UNIVERSITY OF CENTRAL FLORIDA

BART Users Manual

BAYESIAN ATMOSPHERIC RADIATIVE TRANSFER

Authors:
Patricio CUBILLOS
Jasmina BLECIC
Ryan CHALLENER

Supervisor:
Dr. Joseph HARRINGTON

November 8, 2020

Contents

1 Team Members

- Patricio Cubillos[1], University of Central Florida (pcubillos@fulbrightmail.org).
- Jasmina Blecic, University of Central Florida (email).
- Joseph Harrington, University of Central Florida (email).
- Patricio M. Rojo, Universidad de Chile (email).
- M. Oliver Bowman, University of Central Florida (email).
- Madison Stemm, University of Central Florida (email).
- Andrew S. D. Foster, University of Central Florida (email).
- Ryan Challener, University of Central Florida (rchallen@knights.ucf.edu).
- A.J. Foster, University of Central Florida (email).
- Michael D. Himes, University of Central Florida (mhimes@knights.ucf.edu).
- People, Institution (email).

2 Introduction

This document describes the University of Central Florida's computer program BART: Bayesian Atmospheric Radiative Transfer. BART uses a Bayesian approach to retrieve molecular abundances and temperature profiles from exoplanet atmospheres for given spectro-photometric data. The BART package include three stand-alone submodules: the Thermochemical Equilibrium Abundances (TEA) to calculate species equilibrium mole mixing ratios; the radiative-transfer solver Transit, and the statistical package MCcubed.

The detailed BART documentation, The BART theory documents (FINDME), BART User Manual[2], and BART Code Documentation are provided with the package.

This manual describes the BART program usage, inputs, and outputs. Complementary, TEA, Transit, and MCcubed with its documentation.

This manual makes many references to the Transit[3] and TEA user manuals **(FINDME: add url)**.

Section 3 indicates how to obtain the code. Section 7 blah. Section 8 blah. blah blah ...

2.1 BART Package Overview

BART is a Python and C programming language code written in a modular, object oriented style. The Bayesian Atmospheric Radiative Transfer (BART) project combines the radiative-transfer code (Transit) with the Thermochemical Equilibrium Abundances (TEA) module, which calculates

[1] https://github.com/pcubillos/

[2] Most recent version of the manual available at https://exosports.github.io/BART/doc/BART_User_Manual.html

[3] Most recent version of the manual available at https://exosports.github.io/transit/doc/Transit_User_Manual.html

abundances of gaseous species, and the Multi-Core Markov-chain Monte Carlo (MCcubed) statistical module, to assess the temperature and molecular-abundances posterior distributions, given a set of observations.

Note that each one of the submodules (TEA, Transit, and MCcubed) are stand-alone routines that can be used separately on their own for a variety of projects, not necessarily linked to exoplanet atmospheric retrieval.

