BayesCMD Documentation

Release

Joshua Russell-Buckland

Oct 12, 2017

CONTENTS:

BayesCMD is a package intended to expand the capabilities of the Brain/Circulation Modelling (BCMD) framework. It introduces the ability to obtain posterior distributions for model parameters by using Approximate Bayesian Computation (ABC).

CHAPTER

ONE

BCMDMODEL

1.1 Running the BCMD Model

The BCMD model can be run in a number of ways, both using the command lines and the WeBCMD interface. Over time, both the BayesCMD package and the WeBCMD package are expected to merge. As a result, the BCMD model class has been designed to allow flexibility and compatibility with both the current BayesCMD framework and the WeBCMD framework.

1.1.1 bcmd_model

Configure and run a BCMD model.

The BCMD Model class is used to configure and run a BCMD model. It is done by passing a number of configuration variables, creating a model input file and then running the model. Models can also be run from a pre-existing input file and input files can be written to file for later access also.

bayescmd.bcmdModel.bcmd_model.**TIMEOUT**
: `int` – Max number of seconds for a simulation to run before being cancelled. Default is 30 seconds.

bayescmd.bcmdModel.bcmd_model.**BASEDIR**
: `str` – Path to Base directory, which should be 'BayesCMD'. It is found by running the *bayescmd.util.findBaseDir* method, passing either an environment variable or a string to the method.

class bayescmd.bcmdModel.**ModelBCMD**(*model_name*, *inputs=None*, *params=None*, *times=None*, *outputs=None*, *burn_in=999*, *create_input=True*, *input_file=None*, *suppress=False*, *workdir=None*, *deleteWorkdir=False*, *timeout=30*, *basedir='..'*, *debug=False*, *testing=False*)

Use to create inputs, run simulations and parse outputs.

Parameters

- **model_name** (`str`) – Name of model. Should match the modeldef file for model being generated i.e. model_name of 'model'' should have a modeldef file 'model1.modeldef'.
- **inputs** (`dict` or `None`, optional) – Dictionary of model inputs and their values. Has form {'names' : `list` of `str`, 'values' : `list` of `list` of `float`} where *names* should be a list of each model input name, matching up to the model inputs and *values* would be a list of lists, where each sublist is the input values for that time point. With this in mind, the length of *inputs['values']'* should equal length of *times*. Default is None.
- **params** (`dict` of `str`: `float` or `None`, optional) – Dictionary of {'parameter': param_value}. Default is None

- **times** (list of float or int or None, optional) – List of times at which measurement data has been collected and needs to be simulated. Default is None.
- **outputs** (list of str or None, optional) – List of model outputs to return. Default is None.
- **burn_in** (float or int, optional) – Length of burn in period at start of the simulation. Default is 999
- **create_input** (boolean, optional) – Boolean indicator as to whether an input file needs creating. Default is True.
- **input_file** (str or None, optional) – Path to existing input file or to where one needs creating. Default is None.
- **suppress** (boolean, optional) – Indicates if console output should be suppressed during model runs. This will prevent writing of both stderr and stdout. Default is False.
- **workdir** (str or None, optional) – Path to working directory if one exists. If not set, it will default to a temporary directory in 'tmp/'. If you wish to write input files and similar to file, it is recommended that the working directory is set manually by the user.
- **deleteWorkdir** (boolean, optional) – Indicates if the working directory should be deleted after finishing. Default is False.
- **timeout** (float or int, optional) – Maximum length in seconds to let model run before cancelling. Default is *TIMEOUT*.
- **basedir** (str, optional) – Path to base 'BayesCMD' directory. By default it is set to *BASEDIR*.
- **debug** (boolean, optional) – Indicates if debugging information should be written to console. Default is False.
- **testing** (boolean, optional) – If True, appends '_test' to coarse and detailed model output. Useful if you wish to test settings and want to avoid test results becoming mixed in with real result files. Default is False.

model_name
str – Name of model. Should match the modeldef file for model being generated i.e. model_name of 'model'' should have a modeldef file 'model1.modeldef'.

inputs
dict or None – Dictionary of model inputs and their values. Has form {'names' : list of str, 'values' : list of list of float} where *names* should be a list of each model input name, matching up to the model inputs and *values* would be a list of lists, where each sublist is the input values for that time point. With this in mind, the length of *inputs['values']'* should equal length of *times*. Default is None.

params
dict of str: float or None – Dictionary of {'parameter': param_value}. Default is None

times
list of float or int or None – List of times at which measurement data has been collected and needs to be simulated. Default is None.

outputs
list of str or None – List of model outputs to return. Default is None.

burn_in
float or int – Length of burn in period at start of the simulation. Default is 999

create_input
boolean – Boolean indicator as to whether an input file needs creating. Default is True.

input_file
str – Path to existing input file or to where one needs creating. If *create_input* is True and no path is given, the input file will be written to *workdir* as *model_name*.input.

suppress
boolean – Indicates if console output should be suppressed during model runs. This will prevent writing of both stderr and stdout. Default is False.

