = truncnorm.rvs(- cur_loc / cur_scale, np.inf,
   ...:                            loc=cur_loc, scale=cur_scale,
   ...:                            size=n_cells,
   ...:                            random_state=2)
   ...:
   ...: ks = poisson.rvs(mu=N * 1e-4 * (np.exp(expression) - 1),
```

(continues on next page)

(continued from previous page)

```
   ...:                  random_state=3)
   ...:
   ...: # I need to put the ks and the N in a data frame:
   ...: # The column containing the total number of UMI per cell
   ...: # must be 'nCount_RNA'
   ...: data = pd.DataFrame({'my_gene': ks, 'nCount_RNA': N})
   ...:
```

Then the actual MCMC can be run with:

```
In [4]: results = gauss_mcmc(data=data,
   ...:                      col_gene='my_gene', # Put here the colname you put in your␣
↪data
   ...:                      nx=50, # Number of bins in x
   ...:                      osampx=10, # Oversampling factor of the Poisson distribution
   ...:                      osampxpdf=5, # Oversampling factor of the PDF
   ...:                      xmin=0,
   ...:                      xmax=3,
   ...:                      min_scale=0.1, # Minimal value of the scale
   ...:                      xscale="Seurat",
   ...:                      target_sum=10000,
   ...:                      nnorm=1, # We use models with a single Gaussian
   ...:                      nsamples_mcmc=100000, # Number of steps in the MCMC
   ...:                      nsamples_burn=25000, # Number of steps in the burning phase␣
↪of MCMC (we recommand nsampMCMC / 4)
   ...:                      nsplit_burn=10, # The burning phase is splitted in multiple␣
↪sub-phase where the temperature is decreasing
   ...:                      T0_burn=100.0,
   ...:                      output='temp', # Where the npz output should be stored
   ...:                      seed=1)
   ...: print(f'results contains {len(results)} items.')
   ...: # The results are:
   ...: # mu, cov, ox, oxpdf, x, logprob_values, samples
   ...: # mu is the mean of each parameter,
   ...: # cov is the covariance of parameters
   ...: # ox is the oversampled x to compute the poisson noise pdf
   ...: # oxpdf is the oversampled x to compute the pdf
   ...: # x is the x used to compute the likelihood
   ...: # logprob_values is the value of log likelihood at each step fo the MCMC
   ...: # samples is the value of each parameter at each step of the MCMC
   ...:

Step 1000, acceptance rate (since last printing): 0.2330
Step 2000, acceptance rate (since last printing): 0.2580

Step 1000, acceptance rate (since last printing): 0.2400
Step 2000, acceptance rate (since last printing): 0.2370

Step 1000, acceptance rate (since last printing): 0.2450
Step 2000, acceptance rate (since last printing): 0.2430

Step 1000, acceptance rate (since last printing): 0.2350
```

(continues on next page)

(continued from previous page)

```
Step 2000, acceptance rate (since last printing): 0.3930

Step 1000, acceptance rate (since last printing): 0.2580
Step 2000, acceptance rate (since last printing): 0.2060

Step 1000, acceptance rate (since last printing): 0.2680
Step 2000, acceptance rate (since last printing): 0.2440

Step 1000, acceptance rate (since last printing): 0.2640
Step 2000, acceptance rate (since last printing): 0.2440

Step 1000, acceptance rate (since last printing): 0.2350
Step 2000, acceptance rate (since last printing): 0.2400

Step 1000, acceptance rate (since last printing): 0.2450
Step 2000, acceptance rate (since last printing): 0.2340

Step 1000, acceptance rate (since last printing): 0.2530
Step 2000, acceptance rate (since last printing): 0.2460