2.1.1 Thermochemical Equilibrium

2.1.2 Bayesian Statistics

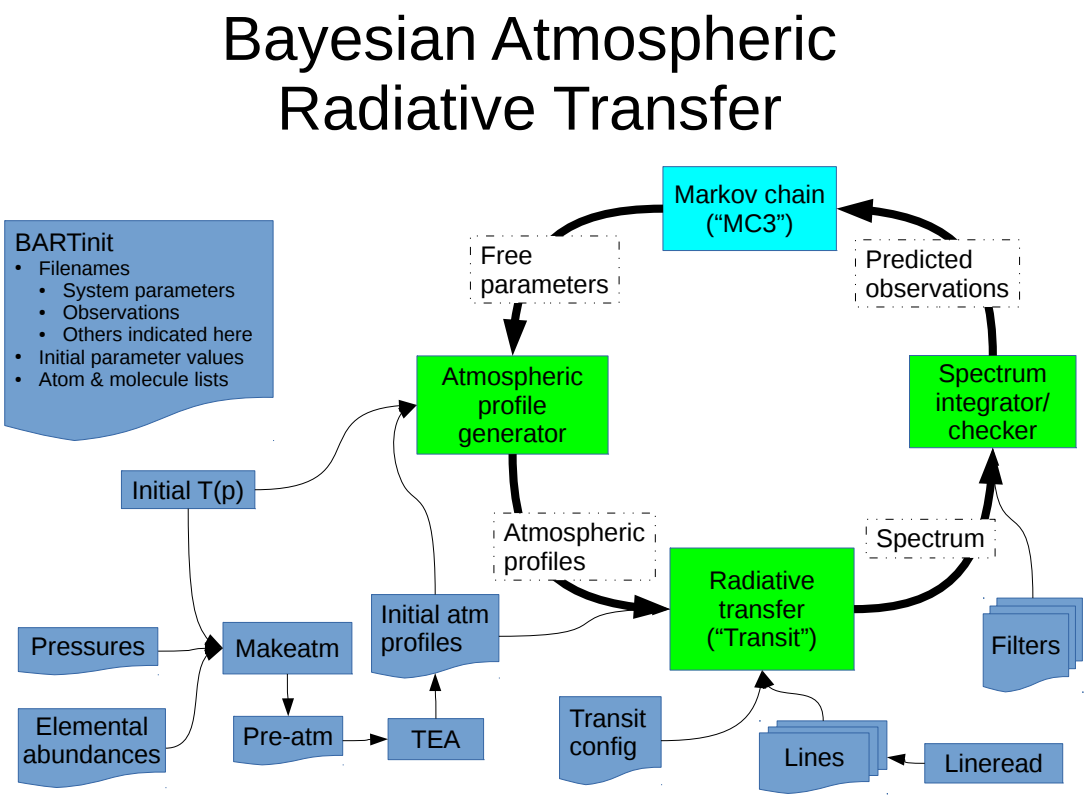
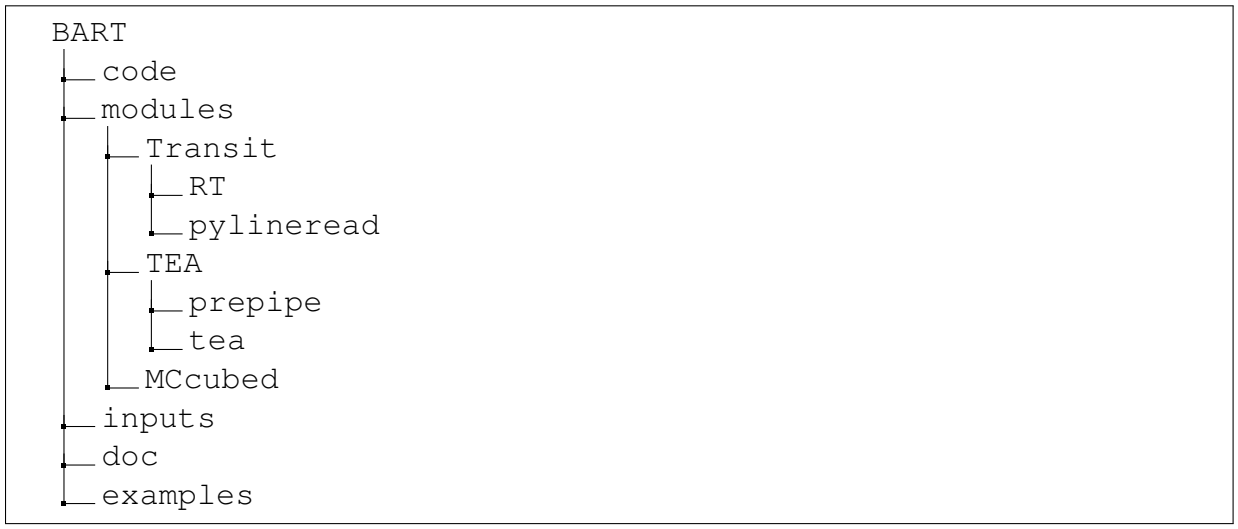


Figure 1: The BART chart. **(FINDME: explain)**. **(FINDME: Remove title)**.

The BART package is organized as follows:



2.2 License

Bayesian Atmospheric Radiative Transfer (BART), a code to infer properties of planetary atmospheres based on observed spectroscopic information.

This project was completed with the support of the NASA Planetary Atmospheres Program, grant NNX12AI69G, held by Principal Investigator Joseph Harrington. Principal developers included graduate students Patricio E. Cubillos and Jasmina Blecic, programmer Madison Stemm, and undergraduates M. Oliver Bowman and Andrew S. D. Foster. The included 'transit' radiative transfer code is based on an earlier program of the same name written by Patricio Rojo (Univ. de Chile, Santiago) when he was a graduate student at Cornell University under Joseph Harrington. Statistical advice came from Thomas J. Loredo and Nate B. Lust.

