BRAINS:
Bayesian Reverberation-mapping Analysis Integrated with Nested Sampling

Yan-Rong Li
Institute of High Energy Physics

November 19, 2018

Contents

1 Installation **2**
1.1 Third-Party Packages . . . 2
1.2 Configure the `Makefile` . . . 2
1.3 Compiling . . . 3

2 Running **4**
2.1 Running Commands . . . 4
2.2 Parameter File . . . 4
2.3 OPTION File for DNest . . . 5

3 BLR Modeling **9**
3.1 Coordinate rotation . . . 9
3.2 Time Lag . . . 9

Part I: Users' Guide

Chapter 1 Installation

1.1 Third-Party Packages

`brains` requires the following third-party packages:

- **GSL**. GSL is used to generate random numbers, perform interpolation, and calculate some special functions. GSL is available at https://www.gnu.org/software/gsl.
- **FFTW**. FFTW is used to implement Gaussian smoothing on the line profiles. FFTW is available at http://fftw.org.
- **LAPACK**. LAPACK is used to perform numerical linear algebra calculations, such as matrix operations and Cholesky decomposition. LAPACK is a fortran library. Its C interface is LAPACKE, included in the LAPACK package. One needs to configure the LAPACK installation options to switch on LAPACKE. LAPACK is available at https://netlib.sandia.gov/lapack.
- **MPICH**. MPICH is a high performace and widely portable implementation of the Message Passing Interface standard. MPICH is available at http://www.mpich.org.
- **DNest**. DNest is a library to implement diffusive nested sampling. DNest is available at https://github.com/LiyrAstroph/DNest_C.

In popular Linux distributions, one can use the system package manager to install the above packages, except for DNest. For example, in Fedora distribution, the corresponding terminal commands are

```
sudo dnf install gsl fftw3 lapack mpich
```

1.2 Configure the `Makefile`

To compile `brains`, one needs first to proporiately configure the `Makefile`. In Linux systems, typical configurations look like

```
ifeq ($(SYSTEM), "Linux")
GSL_INCL    = $(shell pkg-config --cflags gsl)
GSL_LIBS    = $(shell pkg-config --libs gsl)

LAPACK_INCL = -I/usr/include/lapacke
LAPACK_LIBS = -L/usr/lib64 -llapacke -llapack -lblas

DNEST_INCL  = -I /home/liyropt/Projects/GIT/DNest/
DNEST_LIBS  = -L /home/liyropt/Projects/GIT/DNest -ldnest

FFTW_INCL   = $(shell pkg-config --cflags fftw3)
FFTW_LIBS   = $(shell pkg-config --libs fftw3)

MPICHINCL     = $(shell pkg-config --cflags mpich)
MPICHLIB    = $(shell pkg-config --libs mpich)
endif
```

Change the values of the variables in line with your system's configurations.

1.3 Compiling

The command for compiling is

```
make
```

This will create an executable file “ `brains`” in the local directory.

Chapter 2 Running

2.1 Running Commands

First change the directory to where `brains` is installed. To run `brains`, one can use command

```
brains [FILE] [OPTION]
```

or

```
mpiexec [MPI_OPTION] brains [FILE] [OPTION]
```

where `[FILE]` is the name of parameter file, `[MPI_OPTION]` is the options for MPI and `[OPTION]` is the command-line options for `brains`. For example, to run `brains` with 3 cores, use

```
mpiexec -n 3 brains param
```

where `param` is the parameter file.

`brains` recognizes following command-line options:

```
-h
    print help information.
-p
    only do posterior processing.
-r
    restart from the backup.
-t
    specify tempering temperature in posterior processing.
-s
    set a seed for the random number generator.
-c
    only do posterior processing and recalculate the posterior sample information.
-e
    examine the priors.
```