Step   1000, acceptance rate (since last printing): 0.2060
Step   2000, acceptance rate (since last printing): 0.2330
Step   3000, acceptance rate (since last printing): 0.2470
Step   4000, acceptance rate (since last printing): 0.2410
Step   5000, acceptance rate (since last printing): 0.2360
Step   6000, acceptance rate (since last printing): 0.2380
Step   7000, acceptance rate (since last printing): 0.2210
Step   8000, acceptance rate (since last printing): 0.2050
Step   9000, acceptance rate (since last printing): 0.2500
Step  10000, acceptance rate (since last printing): 0.2620
Step  11000, acceptance rate (since last printing): 0.2240
Step  12000, acceptance rate (since last printing): 0.2310
Step  13000, acceptance rate (since last printing): 0.2350
Step  14000, acceptance rate (since last printing): 0.2480
Step  15000, acceptance rate (since last printing): 0.2360
Step  16000, acceptance rate (since last printing): 0.2400
Step  17000, acceptance rate (since last printing): 0.2190
Step  18000, acceptance rate (since last printing): 0.2260
Step  19000, acceptance rate (since last printing): 0.2360
Step  20000, acceptance rate (since last printing): 0.2350
Step  21000, acceptance rate (since last printing): 0.2380
Step  22000, acceptance rate (since last printing): 0.2100
Step  23000, acceptance rate (since last printing): 0.2500
Step  24000, acceptance rate (since last printing): 0.2510
Step  25000, acceptance rate (since last printing): 0.2620
Step  26000, acceptance rate (since last printing): 0.2480
Step  27000, acceptance rate (since last printing): 0.2610
Step  28000, acceptance rate (since last printing): 0.2530
Step  29000, acceptance rate (since last printing): 0.2180
Step  30000, acceptance rate (since last printing): 0.2410
Step  31000, acceptance rate (since last printing): 0.2340
Step  32000, acceptance rate (since last printing): 0.2170
```

(continues on next page)

(continued from previous page)

```
Step  33000, acceptance rate (since last printing): 0.2290
Step  34000, acceptance rate (since last printing): 0.2440
Step  35000, acceptance rate (since last printing): 0.2550
Step  36000, acceptance rate (since last printing): 0.2360
Step  37000, acceptance rate (since last printing): 0.2060
Step  38000, acceptance rate (since last printing): 0.2360
Step  39000, acceptance rate (since last printing): 0.2630
Step  40000, acceptance rate (since last printing): 0.2340
Step  41000, acceptance rate (since last printing): 0.2150
Step  42000, acceptance rate (since last printing): 0.2460
Step  43000, acceptance rate (since last printing): 0.2600
Step  44000, acceptance rate (since last printing): 0.2210
Step  45000, acceptance rate (since last printing): 0.2310
Step  46000, acceptance rate (since last printing): 0.2350
Step  47000, acceptance rate (since last printing): 0.2210
Step  48000, acceptance rate (since last printing): 0.2280
Step  49000, acceptance rate (since last printing): 0.2480
Step  50000, acceptance rate (since last printing): 0.2200
Step  51000, acceptance rate (since last printing): 0.2370
Step  52000, acceptance rate (since last printing): 0.2270
Step  53000, acceptance rate (since last printing): 0.2590
Step  54000, acceptance rate (since last printing): 0.2170
Step  55000, acceptance rate (since last printing): 0.2240
Step  56000, acceptance rate (since last printing): 0.2530
Step  57000, acceptance rate (since last printing): 0.1990
Step  58000, acceptance rate (since last printing): 0.2370
Step  59000, acceptance rate (since last printing): 0.2200
Step  60000, acceptance rate (since last printing): 0.2560
Step  61000, acceptance rate (since last printing): 0.2360
Step  62000, acceptance rate (since last printing): 0.2290
Step  63000, acceptance rate (since last printing): 0.2600
Step  64000, acceptance rate (since last printing): 0.2060
Step  65000, acceptance rate (since last printing): 0.2260
Step  66000, acceptance rate (since last printing): 0.2550
Step  67000, acceptance rate (since last printing): 0.2470
Step  68000, acceptance rate (since last printing): 0.2410
Step  69000, acceptance rate (since last printing): 0.2490
Step  70000, acceptance rate (since last printing): 0.2450
Step  71000, acceptance rate (since last printing): 0.2160
Step  72000, acceptance rate (since last printing): 0.2130
Step  73000, acceptance rate (since last printing): 0.2340
Step  74000, acceptance rate (since last printing): 0.2410
Step  75000, acceptance rate (since last printing): 0.2290
Step  76000, acceptance rate (since last printing): 0.2240
Step  77000, acceptance rate (since last printing): 0.2350
Step  78000, acceptance rate (since last printing): 0.2460
Step  79000, acceptance rate (since last printing): 0.2280
Step  80000, acceptance rate (since last printing): 0.2580
Step  81000, acceptance rate (since last printing): 0.2530
Step  82000, acceptance rate (since last printing): 0.2240
Step  83000, acceptance rate (since last printing): 0.2490
Step  84000, acceptance rate (since last printing): 0.2190
```

(continues on next page)

(continued from previous page)