Patricio Cubillos <pcubillos[at]fulbrightmail.org>
Jasmina Blecic <jasmina[at]physics.ucf.edu>
Joseph Harrington <jh[at]physics.ucf.edu>

or alternatively,

Joseph Harrington, Patricio Cubillos, and Jasmina Blecic
UCF PSB 441
4111 Libra Drive
Orlando, FL 32816-2385
USA

Thank you for using BART!

3 Installation

3.1 System Requirements

BART was developed on a Unix/Linux machine using Python 2.7.6, Numpy 1.8.2, Matplotlib 3.0.2, and mpi4py 1.3.1.
– Python 2.7.6+.
– NumPy 1.8.2+.
– matplotlib 3.0.2+.
– SymPy (version 0.7.1.rc1 to ensure better performance)
– mpi4py 1.3.1+.
– An MPI distribution (MPICH preferred).
– A C compiler (e.g., gcc).
– A Fortran compiler (e.g., gfortran).

3.2 Install and Compile

To obtain the latest stable version from the BART releases page **(TBD)**. Alternatively, clone the repository to your local machine with the following Linux terminal commands. First create a top-level directory to place the code:

```
mkdir BART_demo/
cd BART_demo/
topdir=`pwd`
```

(FINDME: Backtick marks don't work when copy-pasting text to the terminal.)

Clone the repository with all its sub-modules:

```
git clone --recursive https://github.com/exosports/BART $topdir/BART/
```

Navigate to the directory, build, and activate the conda environment:

```
cd $topdir/BART/
conda env create -f environment.yml
conda activate bart
```

Compile the transit module programs:

```
cd $topdir/BART/modules/transit/
make
```

Compile the `MCcubed` routines:

```
cd ../MCcubed/
make
```

To remove the program binaries, execute (in the respective directories):

```
make clean
```

Note that there may be warnings.

4 Quick Example

The following script lets you quickly fit a methane emission spectrum model to a set of 10 filters between 2 and 4 μm. These instructions are meant to be executed from the shell Linux terminal. To begin, follow the instructions in the previous Section to install and compile the code. Now, create a working directory in your top directory to place the files and execute the programs:

```
cd $topdir
mkdir run/
cd run/
```

Download the methane line-transition database from the HITRAN server:

```
wget --user=HITRAN --password=getdata -N https://www.cfa.harvard.edu/HITRAN/
   HITRAN2012/HITRAN2012/By-Molecule/Compressed-files/06_hit12.zip
unzip 06_hit12.zip
```

Copy the pylineread configuration file and run pylineread to make the transition-line-information (TLI) file:

```
cp $topdir/BART/examples/demo/pyline_demo.cfg .
$topdir/BART/modules/transit/pylineread/src/pylineread.py -c pyline_demo.cfg
```

Copy the transit configuration file and run it to make a table of opacities:

```
cp $topdir/BART/examples/demo/transit_demo.cfg .
$topdir/BART/modules/transit/transit/transit -c transit_demo.cfg --justOpacity
```

Copy the BART configuration files and run the MCMC code to sample the posteriors:

```
cp $topdir/BART/examples/demo/BART_eclipse.cfg .
$topdir/BART/BART.py -c BART_eclipse.cfg
cp $topdir/BART/examples/demo/BART_transit.cfg .
$topdir/BART/BART.py -c BART_transit.cfg
```

5 Examples

5.1 WASP-12b

The following commands will run a test case for BART, based on eclipse depths from Spitzer observations. This test is intended to be used to check that new additions to the BART code have not broken it. Keep in mind that the MCMC algorithm has an element of randomness and thus the output will not be identical every time, but with enough iterations and/or chains, differences should be small.