DEVNULL
_io.BufferedWriter – If *suppress* is set to True, this will be an io buffer that redirects stderr and stdout to the null device.

workdir
str or None – Path to working directory if one exists. If not set, it will default to a temporary directory in 'tmp/'. If you wish to write input files and similar to file, it is recommended that the working directory is set manually by the user.

If no working directory is given, the *deleteWorkdir* attribute will be set to True in order to ensure that the file space does not become excessively full during batch runs.

deleteWorkdir
boolean – Indicates if the working directory should be deleted after finishing. Default is False, but this will always be set to True if *workdir* is set to None.

timeout
float or int – Maximum length in seconds to let model run before cancelling. Default is *TIMEOUT*.

basedir
str – Path to base 'BayesCMD' directory. By default it is set to *BASEDIR*.

debug
boolean – Indicates if debugging information should be written to console.

program
str – Path to the compiled model file. This is expected to be in '*basedir*/build', with the name *model_name*.model.

output_{coarse,detail}
str – Location to write coarse and detailed output files to. This will be the working directory, with coarse output files having the suffix '.out' and detailed output files having the suffix '.detail'.

output_dict
collections.defaultdict(:obj:`list)' – Dictionary of output data.

_cleanupTemp()
Delete working directory.

create_default_input()
Create configured default input file and write to string buffer.

Using this method allows the configured input file to be written to memory, thus reducing the number of operations involving writing to disk.

create_initialised_input()
Create a custom, initialised input and write to buffer.

Create an input file using configuration values specified by *inputs*, *times*, *params* and *outputs*, and write it to the file specified by *input_file*.

The configured input file will be written to memory, thus reducing the number of operations involving writing to disk.

get_defaults()
Obtain default model configuration.

output_parse()
Parse the output files into a dictionary.

This allows the model output to be sent to JSON, if using WeBCMD, or simply processed in a more pythonic fashion for any Bayesian (or similar) analysis.

run_from_buffer()
Run the model using an input file written to memory.

The input file will need to have been created using `create_initialised_input` or `create_default_input`.

run_from_file()
Run model using an input file found at *input_file*.

The model will be run using an already created input file that has been written manually or using the `write_default_input` or `write_initialised_input` methods.

write_default_input()
Create and write a default input to file.

Create an input file using default configuration values and write it to the file specified by *input_file*. All inputs, outputs and parameters are set to default values and there is no burn in.

write_initialised_input()
Create and write a custom, initialised input to file.

Create an input file using configuration values specified by *inputs*, *times*, *params* and *outputs*, and write it to the file specified by *input_file*.

1.2 Input Creation

Input files are required by the BCMD model. A special class has been created that will create a correctly formatted input file for a variety of use cases.

1.2.1 input_creation

Create input files for use with a BCMD model.

Input files are needed in order to set model parameters and provide driving inputs.

class bayescmd.bcmdModel.**InputCreator**(*times*, *inputs*, *outputs=None*, *params=None*, *filename=None*)
Create an input file by passing relevant information to the class.

This input file is then used to create an input file that can either be written to file or kept in buffer and passed directly to the model.

Parameters

- **times** (`list` of `float` or `int`) – List of times at which measurement data has been collected and needs to be simulated.
- **inputs** (*dict*) – Dictionary of model inputs and their values. Has form {'names' : `list` of `str`, 'values' : `list` of `list` of `float`} where *names* should be a list of each model input name, matching up to the model inputs and *values* would be a list of lists, where

each sublist is the input values for that time point. With this in mind, the length of *inputs['values']'* should equal length of *times*.

- **filename** (str, optional) – Name of the input file to be written to if writing to file is required. Default is None.
- **params** (dict of str: float, optional) – Dictionary of {'parameter': param_value}
- **outputs** (list of str, optional) – List of model outputs to return.

times
list of float or int – List of times at which measurement data has been collected and needs to be simulated.

inputs
dict – Dictionary of model inputs and their values. Has form {'names' : list of str, 'values' : list of list of float} where *names* should be a list of each model input name, matching up to the model inputs and *values* would be a list of lsits, where each sublist is the input values for that time point. With this in mind, the length of *inputs['values']'* should equal length of *times*.

f_out
StringIO() – String buffer object to which the input file will be written.

filename
str – Name of the input file to be written to if writing to file is required. Default is None.

params
dict of str: float. – Dictionary of {'parameter': param_value}

outputs
list of str – List of model outputs to return.

default_creation()
Create a default input file from given arguments.

Assumes parameters remain unchanged from default values.

Returns Returns the input file as a String.IO() buffer object.

Return type String.IO()

initialised_creation(*burn_in*)
Create an input file from given arguments.