```
Step  85000, acceptance rate (since last printing): 0.2350
Step  86000, acceptance rate (since last printing): 0.2160
Step  87000, acceptance rate (since last printing): 0.2480
Step  88000, acceptance rate (since last printing): 0.2460
Step  89000, acceptance rate (since last printing): 0.2430
Step  90000, acceptance rate (since last printing): 0.2400
Step  91000, acceptance rate (since last printing): 0.2230
Step  92000, acceptance rate (since last printing): 0.2390
Step  93000, acceptance rate (since last printing): 0.2280
Step  94000, acceptance rate (since last printing): 0.2430
Step  95000, acceptance rate (since last printing): 0.2350
Step  96000, acceptance rate (since last printing): 0.2210
Step  97000, acceptance rate (since last printing): 0.2490
Step  98000, acceptance rate (since last printing): 0.2540
Step  99000, acceptance rate (since last printing): 0.2290
Step 100000, acceptance rate (since last printing): 0.2270
Saving
Saved. It took 0.13 seconds.
results contains 7 items.
```

1.5.2 baredSC_2d

Use the npz content

baredSC is mainly used with the command line interface. By default, only the numpy compressed .npz file is output, but if the --figure argument is used, it outputs much more. All the outputs are described in the *Outputs* page.

We provide an example of a plot using the text output _pdf2d.txt in *Customize your baredSC plots*. However, sometimes the user may want to plot information which is not part of the text outputs. We describe here a script which will use the baredSC_2d output to plot the mean and median on the same plot and another script which will use more bins in the output to get smoother results.

First we get the value of parameters at each step of MCMC:

```
In [5]: import numpy as np
   ...: import baredSC.twod
   ...: import matplotlib.pyplot as plt
   ...:
   ...: output_baredSC = "../example/second_example_2d_cellgroup1_1gauss_nx20.npz"
   ...:
   ...: # First option, use the function extract_from_npz
   ...: mu, cov, ox, oy, oxpdf, oypdf, x, y, \
   ...:   logprob_values, samples = \
   ...:   baredSC.twod.extract_from_npz(output_baredSC)
   ...: # mu is the mean of each parameter,
   ...: # cov is the covariance of parameters
   ...: # ox, oy are the oversampled x, y to compute the poisson noise pdf
   ...: # oxpdf, oypdf are the oversampled x, y to compute the pdf
   ...: # x, y are the x, y used to compute the likelihood
   ...: # logprob_values is the value of log likelihood at each step fo the MCMC
   ...: # samples is the value of each parameter at each step of the MCMC
   ...:
```

(continues on next page)

(continued from previous page)

```
   ...: # Second option, get the samples and x, y, oxpdf, oypdf, samples from the npz:
   ...: mcmc_res = np.load(output_baredSC, allow_pickle=True)
   ...: samples = mcmc_res['samples']
   ...: x = mcmc_res['x']
   ...: y = mcmc_res['y']
   ...: oxpdf = mcmc_res['oxpdf']
   ...: oypdf = mcmc_res['oypdf']
   ...:
   ...: # From the sample size we deduce the number of Gaussians
   ...: nnorm = (samples.shape[1] + 1) // 6
   ...: # We display the parameter names:
   ...: p_names = [f'{pn}{i}' for i in range(nnorm)
   ...:            for pn in ['xy_amp', 'xy_mux', 'xy_muy', 'xy_scalex',
   ...:                       'xy_scaley', 'xy_corr']][1:]
   ...: print(f'The parameters are: {p_names}')
   ...:
Reading
Read. It took 0.02 seconds.
The parameters are: ['xy_mux0', 'xy_muy0', 'xy_scalex0', 'xy_scaley0', 'xy_corr0']
```

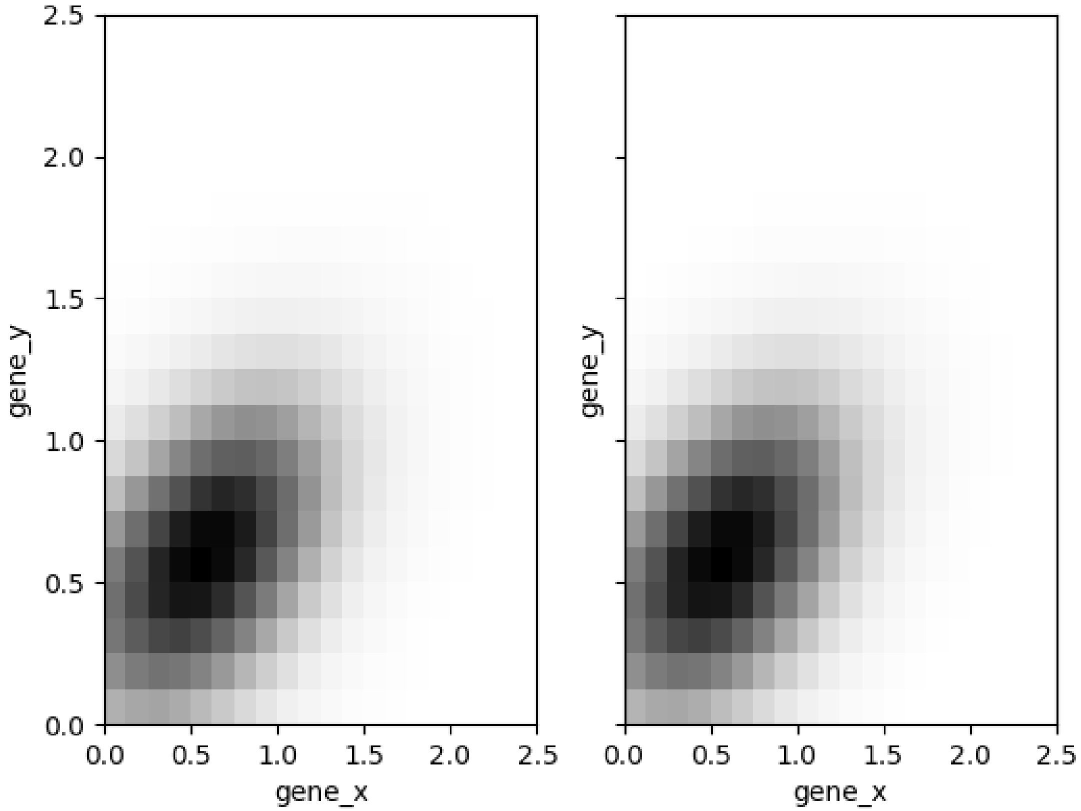
Then we compute the pdf for each step of the MCMC and we plot the mean and median:

```
In [6]: # We assume x and y are equally spaced
   ...: dx = x[1] - x[0]
   ...: nx = x.size
   ...: dy = y[1] - y[0]
   ...: ny = y.size
   ...: noxpdf = oxpdf.size
   ...: # We assume oxpdf is equally spaced
   ...: odxpdf = oxpdf[1] - oxpdf[0]
   ...:
   ...: noypdf = oypdf.size
   ...: # We assume oypdf is equally spaced
   ...: odypdf = oypdf[1] - oypdf[0]
   ...:
   ...: odxypdf = odxpdf * odypdf
   ...: oxypdf = np.array(np.meshgrid(oxpdf, oypdf)).transpose(1, 2, 0)
   ...:
   ...: # Compute the pdf for each sample
   ...: # This can be long
   ...: pdf = np.array([baredSC.twod.get_pdf(p, nx, ny, noxpdf,
   ...:                                      noypdf, oxypdf, odxypdf)
   ...:                 for p in samples])
   ...: # We plot:
   ...: xmin = x[0] - dx / 2
   ...: xmax = x[-1] + dx / 2
   ...: ymin = y[0] - dy / 2
   ...: ymax = y[-1] + dy / 2
   ...:
   ...: x_borders = np.linspace(xmin, xmax, len(x) + 1)
   ...: y_borders = np.linspace(ymin, ymax, len(y) + 1)
   ...:
```

(continues on next page)

(continued from previous page)