Navigate to the correct directory:

```
cd $topdir/run/
```

Run these scripts to download and unzip the following molecular line-list files:

1. H_2O from HITEMP:

```
cp ../BART/examples/WASP-12b/wget-list_HITEMP-H2O.txt
wget --user=HITRAN --password=getdata -N -i wget-list_HITEMP-H2O.txt
unzip '01_*HITEMP2010.zip'
```

2. CO_2 from HITEMP:

```
cp ../BART/examples/WASP-12b/wget-list_HITEMP-CO2.txt
wget --user=HITRAN --password=getdata -N -i wget-list_HITEMP-CO2.txt
unzip '02_*HITEMP2010.zip'
```

3. CO from HITEMP:

```
wget --user=HITRAN --password=getdata -N https://www.cfa.harvard.edu/
   HITRAN/HITEMP-2010/CO_line_list/05_HITEMP2010.zip
unzip 05_HITEMP2010.zip
```

4. CH_4 from HITRAN:

```
wget --user=HITRAN --password=getdata -N https://www.cfa.harvard.edu/
   HITRAN/HITRAN2008/HITRAN2008/By-Molecule/Compressed-files/06_hit08.zip
unzip 06_hit08.zip
```

Copy the pylineread configuration file and run pylineread to make a transition-line-information (TLI) file:

```
cp $topdir/BART/examples/WASP-12b/pyline.cfg .
$topdir/BART/modules/transit/pylineread/src/pylineread.py -c pyline.cfg
```

Copy the Transiting Exoplanet (TEP) file from the examples directory:

```
cp $topdir/BART/examples/WASP-12b/WASP-12b.tep ../BART/inputs/tep/WASP-12b.tep
```

Download the stellar Kurucz file:

```
wget -N kurucz.harvard.edu/grids/gridp03/fp03k2.pck -O ../BART/inputs/kurucz/wasp12b-fp03k2.
```

Copy the BART configuration file and run BART. This will first generate a table of opacities and then run the MCMC code to sample the posteriors:

```
cp $topdir/BART/examples/WASP-12b/BART.cfg .
$topdir/BART/BART.py -c BART.cfg
```

This takes several hours on a 10-core machine, and considerably longer on machines with fewer than 10. The best-fit spectrum should look comparable to figure 2.

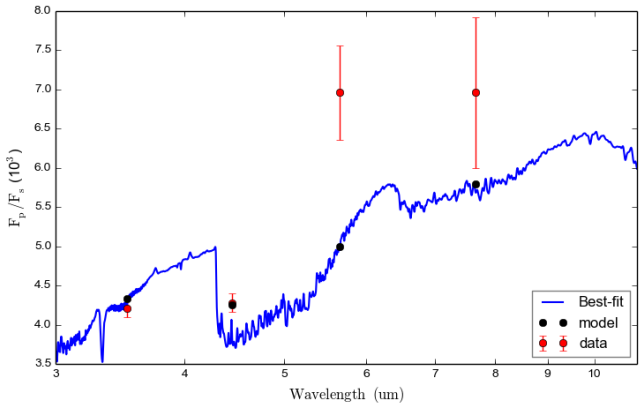


Figure 2: Best-fit spectrum of WASP-12b Spitzer data.

6 Walkthrough

7 Program Inputs

The executable BART.py is the main BART program to perform most calculations. This program can take command-line arguments or a configuration file (preferred) to set up the calculation details.

Most of the BART inputs can either be given as pre-existing files or can be produced upon execution. BART requires the following input files to run:

7.1 BART Configuration File

The BART configuration file is the main file that sets the arguments for a BART run. The arguments follow the format `argument = value`, where `argument` is any of the possible command-line arguments. While all options can be set from the command line, the number of required arguments makes using a configuration file much easier. Note that some of these argument inputs are files that should exist before running BART.

(FINDME: Speak about the folder structure).

7.2 Transit-Line Information (TLI) File

Transit Line-Information (TLI) files are created by the Pylineread routine placed in the BART/-modules/transit/ sub-module. They contain information about the emission and absorption lines of molecules found in the exoplanet's atmosphere which ais necessary to calculate opacity. The file contains the following for each line:

- molecule isotope
- central wavenumber
- oscillator strength
- lower-state energy

To run this module and produce a transit linelist input (TLI) file do the following: Starting from the top directory, navigate to the pylineread module

```
cd BART/modules/transit/pylineread
```

Inside the pylineread directory, make a TLI sub-directory and enter it.

```
mkdir TLI
cd TLI
```

Copy the pylineread example configuration file.

```
cp ../examples/pyline_example.cfg pyline_run01.cfg
```

Read the configuration file header and download the desired databases with the partition functions from the provided links.

Edit the configuration file with the following information:

- Paths to the databases
- Types of the input databases
- Desired output TLI filename
- Desired initial and final wavelength range in microns
- Verbosity level

Run the pylineread module.

```
../src/pylineread.py -c pyline\_run01.cfg
```

This will produce the TLI file with the requested name. If unspecified, the file will be placed in the current working directory (/TLI). Otherwise, it will be place in the specified directory. A more in-depth description of the TLI files and the pylineread module can be found in the Transit User Manual.