2.2 Parameter File

A typical paramete file looks like

```
% parameter file
% lines beginning with '%' are regarded as comments and are neglected
%

FileDir                     /home/liyropt/Projects/GIT/BRAINS

FlagDim                     2

FlagBLRModel                6

%============================================================================
% data file

ContinuumFile               data/sim_con.txt
LineFile                    data/sim_hb.txt
Line2DFile                  data/sim_hb2d.txt
```

```
%===========================================================================
% reconstruction
NConRecon                   200
FlagTrend                   0
FlagTrendDiff               0
ConConstructFileOut         data/pcon.txt

FlagFixVar                  0
NLineRecon                  100
LineConstructFileOut        data/pline.txt
TranFileOut                 data/tran.txt

NVelRecon                   42
Line2DConstructFileOut      data/pline2d.txt
Line2DDataConstructFileOut  data/pline2d_data.txt
Tran2DFileOut               data/tran2d.txt
Tran2DDataFileOut           data/tran2d_data.txt

NCloudPerCore               5000
NVPerCloud                  5

NTau                        200
RCloudMax                   -1
TimeBack                    -1

FlagCloudsOut               1
CloudsFileOut               data/clouds.txt

%===========================================================================
%
FlagCloudsForceUpdate       1

FlagConSysErr               0
FlagLineSysErr              0

%===========================================================================
% spectral broadening

InstRes                     220


InstResErr                  34.0

InstResFile                 data/arp151_broaden.txt

%===========================================================================
% narrow-line component
% use a gaussian to model the narrow-line component

FlagNarrowLine              0

FluxNarrowLine              1.5
FluxNarrowLineErr           0.50
WidthNarrowLine             14.0
WidthNarrowLineErr          4.4
ShiftNarrowLine             10.0
ShiftNarrowLineErr          0.0

%===========================================================================
%
FlagLineCenter              1
LineCenterErr               62.0

%===========================================================================
% set fixed BLR parameters and their fixed values
% do not put sapce in the strings
% 1: fixed; 0: not fixed;
% values are separated by ":"

BLRParFix                   0000000000
BLRParFixVal                0.0:1.0
```

In parameter file, lines beginning with '%' are regarded as comments and are neglected, so that you can freely add comments for your convinence as in the above example. Also, the orders of parameters can be changed freely. Most of the parameters are given a name that largely indicates the corresponding meanings.

2.3 OPTION File for DNest

The format of option file for DNest looks like as follows,

```
# File containing parameters for DNest
# Put comments at the top, or at the end of the line.
# Do not change the order of lines.
# Lines beginning with '#' are regarded as comments.

2     # Number of particles
2000 # new level interval
2000 # save interval
100   # threadSteps - how many steps each thread should do independently before communication
20   # maximum number of levels
10   # Backtracking scale length (lambda in the paper)
100   # Strength of effect to force histogram to equal push. 0-10 is best. (beta in the paper)
1000   # Maximum number of saves (0 = infinite)
data/sample.txt                # sample file
data/sample_info.txt           # sample_info file
data/levels.txt                # level file
data/sampler_state.txt         # sample state file
data/posterior_sample.txt      # posterior sample file
data/posterior_sample_info.txt  # posterior sample info file
data/limits.txt                # limits file
```

The option file for continuum reconstruction is `OPTIONSCON`, for 1d RM is `OPTIONS1D`, and for 2d RM is `OPTIONS2D`. `brains` will automatically read these options appropriately.

One should not change the orders of lines. Lines beginning with '#' are regarded as comments. There is not a general rule to set the values of options. The most important options are the options for new level interval and maximum number of levels. Sufficently large values will work better, but also will cause extra computation time. The option for maximum number of saves controls the length of the Markov chains. Note that this is not the length of the final posterior sample.

To check whether the values of options are appropriate, one may run the python script `postprocess.py` in the subdirectory to inspect the log-likelihood-curve (Brewer et al. 2011); see also the user mannual in the package DNest3 developed by Brendon J. Brewer, which is available at `https://github.com/eggplantbren/DNest3`.