```
   ...: # Plot 2 panels plot
   ...: fig, axs = plt.subplots(1, 2, sharex='row', sharey='row')
   ...: axs[0].pcolormesh(x_borders, y_borders, np.mean(pdf, axis=0),
   ...:                   shading='flat', rasterized=True, cmap='Greys')
   ...: axs[0].set_xlabel('gene_x')
   ...: axs[0].set_ylabel('gene_y')
   ...: axs[1].pcolormesh(x_borders, y_borders, np.median(pdf, axis=0),
   ...:                   shading='flat', rasterized=True, cmap='Greys')
   ...: axs[1].set_xlabel('gene_x')
   ...: axs[1].set_ylabel('gene_y')
   ...:
Out[6]: Text(0, 0.5, 'gene_y')
```



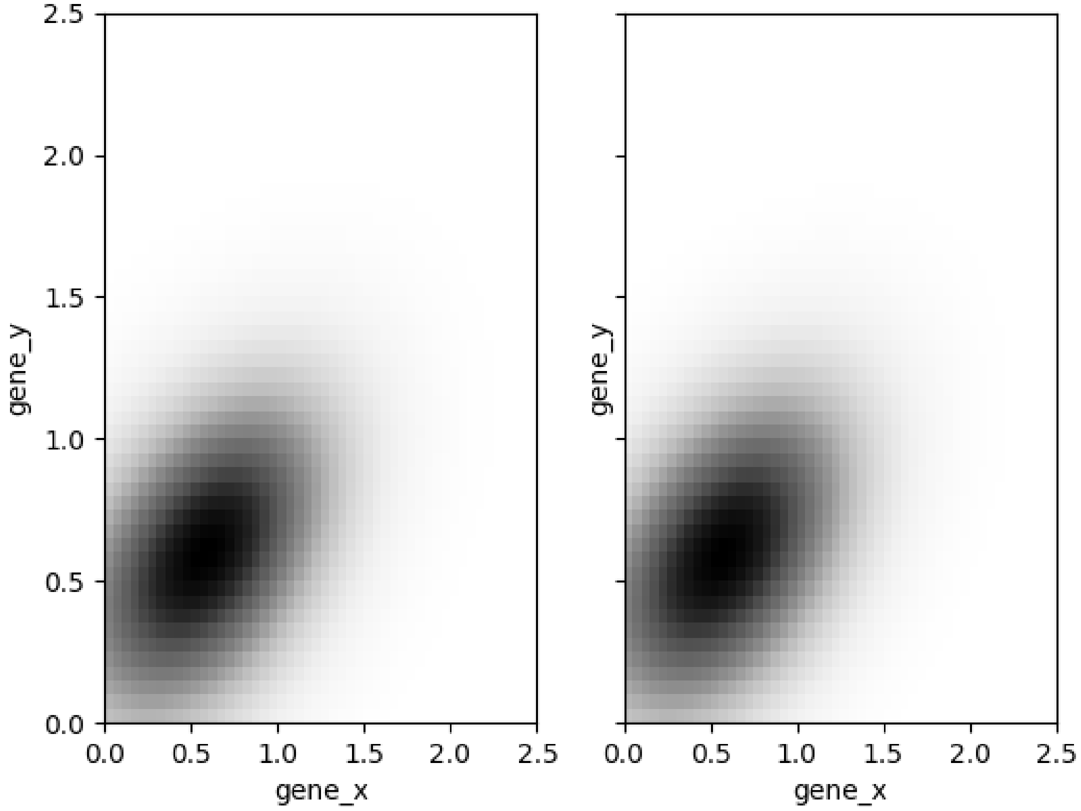
If you want to get more bins, you just need to change x and y. We want to warn the user that what will be plotted will be different from what was used for the likelihood evaluation:

```
In [1]: # We assume x and y are equally spaced
   ...: dx = x[1] - x[0]
   ...: dy = y[1] - y[0]
   ...: xmin = x[0] - dx / 2
   ...: xmax = x[-1] + dx / 2
   ...: ymin = y[0] - dy / 2
   ...: ymax = y[-1] + dy / 2
```

(continues on next page)

(continued from previous page)

```
   ...:
   ...: # We set pretty_bins_x and y
   ...: pretty_bins_x = 50
   ...: pretty_bins_y = 50
   ...: from baredSC.common import get_bins_centers
   ...: nx = pretty_bins_x
   ...: x = get_bins_centers(xmin, xmax, nx)
   ...: dx = x[1] - x[0]
   ...: noxpdf = nx
   ...: oxpdf = x
   ...: odxpdf = dx
   ...: ny = pretty_bins_y
   ...: y = get_bins_centers(ymin, ymax, ny)
   ...: dy = y[1] - y[0]
   ...: noypdf = ny
   ...: oypdf = y
   ...: odypdf = dy
   ...:
   ...: odxypdf = odxpdf * odypdf
   ...: oxypdf = np.array(np.meshgrid(oxpdf, oypdf)).transpose(1, 2, 0)
   ...:
   ...: # Compute the pdf for each sample
   ...: # This can be long
   ...: pdf = np.array([baredSC.twod.get_pdf(p, nx, ny, noxpdf,
   ...:                                      noypdf, oxypdf, odxypdf)
   ...:                 for p in samples])
   ...: # We plot:
   ...: x_borders = np.linspace(xmin, xmax, len(x) + 1)
   ...: y_borders = np.linspace(ymin, ymax, len(y) + 1)
   ...:
   ...:
   ...: # Plot 2 panels plot
   ...: fig, axs = plt.subplots(1, 2, sharex='row', sharey='row')
   ...: axs[0].pcolormesh(x_borders, y_borders, np.mean(pdf, axis=0),
   ...:                   shading='flat', rasterized=True, cmap='Greys')
   ...: axs[0].set_xlabel('gene_x')
   ...: axs[0].set_ylabel('gene_y')
   ...: axs[1].pcolormesh(x_borders, y_borders, np.median(pdf, axis=0),
   ...:                   shading='flat', rasterized=True, cmap='Greys')
   ...: axs[1].set_xlabel('gene_x')
   ...: axs[1].set_ylabel('gene_y')
   ...:
```



Run baredSC_2d

You can also run the MCMC from python directly.

However, it requires formating of the input:

```
In [7]: import numpy as np
   ...: import pandas as pd
   ...: from scipy.stats import lognorm, truncnorm, poisson
   ...: from baredSC.baredSC_2d import gauss_mcmc
   ...: from baredSC.twod import trunc_norm2d
   ...:
   ...:
   ...: def trunc_norm_2d(mu, sigma, corr, size, seed):
   ...:   try:
   ...:     rng = np.random.default_rng(seed)
   ...:   except AttributeError:
   ...:     # For older numpy versions:
   ...:     np.random.seed(seed)
   ...:     rng = np.random
   ...:   cov = np.array([[sigma[0] * sigma[0], sigma[0] * sigma[1] * corr],
   ...:                   [sigma[0] * sigma[1] * corr, sigma[1] * sigma[1]]])
   ...:   values = rng.multivariate_normal(mu, cov, size)
```

(continues on next page)

(continued from previous page)

```
   ...:   mask_0 = [v[0] < 0 or v[1] < 0 for v in values]
   ...:   # Because we want only positive expression:
   ...:   while sum(mask_0) > 0:
   ...:     values[mask_0] = rng.multivariate_normal(mu, cov, sum(mask_0))
   ...:     mask_0 = [v[0] < 0 or v[1] < 0 for v in values]
   ...:   return(values)
   ...:
   ...:
   ...: # I generate 200 cells with normal expression at 1.5 with scale of 0.2 with␣
↪correlation of 0.5
   ...: # In the Seurat scale (log(1 + 10^4 X))
   ...: n_cells = 200
   ...: cur_mu = [1.5, 1.5]
   ...: cur_sigma = [0.2, 0.2]
   ...: cur_corr = 0.5
   ...: N = lognorm.rvs(s=0.3, scale=16000, size=n_cells, random_state=1).astype(int)
   ...: expression = trunc_norm_2d(mu=cur_mu, sigma=cur_sigma,
   ...:                            corr=cur_corr,
   ...:                            size=n_cells,
   ...:                            seed=2)
   ...: exp_values_x, exp_values_y  = np.transpose(expression)
   ...: ks_x = poisson.rvs(mu=N * 1e-4 * (np.exp(exp_values_x) - 1),
   ...:                    random_state=3)
   ...: ks_y = poisson.rvs(mu=N * 1e-4 * (np.exp(exp_values_y) - 1),
   ...:                    random_state=4)
   ...:
   ...: # I need to put the ks and the N in a data frame:
   ...: # The column containing the total number of UMI per cell
   ...: # must be 'nCount_RNA'
   ...: data = pd.DataFrame({'my_gene_x': ks_x,
   ...:                      'my_gene_y': ks_y,
   ...:                      'nCount_RNA': N})
   ...:
```

Then the actual MCMC can be run with:

```
In [8]: results = gauss_mcmc(data=data,
   ...:                      genex='my_gene_x', # Put here the colname you put in your␣
↪data
   ...:                      geney='my_gene_y', # Put here the colname you put in your␣
↪data
   ...:                      nx=20, # Number of bins in x
   ...:                      osampx=10, # Oversampling factor of the Poisson distribution
   ...:                      osampxpdf=5, # Oversampling factor of the PDF
   ...:                      xmin=0,
   ...:                      xmax=3,
   ...:                      ny=20, # Number of bins in y
   ...:                      osampy=10, # Oversampling factor of the Poisson distribution
   ...:                      osampypdf=5, # Oversampling factor of the PDF
   ...:                      ymin=0,
   ...:                      ymax=3,
   ...:                      min_scale_x=0.1, # Minimal value of the scale in x
```

(continues on next page)

(continued from previous page)