7.3 Cross-section (CS) File

Cross-section (CS) files contain information for any opacity continuum. They must be in HITRAN cross-section format. Several Collision-Induced Absorption (CIA) files are included in the BART/-modules/transit/inputs/ directory. These may be used as CS files and are named according to the molecular collisions they represent.

CS files are handled by `Transit` and thus a more in-depth description can be found in the Transit User Manual.

7.4 Transiting Extrasolar Planet (TEP) File

Transiting Extrasolar Planet (TEP) files contain information about the particular exoplanet in question. The file includes many orbital geometry parameters necessary for spectral calculation. Some TEP files can be found in the BART/modules/transit/inputs/tep/ directory.

TEP files are handled by `Transit` and thus a more in-depth description can be found in the Transit User Manual.

7.5 Stellar Spectrum File

The stellar spectrum is provided by a Kurucz stellar spectrum file. They are dependent on the star's metallicity and can be found on Robert Kurucz's website. Some Kurucz files are provided in the BARt/inputs/kurucz directory.

Kurucz stellar spectrum files are handled by `Transit` and thus a more in-depth description can be found in the Transit User Manual.

7.6 Elemental Abundances File

The elemental abundances file contains solar abundances of many elements (and some isotopes), along with atomic number, atomic symbol, atomic name, and molar mass. This file is used by TEA to calculate thermochemical equilibrium abundances. The solar abundances as found by Asplund et al, 2009 are found in the BART/inputs/abundances_Asplund2009.txt file.

7.7 Filter Files

The filter files tell BART how to integrate over the `transit` output spectrum to compare with the given data points. They consist of two columns: wavenumber and filter efficiency. BART will interpolate the spectrum to the filter wavenumbers and multiply the spectrum by the filter before integrating over the bandpass.

Several *Spitzer* filter files are provided in the BART/inputs/filters directory.

7.8 Command-Line Arguments

The user provides the BART configuration arguments in a configuration file. The configuration file follows the ConfigParser format. See, e.g., the example below:

```
[MCMC]
# Output directory:
loc_dir = HD209458b

# Elemental species:
in_elem  = H He C N O

# Eclipse-depth data:
data   = 0.00119       ; Spitzer 3.6 microns
         0.00123       ; Spitzer 4.5 microns
```

The [MCMC] line defines a section (do not edit this line). Argument entries are input in the format: **'parameter = value'**. Multiple valued arguments are set either with blank spaces or in separated lines. Blank lines are allowed. Lines beginning with the **'#'** or the **';'** characters are comments and are ignored by the code. The **';'** character can also be use as an in-line comment.

7.8.1 General Arguments

`-h, --help`
Show the help message and exit.

`-c FILE, --config_file FILE`
Configuration filename (string).

`-v INT, --verbose-level VERB`
Verbosity level (integer).

`--justTEA`
Run only TEA.

`--justOpacity`
Run just Transit and quit after generating opacity table.

`--resume`
Resume a previous run.

`--tint FLT`
Internal temperature of the planet.

`--filter FILE`
Filter file names (corresponding to data).

`--kurucz_file FILE`
Kurucz file name.

`--solution STR`
Solution type (transit or eclipse).

7.8.2 Atmospheric Pressure Layers

`--n_layers INT`
Number of atmospheric layers.

`--p_top DBL`
Pressure at the top of the atmosphere (bars).

`--p_bottom DBL`
Pressure at the bottom of the atmosphere (bars).

`--log BOOL`
Use log (True) or linear (False) scale sampling.

`--press_file FILE`

Input/Ouput file with pressure array.

7.8.3 Atomic Abundances

`--abun_basic FILE`
Input elemental abundances file.

`--abun_file FILE`
Inpu/Output modified elemental abundances file.

`--solar_times INT`
Multiplication factor for metal abundances.

`--COswap BOOL`
Swap Carbon and Oxygen abundances if True.

7.8.4 Temperature Profile

`--PTtype STR`
Temperature profile model ('line' or 'madhu')

`--PTinit DBL`
1D array of initial temperature profile model parameters.

7.8.5 Atmospheric File (TEA) Arguments

`--in_elem STR`
1D array of input elements.