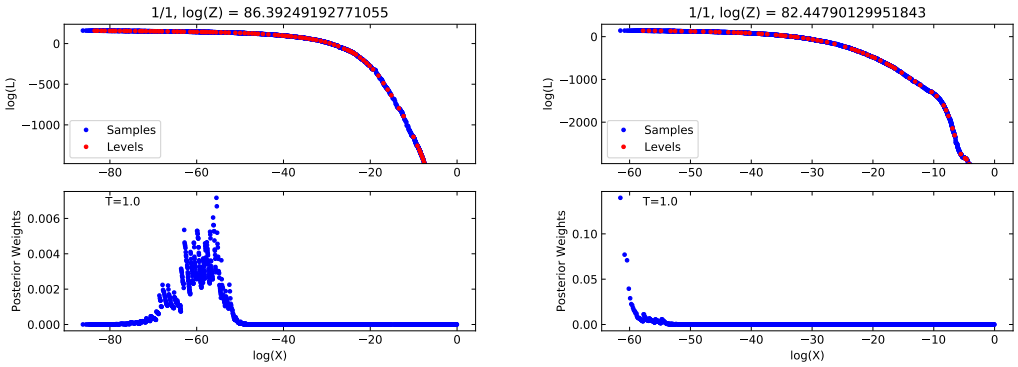


Figure 2.1: Examples for log-likelihood cruve: (left) a good run with sufficiently appropriate options and (right) a bad run with inappropriate options.

Bibliography

Li, Y.-R., Songsheng, Y.-Y., Qiu, J. et al. 2018, ApJ in press (arXiv: 1811.06302)

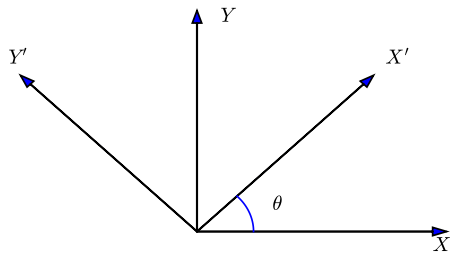
Brewer, B. J. & Foreman-Mackey, D. 2016, Journal of Statistical Software, 86, 1297 (arXiv:1606.03757)

Brewer, B., J. et al., 2011, Statistics and Computing, 21, 649 (arXiv: 0912.2380)

Part II: Broad-Line Region Modeling

Chapter 3 BLR Modeling

3.1 Coordinate rotation



For the rotation of a coordinate (XOY) by an angle of θ to (X'OY'), there are relations

$$\begin{bmatrix} e_{x'} \\ e_{y'} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} e_x \\ e_y \end{bmatrix}. \tag{3.1}$$

and

$$\begin{bmatrix} e_x \\ e_y \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} e_{x'} \\ e_{y'} \end{bmatrix}. \tag{3.2}$$

Therefore, for a vector A, its components in (XOY) and (X'OY') are related by

$$A = [x', y'] \begin{bmatrix} e_{x'} \\ e_{y'} \end{bmatrix} = [x, y] \begin{bmatrix} e_x \\ e_y \end{bmatrix} = [x, y] \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} e_{x'} \\ e_{y'} \end{bmatrix}. \tag{3.3}$$

This yields

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}. \tag{3.4}$$

In right-handed coordinate frame, we perform a rotation around y-axis by an angle of l_θ and then a rotation around z-axis by an angle of l_ϕ. The transformation matrix is

$$\begin{bmatrix} \cos l_\phi & \sin l_\phi & 0 \\ -\sin l_\phi & \cos l_\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos l_\theta & 0 & -\sin l_\theta \\ 0 & 1 & 0 \\ \sin l_\theta & 0 & \cos l_\theta \end{bmatrix} = \begin{bmatrix} \cos l_\phi \cos l_\theta & \sin l_\phi & -\cos l_\phi \sin l_\theta \\ -\sin l_\phi \cos l_\theta & \cos l_\phi & \sin l_\phi \sin l_\theta \\ \sin l_\theta & 0 & \cos l_\theta \end{bmatrix}. \tag{3.5}$$

3.2 Time Lag

The obsever is located at $(D \to \infty, 0, 0)$, i.e., the line of sight is along x–axis. For a cloud at (x, y, z), its time lag is

$$\tau = \sqrt{x^2+y^2+z^2} + \sqrt{(D-x)^2+y^2+z^2} - D \approx r + D(1-x/D) - D \approx r - x, \tag{3.6}$$

where $r = \sqrt{x^2+y^2+z^2}$.

The angle between the line of sight and the line of cloud is

$$\cos\varphi = \frac{D \cdot x}{D \cdot r} = \frac{x}{r}. \tag{3.7}$$

waiting for update...