```
   ...:                       min_scale_y=0.1, # Minimal value of the scale in y
   ...:                       scale_prior=0.3, # Scale of the truncnorm used in the prior␣
↪for the correlation
   ...:                       scale="Seurat",
   ...:                       target_sum=10000,
   ...:                       nnorm=1, # We use models with a single Gaussian
   ...:                       nsamples_mcmc=100000, # Number of steps in the MCMC
   ...:                       nsamples_burn=25000, # Number of steps in the burning phase␣
↪of MCMC (we recommand nsampMCMC / 4)
   ...:                       nsplit_burn=10, # The burning phase is splitted in multiple␣
↪sub-phase where the temperature is decreasing
   ...:                       T0_burn=100.0,
   ...:                       output='temp', # Where the npz output should be stored
   ...:                       seed=1)
   ...: print(f'results contains {len(results)} items.')
   ...: # The results are:
   ...: # mu, cov, ox, oy, oxpdf, oypdf, x, y, \
   ...: #   logprob_values, samples
   ...: # mu is the mean of each parameter,
   ...: # cov is the covariance of parameters
   ...: # ox, oy are the oversampled x, y to compute the poisson noise pdf
   ...: # oxpdf, oypdf are the oversampled x, y to compute the pdf
   ...: # x, y are the x, y used to compute the likelihood
   ...: # logprob_values is the value of log likelihood at each step fo the MCMC
   ...: # samples is the value of each parameter at each step of the MCMC
   ...:

Step 1000, acceptance rate (since last printing): 0.2320
Step 2000, acceptance rate (since last printing): 0.2180

Step 1000, acceptance rate (since last printing): 0.2710
Step 2000, acceptance rate (since last printing): 0.2270

Step 1000, acceptance rate (since last printing): 0.2670
Step 2000, acceptance rate (since last printing): 0.1990

Step 1000, acceptance rate (since last printing): 0.2650
Step 2000, acceptance rate (since last printing): 0.2220

Step 1000, acceptance rate (since last printing): 0.3320
Step 2000, acceptance rate (since last printing): 0.1970

Step 1000, acceptance rate (since last printing): 0.2010
Step 2000, acceptance rate (since last printing): 0.2160

Step 1000, acceptance rate (since last printing): 0.2570
Step 2000, acceptance rate (since last printing): 0.2210

Step 1000, acceptance rate (since last printing): 0.2650
Step 2000, acceptance rate (since last printing): 0.2300

Step 1000, acceptance rate (since last printing): 0.2410
```

(continues on next page)

(continued from previous page)

```
Step 2000, acceptance rate (since last printing): 0.2590

Step 1000, acceptance rate (since last printing): 0.2520
Step 2000, acceptance rate (since last printing): 0.2280

Step   1000, acceptance rate (since last printing): 0.2490
Step   2000, acceptance rate (since last printing): 0.2170
Step   3000, acceptance rate (since last printing): 0.2480
Step   4000, acceptance rate (since last printing): 0.2440
Step   5000, acceptance rate (since last printing): 0.2110
Step   6000, acceptance rate (since last printing): 0.2280
Step   7000, acceptance rate (since last printing): 0.2620
Step   8000, acceptance rate (since last printing): 0.2480
Step   9000, acceptance rate (since last printing): 0.2350
Step  10000, acceptance rate (since last printing): 0.2210
Step  11000, acceptance rate (since last printing): 0.2130
Step  12000, acceptance rate (since last printing): 0.2540
Step  13000, acceptance rate (since last printing): 0.2420
Step  14000, acceptance rate (since last printing): 0.2330
Step  15000, acceptance rate (since last printing): 0.2300
Step  16000, acceptance rate (since last printing): 0.2430
Step  17000, acceptance rate (since last printing): 0.2440
Step  18000, acceptance rate (since last printing): 0.2470
Step  19000, acceptance rate (since last printing): 0.2230
Step  20000, acceptance rate (since last printing): 0.2260
Step  21000, acceptance rate (since last printing): 0.2460
Step  22000, acceptance rate (since last printing): 0.2310
Step  23000, acceptance rate (since last printing): 0.2400
Step  24000, acceptance rate (since last printing): 0.2580
Step  25000, acceptance rate (since last printing): 0.2510
Step  26000, acceptance rate (since last printing): 0.2530
Step  27000, acceptance rate (since last printing): 0.2200
Step  28000, acceptance rate (since last printing): 0.2520
Step  29000, acceptance rate (since last printing): 0.2380
Step  30000, acceptance rate (since last printing): 0.2460
Step  31000, acceptance rate (since last printing): 0.2560
Step  32000, acceptance rate (since last printing): 0.2170
Step  33000, acceptance rate (since last printing): 0.2500
Step  34000, acceptance rate (since last printing): 0.2540
Step  35000, acceptance rate (since last printing): 0.2340
Step  36000, acceptance rate (since last printing): 0.2390
Step  37000, acceptance rate (since last printing): 0.2450
Step  38000, acceptance rate (since last printing): 0.2440
Step  39000, acceptance rate (since last printing): 0.2410
Step  40000, acceptance rate (since last printing): 0.2480
Step  41000, acceptance rate (since last printing): 0.2160
Step  42000, acceptance rate (since last printing): 0.2420
Step  43000, acceptance rate (since last printing): 0.2500
Step  44000, acceptance rate (since last printing): 0.2280
Step  45000, acceptance rate (since last printing): 0.2300
Step  46000, acceptance rate (since last printing): 0.2540
Step  47000, acceptance rate (since last printing): 0.2120
```

(continues on next page)

(continued from previous page)