`--out_spec STR`
Output species to include in the atmospheric model

`--preatm_file FILE`
Pre-atmospheric file name (with elemental abundances per layer.

`--atmfile FILE`
Atmospheric model file (output).

`--uniform DBL`
If not None, 1D array of uniform abundances for each species in out_spec.

`--refpress FLT`
Reference pressure level (bars) corresponding to the planet radius.

7.8.6 MCMC Arguments

`--params DBL`
Initial model-fitting parameters.

`--molfit STR`

1D string array of molecules to fit.

`--Tmin FLT`
Lower temperature boundary (K).

`--Tmax FLT`
Upper temperature boundary (K).

`--quiet`
Set verbosity to minimum.

`--stepsize FLT`
1D array of parameter step sizes.

`--burnin INT`
Number of burn-in iterations per chain.

`--data FLT`
1D array of eclipse/transit depths.

`--uncert FLT`
1D array of data uncertainties.

7.8.7 Transit Arguments

`--tconfig FILE`
Transit configuration file.

`--opacityfile FILE`
Opacity table file (created if nonexistant).

`--outflux FILE`
Ouput file with flux values (for eclipse geometry).

`--outmod FILE`
Output file with modulation values (for transit geometry).

`--shareOpacity BOOL`
If True, use shared memory for the Transit opacity file.

7.9 Configuration File

8 Program Outputs

BART submodules produce the following outputs:

* TEA produces one output file:

    - 1. atmopsheric_file.tea. This file and the BART modified version are placed in the BART output folder

* MCcubed produces one output file:
    - 1. output.npy - a binary file containing 3D array of parameters per each chain and each iteration
* `Transit` produces three output files:
    - 1. _toomuch.dat - list of radii at each wavelength sample where the optical depth reaches the maximum 'toomuch' value
    - 2. _sampling.dat - provides sampling information
    - 3. _spectrum.dat - gives the flux/modulation (eclipse/transit geometry) vs. wavelength
* BART produces the following output files:
    - 1. output files:
        - band_eclipse.npy - a binary file containing 3D array of eclipse depths per each chain and each iteration
    - 2. plots:
        - best fitting T-P profile with boundaries
        - best fitting abundances
        - contribution functions
        - normalized contribution functions
        - posterior histograms
        - posterior correlation plots
        - spectrum

9 Running BART

From the top directory (location of BART), make a current working directory and move into it:

```
mkdir run
cd run
```

In the BART/examples/ folder, an example of the BART configuration file is provided. Copy the BART.cfg example file to the run/ directory:

```
cp BART/examples/BART.cfg BART.cfg
```

Edit the file with the correct information:

1. Give paths to the input tep and kurucz files and the output directory
2. ::: Atmospheric pressure layers ::: Provide the information needed to make a pressure file with desired number of layers, range and scale.
3. ::: Elemental-abundances file ::: Provide the name of the basic abundances files, information how it needs to be changed, and the desired name of the final abundances file.

4. ::: Temperature profile ::: There two parametrized temperature-pressure profiles that BART can work with: Madhu (based on Madhusudhan adn Seager 2009) and Line (based on Line et al 2013). Provide the model type and its initial parameter values.

5. ::: Atmospheric Elemental Abundances (pre-atmospheric) File ::: Provide the name of the desired pre-atmospheric file that will contain the temperature-pressure and elemental abundances information for each layer in the atmosphere, and desired elemental and molecular species. Note: the output species names must use naming convention produced by the ../modules/TEA/ module.

6. ::: Atmospheric File (P, T, species-abundances) ::: Provide the desired name of the final atmospheric file, that will be used as input for ../modules/transit/ module.

7. ::: MCMC arguments ::: Provide information to run ../modules/mccubed/ module:

    - eclipse/transit depths and their uncertainties
    - path to the filter files
    - 3-element tuple to locate the fitting function (function_name routine_name path_to_routine)
    - list of molecules that have line list data
    - temperature boundaries (this information depends on the line-list and CS databases temperature range)
    - model fitting parameters (temperature-pressure and abundances scale factors).
    - set basic MCMC variables and define the filename to store model fit
    - set desired verbosity level

8. ::: Transit variables ::: Provide information to run ../modules/transit module:

    - set the desired name of the ../modules/transit/ module configuration file that will be produced based on the information provided in this section
    - locate TRANSIT line information and CS files
    - spectrum information (Note: The range given must be within the wavelength range given for the TLI file creation)
    - the desired geometry solution (transit or eclipse), max optical depth, and number of HWHM for Voight profile calculation
    - planetary surface gravity and reference radius and pressure
    - opacity lines' strength threshold
    - information to make an opacity-grid file.
        i. If variable values and new file name is provided, the opacity grid file will be made. This information must include the temperatures from the atmospheric file. Execution takes couple of hours.
        ii. If the opacity file already exists give the path to it. Other variables are ignored. Execution will be successful only if the pressure and wavenumber samplings are exactly the same as in the atmospheric and TLI file, respectively. Execution takes around 15 seconds.

iii. If opacityfile name is commented, the opacity calculation will be done on spot. Execution takes couple of minutes.

- provide the desired names of the output files

9.1 Real-Case Run

This example shows how to approach a real-life BART run. In this case you don't want to use demo files (Section 4), because it uses many defaults that need to be specified for a given analysis. See below a step-by-step guide to analyze a planet.

1. Clone the BART repository to your working directory (See Section 3.2).

2. Copy this BART configuration file: BART/examples/BART_example.cfg to your run folder. Start editing the arguments in the configuration file.

3. Determine what observing geometry you will test, and set **solution** to eclipse or transit accordingly. Based on that, search for the available data (eclipse or transit depth, respectively) from the literature or your own data analysis. Put those values in the **data** and **uncert** arguments.

4. For each data point, you will need to search for/make the instrumental transmission filter file (See Section 7.7). Most observatories provide the filter files, but make sure that the file has the right format (Sec. 7.7), edit if needed. If you use spectroscopic data (e.g., HST's WFC3 data), you will have to reproduce the data analysis of the observer (downsampling, spectral binning). Set the path to the filter files in the **filter** argument.

5. Find/make a TEP file for your planet (See Section 7.4). In practice, the only parameters that BART requires are the stellar temperture, the stellar radius, the semi-major axis, the planetary radius, the planetary mass, and the stellar $\log(g)$. This information can be typically found in exoplanets.org/ or exoplanet.eu/. Set the path to the file in **tep_name**.

6. Find a Kurucz stellar file (See Section 7.5). Follow instructions in /home/esp01/doc/info_kurucz. **(FINDME: Copy those instruction here or in Sec 7.5)** Set the path to the Kurucz file in **kurucz**.

7. Run pylineread to make a TLI file for the species of interest (See Section 7.2). Set the path to the TLI file in **linedb**.

8. Determine if you want to load an existing atmospheric file or create a new one. If the former, set **atmfile** to an existing file (and skip to step (14)) If you did not change any atmospheric parameters, loading an existing file can save several minutes of runtime.

9. Set the atmospheric pressure layers with the **press_file**, **n_layers**, **p_top**, **p_bottom**, and **log** arguments.

10. Set the initial atmospheric temperature profile. Set with `PTtype` if you want to use a model as in **?** or **?**. Adjust the model parameters in **PTinit** to set the temperature profile (See Section **??**). What is an appropriate initial temperature profile? That is a question yet to be answered. Having no prior knowledge on the planet's properties, temperatures around the

equilibrium temperature would be an initial guess. As a help, run the scripts in BART/scripts/quickguide.py to see how the temperature profile looks for a given set of model parameters. If TEA is creating a new atmospheric file it will allow you to inspect the profile before continuing.

11. Determine the species present in the atmosphere by setting the **out_spec** argument. Follow the instruction in Section **??**.

12. Determine if you want to start with an atmosphere in thermochemical equilibrium or with uniform species abundances. If the former, BART will calculate the abundances with the TEA module. Determine the elemental abundances by adjusting the **abun_basic**, **solar_times**, **COswap**, **abun_file**, **preatm_file**, and **in_elem** arguments (See Section **??**).

13. If the user chooses to use vertical-uniform abundance profiles. Set in the **uniform** argument the desired species abundances (the order must match that of the **out_spec**).

14. Set in **molfit** the species whose abundances you want to fit.

15. Set in **Tmin** and **Tmax** the minimum and maximum boundaries of the temperature profiles to sample. These boundaries should not lie beyond the limits set by the TLI and CS data.

