Tutorial on BayesVP

Release 0.1

Tanvir Hussain

Jul 04, 2019

CONTENTS

1 Installation instructions **3**
1.1 Installation 3

2 Documentation **5**
2.1 Absorption line fit using `bayesvp` 5

`bayesvp` is a Bayesian MCMC parallel Voigt profile fitting routine written by Cameron Liang & Andrey Kravtsov.

`bayesvp` provides a number of helpful executable scripts that work with command line arguments (saved in your environment `PATH`). The main functionality is the MCMC Voigt profile fitting (bvpfit) where the user supplies a config file that specifies parameters for the fitting. These include parameter priors, number of walkers, parallel threads, line spread function, continuum model, Bayesian model comparisons, and etc. There are utility functions that allow users to quickly create an example configuration file, process and plot the chains, process and plot the best fit models and more. For details on the code, refer to the papers Liang & Kravtsov 2017 and Liang et al. 2017.

CHAPTER

ONE

INSTALLATION INSTRUCTIONS

1.1 Installation

`bayesvp` is originally written and tested for Python 2.7. For Python 3.6 and above see the instructions below.

1.1.1 Dependencies

`bayesvp` depends on:

- numpy
- scipy
- matplotlib
- pyfits

You can install these using your favorite Python package manager such as pip or conda.

It also depends on the following MCMC samplers:

- kombine
- emcee

1.1.2 Using pip

The easiest way to install the stable version of `bayesvp` is using pip with the `--user` flag:

```
pip install bayesvp --user
```

To install it system-wide and you might need to add `sudo` in the beginning.

1.1.3 From source

Alternatively, you can get the source by cloning the git repository:

```
git clone https://github.com/cameronliang/bayesvp.git
```

Once you've downloaded the source, you can navigate to the root directory and run:

```
python setup.py install
```

For Python 3.6 and above

Users with Python 3.6 and above need to convert the downloaded source code scripts to Python 3 using the python package 2to3.

For example to convert *config.py* type:

```
2to3 -w config.py
```

Once all the python scripts are converted to Python 3, run:

```
python setup.py install
```

1.1.4 Test the installation

If the installation went smoothly, you should run a unit tests to ensure the package works as expected. The simplest way to do this is inside a python shell:

```
import bayesvp
from bayesvp.tests import run_tests
```

The output should look something like this:

```
The output should look something like this:

test_config_file_exists (bayesvp.tests.test_config.TCConfigFile) ... ok
test_default_no_continuum_params (bayesvp.tests.test_config.TCConfigFile) .
↪.. ok
test_example_mcmc_params (bayesvp.tests.test_config.TCConfigFile) ... ok
...
test_prior (bayesvp.tests.test_likelihood.TCPosterior) ... ok
test_general_intensity (bayesvp.tests.test_model.TCSingleVP) ... ok
test_simple_spec (bayesvp.tests.test_model.TCSingleVP) ... ok

----------------------------------------------------------------------
Ran 13 tests in 3.654s

OK
```

Test run with **no error** would ensure that `bayesvp` is successfully installed.

CHAPTER

TWO

DOCUMENTATION

2.1 Absorption line fit using `bayesvp`

We describe an example on how to fit a simple absorption line using `bayesvp`. The fitting routine can run in an interactive python session or using a python script.

To start `bayesvp`, first we need to setup a configuration file. `bayesvp` is meant to run with this file in background as it can take few minutes for MCMC sampling depending on the chosen number of walkers, steps, and parallel processes.

Below we illustrate a simple interactive use of the code and setting up of a configuration file to fit an O VI transition with rest wavelength of 1031.926 Å.

2.1.1 Setup a config file

First step is to import `bayesvp` package. Next we import an object, `bvp_write_config`, that interactively asks the user a few questions to create a config file.

```
In [1]: import bayesvp

In [2]: from bayesvp.scripts import bvp_write_config as wc
```

Let us assume that the spectrum in question is located in the following directory:

```
In [3]: spectrum_path = '/home/<username>/bayesvp_tutorial/codes/examples/'
```

Suppose the file name of the example spectrum is `OVI.spec` with three of columns of data: wave, flux, error. We can use the `bvp_write_config` routine to set up the config file as:

```
In [4]: config_writer = wc.WriteBayesVPConfig().print_to_file("-i")
```

On executing the above command, the routine will ask the user few questions:

The questions are on the left, while on the right, as an example, the answers are written. These answers can change based on different user preferences.

`Path to spectrum:` /home/<username>/bayesvp_tutorial/codes/examples

`Spectrum filename:` OVI.spec

`filename for output chain:` o6

Enter the name of the ion to be fitted: first atom name, next its ionization state

`atom:` O

`state:` VI

Enter the number of components to used in the fitting routine. If you wish to use more components then type in the number

`Maximum number of components to try:` 1

Enter the observed wavelength region to be used in the fitting routine

`Starting wavelength(A):` 1030

`Ending wavelength(A):` 1033

`Enter the priors:`

Enter the assumed minimum and maximum values of column density for the absorption feature to be fitted

`min logN [cm^-2] =` 10

`max logN [cm^-2] =` 18

Similarly, enter the assumed minimum and maximum values of Doppler b-parameter for the absorption feature to be fitted

`min b [km/s] =` 0

`max b [km/s] =` 100

Enter the central redshift of the absorption feature corresponding to the ion to be fitted

`central redshift =` 0

`velocity range [km/s] =`300

Here we enter parameters to be used for MCMC sampling

`Enter the MCMC parameters:`

`Number of walkers:` 400

`Number of steps:` 2000

`Number of processes:` 8

`Model selection method bic(default),aic,bf:` bic

`MCMC sampler kombine(default), emcee:` kombine

On completion of the above step, the configuration file is automatically written within a sub-directory where the spectrum is located.

`Written config file:` /home/<username>/bayesvp_tutorial/codes/examples/bvp_configs/config_OVI.dat

The saved configuration file 'config_OVI.dat' will look like the following:

```
spec_path /home/<username>/bayesvp_tutorial/codes/examples
output o6
mcmc 400 2000 8 bic kombine
%% OVI.spec 1030.000000 1033.000000
% O VI 15 30 0.000000
logN 10.00 18.00
```

(continues on next page)

(continued from previous page)

```
b     0.00 100.00
z     0.000000 300.00
```

For HST/COS database

To deal with HST/COS data, which requires incorporation of line-spread-functions (LSFs), the above saved **config_OVI.dat** needs to be modified as such:

```
spec_path /home/<username>/bayesvp_tutorial/codes/examples
output o6
mcmc 400 2000 8 bic kombine
%% OVI.spec 1030.000000 1033.000000
% O VI 15 30 0.000000
lsf COS_res<central_wavel>.dat
logN 10.00 18.00
b     0.00 100.00
z     0.000000 300.00
```

Note the changes made above. An additional statement: `lsf <LSF kernel at the central wavelength>` *is added.*

The HST/COS LSFs should be saved in a sub-directory (named **database**) where the spectrum is located.

2.1.2 Run a MCMC fit

To start MCMC fit, firstly we need to import three more objects:

```
In [5]: from bayesvp.config import DefineParams

In [6]: from bayesvp.mcmc_setup import bvp_mcmc_single as mc_single

In [7]: from bayesvp.mcmc_setup import bvp_mcmc as mcmc
```

`bayesvp` can be run by supplying the full path to the config file as a command line argument. `bayesvp` will print to screen the relevant information from the config file