```
Step  48000, acceptance rate (since last printing): 0.2340
Step  49000, acceptance rate (since last printing): 0.2380
Step  50000, acceptance rate (since last printing): 0.2460
Step  51000, acceptance rate (since last printing): 0.2500
Step  52000, acceptance rate (since last printing): 0.2440
Step  53000, acceptance rate (since last printing): 0.2310
Step  54000, acceptance rate (since last printing): 0.2330
Step  55000, acceptance rate (since last printing): 0.2340
Step  56000, acceptance rate (since last printing): 0.2320
Step  57000, acceptance rate (since last printing): 0.2440
Step  58000, acceptance rate (since last printing): 0.2560
Step  59000, acceptance rate (since last printing): 0.2360
Step  60000, acceptance rate (since last printing): 0.2550
Step  61000, acceptance rate (since last printing): 0.2240
Step  62000, acceptance rate (since last printing): 0.2500
Step  63000, acceptance rate (since last printing): 0.2190
Step  64000, acceptance rate (since last printing): 0.2370
Step  65000, acceptance rate (since last printing): 0.2200
Step  66000, acceptance rate (since last printing): 0.2170
Step  67000, acceptance rate (since last printing): 0.2250
Step  68000, acceptance rate (since last printing): 0.2400
Step  69000, acceptance rate (since last printing): 0.2520
Step  70000, acceptance rate (since last printing): 0.2780
Step  71000, acceptance rate (since last printing): 0.2220
Step  72000, acceptance rate (since last printing): 0.2310
Step  73000, acceptance rate (since last printing): 0.2450
Step  74000, acceptance rate (since last printing): 0.2380
Step  75000, acceptance rate (since last printing): 0.2190
Step  76000, acceptance rate (since last printing): 0.2470
Step  77000, acceptance rate (since last printing): 0.2450
Step  78000, acceptance rate (since last printing): 0.2320
Step  79000, acceptance rate (since last printing): 0.2380
Step  80000, acceptance rate (since last printing): 0.2530
Step  81000, acceptance rate (since last printing): 0.2330
Step  82000, acceptance rate (since last printing): 0.2370
Step  83000, acceptance rate (since last printing): 0.2360
Step  84000, acceptance rate (since last printing): 0.2310
Step  85000, acceptance rate (since last printing): 0.2290
Step  86000, acceptance rate (since last printing): 0.2660
Step  87000, acceptance rate (since last printing): 0.2400
Step  88000, acceptance rate (since last printing): 0.2110
Step  89000, acceptance rate (since last printing): 0.2380
Step  90000, acceptance rate (since last printing): 0.2530
Step  91000, acceptance rate (since last printing): 0.2200
Step  92000, acceptance rate (since last printing): 0.2400
Step  93000, acceptance rate (since last printing): 0.2170
Step  94000, acceptance rate (since last printing): 0.2320
Step  95000, acceptance rate (since last printing): 0.2320
Step  96000, acceptance rate (since last printing): 0.2260
Step  97000, acceptance rate (since last printing): 0.2440
Step  98000, acceptance rate (since last printing): 0.2390
Step  99000, acceptance rate (since last printing): 0.2790
```

(continues on next page)

(continued from previous page)

```
Step 100000, acceptance rate (since last printing): 0.2310
Saving
Saved. It took 0.17 seconds.
results contains 10 items.
```

1.6 Releases

1.6.1 1.1.2

Bug fix:

- When `--minNeff` was set with `--nsampMCMC`, the number of samples was not multiplied more than once by 10.

Improvements:

- The output `means.txt.gz` is now in documentation.

1.6.2 1.1.1

Improvements:

- The online documentation has been improved.
- More information obtained by `--help`.

1.6.3 1.1.0

Improvements:

- Support annData as input with `--inputAnnData`

1.6.4 1.0.0

First release

1.7 Citation

If you use baredSC in your analysis, you can cite the following paper:

Lucille Lopez-Delisle and Jean-Baptiste Delisle. baredSC: Bayesian Approach to Retrieve Expression Distribution of Single-Cell. bioRxiv 2021.05.26.445740; doi:10.1101/2021.05.26.445740.