16. The **params** argument sets the initial guess for the list of fitting parameters. This list corresponds (in this specific order) the list of the temperature-model parameters (defined by **PTtype**), the planetary 'surface' level (only for transit geometry), and the species abundances (defined by **molfit**).

17. The **pmin** and **pmax** arguments set the fitting parameter-phase space boundaries.

18. The **stepsize** determines the initial stepsize and which fitting parameters are free or fixed (See Section **??**).

19. See Section **??** to lear about the rest of MCMC arguments.

20. wllow and wlhigh set the spectrum boundaries. What's important to mention is that we are limited by the boundaries in the CS files. They go from $\sim$0.6 um to 400um. Thus, for the moment don't go shorter than 0.6um. Also, the shorter the wllow value, the computation time increases (more than linearly (the spectrum is equi-spaced in wavenumber)), so stay as close to the shortest filter boundary as possible.

21. For the rest of the wavelength/wavenumber arguments, the defaults are ok. Again, ask me if the descriptions are not clear.

22. The tlow, thigh, temp-delt, opacityfile arguments make the opacity file.

23. If you do transit geometry instead of eclipse, you will need to change outflux to outmod.

See the transit documentation to learn more about all the Transit settings. Keep in mind that it is undergoing constant change, but most of the info is current.

9.2 Before Running

Before running the BART code, we need to generate the required input files. A Transit-Line-Information (TLI) file and an Opacity-Grid file.

9.2.1 TLI file

These are quick guidelines to generate a TLI file with the Transit's pylineread module. For a detailed description see the Transit User Manual. The following lines are intended to be run from the terminal.

From the BART directory, make a TLI directory and move into it:

```
mkdir TLI
cd TLI
```

Copy a pylineread configuration-file template:

```
cp ../modules/transit/pylineread/examples/pyline_example.cfg ./
   pyline.cfg
```

Edit the configuration file and run it:

```
../modules/transit/pylineread/src/pylineread.py -c ./pyline.cfg
```

Runtime can vary from a few minutes to an hour, depending on the number of line databases included. At the end, a TLI file will be saved in the working directory with the specified name.

9.2.2 Cross-section file

See the Transit user's manual to set up a Cross-section file.

9.2.3 Atmospheric file

Should I start with the chicken or the egg?
(FINDME: Something like (to run just the atmosphere part):)
`../BART.py -atmosphere`

(FINDME: jh is not happy with the file extensions.)

9.2.4 Opacity file

(FINDME: If BART doesn't find these files, the code should created them before running.)

These are quick guidelines to generate an Opacity-grid file with Transit. For a detailed description see the Transit user's manual.

Create a directory to run Transit and place the Opacity file:

```
cd ../
mkdir Opacity
```

```
cd Opacity
```

Copy a Transit configuration-file template:

```
cp ../modules/transit/run/config_sample.cfg ./opacity.cfg
```

Edit the configuration file and run it:

```
../modules/transit/transit/transit -c opacity.cfg
```

(FINDME: It would be nice to run Transit from BART, something like:)

```
../BART.py -opacity
```

9.2.5 TEA setup

Copy the BART and TEA configuration-file templates from the examples directory into your running dir, e.g.:

```
cp ../myBART/examples/BART.cfg .
cp ../myBART/examples/TEA.cfg .
```

Edit the parameters in the TEA configuration file. **(FINDME: Set this up from the BART config file)**

```
# === Full path to TEA package ===
location_TEA = ../modules/TEA/

# === Full path to abundances file ===
abun_file = ../modules/TEA/lib/abundances.txt

# === Full path to TEA working directory ===
location_out = ./HD209458b/
```

Even though transit has many command-line arguments, only a few of them are strictly necessary for a run. See for example the file in `mytransit/examples/example03.cfg`. **(FINDME: For Jasmina: see /home/patricio/ast/esp01/bart/transit/develop/examples/example03.cfg)**

9.2.6 BART setup

BART can be run from any folder, provided the user gives the path to the executable. We recommend to create a folder exclusively dedicated to run and hold the transit results to keep the results organized, e.g.:

```
cd myBART/
mkdir run
cd run/
```

The BART code consists of an initialization step and the proper MCMC run on the spectrum modeling.

10 Be Kind

Please cite these papers if you found this package useful for your research:

- Cubillos et al. (2015) (in preparation).
- Blecic et al. (2015) (in preparation).
- Harrington et al. (2015) (in preparation).

Thanks!