Creates an input file thatcan have non-default parameter values and outputs, as well as a burn in period. Assumes parameters remain constant for the full duration of the simulation.

Parameters **burn_in** (float or int) – Length of burn in period at start of the simulation.

Returns Returns the input file as a String.IO() buffer object.

Return type String.IO()

input_file_write()
Write input file from buffer to file.

CHAPTER

TWO

ABC

The *abc* subpackage is used to handle the Approximate Bayesian Computation (ABC) specific components of BayesCMD. This includes running the model multiple times in a batch process, calculating distances between datasets and generating priors for parameters.

2.1 Distances

Use to generate distance measures between simulated and real time series.

bayescmd.abc.distances.**DISTANCES**
dict – Dictionary contianing the distance aliases, mapping to the functions.

exception bayescmd.abc.distances.**Error**
Base class for exceptions in this module.

exception bayescmd.abc.distances.**ZeroArrayError**
Exception raised for errors in the zero array.

bayescmd.abc.distances.**check_for_key**(*dictionary*, *target*)
Check that a dictionary contains a key, and if so, return its data.

Parameters

- **dictionary** (*dict*) – Dictionary to check for *target* key.
- **target** (*str*) – String containing the target variable that is expected to be found in *dictionary*

Returns **data** – List of data found in *dictionary*. This is likely to be the time series data collected experimentally or generated by the model.

Return type list

bayescmd.abc.distances.**euclidean_dist**(*data1*, *data2*)
Get the euclidean distance between two numpy arrays.

Parameters

- **data1** (*np.ndarray*) – First data array.

  The shape should match that of data2 and the number of rows should match the number of model outputs i.e. 2 model outputs will be two rows.

- **data2** (*np.ndarray*) – Second data array.

  The shape should match that of data1 and the number of rows should match the number of model outputs i.e. 2 model outputs will be two rows.

Returns **d** – Euclidean distance measure

Return type float

bayescmd.abc.distances.**get_distance**(*actual_data*, *sim_data*, *targets*, *zero_flag*, *distance='euclidean'*, *normalise=False*)

Obtain distance between two sets of data.

Get a distance as defined by *distance* between two sets of data as well as between each signal in the data.

Parameters

- **actual_data** (*dict*) – Dictionary of actual data, as generated by `bayescmd.abc.data_import.import_actual_data()`
- **sim_data** (*dict*) – Dictionary of simulated data, as created by *bayescmd.bcmdModel.ModelBCMD.output_parse()*
- **targets** (list of `str`) – List of model targets, which should all be strings.
- **zero_flag** (*dict*) – Dictionary of form target(`str`): bool, where bool indicates whether to zero that target.

  Note: zero_flag keys should match targets list.
- **distance** (*str, optional*) – Name of distance measure to use. One of ['euclidean', 'manhattan', 'MAE', 'MSE'], where default is 'euclidean'.
- **normalise** (*bool, optional*) – Boolean flag to indicate whether the signals need normalising, default is False. Current normalisation is done using z-score but that is likely to change with time.

Returns

distances –

Dictionary of form: {'TOTAL': summed distance of all signals, 'target1: distance of 1st target', ... 'targetN': distance of Nth target }

Return type dict

bayescmd.abc.distances.**manhattan_dist**(*data1*, *data2*)

Get the Manhattan distance between two numpy arrays.

Parameters

- **data1** (*np.ndarray*) – First data array.

  The shape should match that of data2 and the number of rows should match the number of model outputs i.e. 2 model outputs will be two rows.
- **data2** (*np.ndarray*) – Second data array.

  The shape should match that of data1 and the number of rows should match the number of model outputs i.e. 2 model outputs will be two rows.

Returns **d** – Manhattan distance measure

Return type float

bayescmd.abc.distances.**mean_absolute_error_dist**(*data1*, *data2*)

Get the normalised manhattan distance between two numpy arrays.

Parameters

- **data1** (*np.ndarray*) – First data array.

  The shape should match that of data2 and the number of rows should match the number of model outputs i.e. 2 model outputs will be two rows.

- **data2** (*np.ndarray*) – Second data array.

  The shape should match that of data1 and the number of rows should match the number of model outputs i.e. 2 model outputs will be two rows.

Returns **d** – Normalised Manhattan distance measure

Return type float

bayescmd.abc.distances.**mean_square_error_dist**(*data1*, *data2*)

Get the Mean Square Error distance between two numpy arrays.

Parameters

- **data1** (*np.ndarray*) – First data array.

  The shape should match that of data2 and the number of rows should match the number of model outputs i.e. 2 model outputs will be two rows.

- **data2** (*np.ndarray*) – Second data array.

  The shape should match that of data1 and the number of rows should match the number of model outputs i.e. 2 model outputs will be two rows.

Returns **d** – Mean Square Error distance measure

Return type float

bayescmd.abc.distances.**zero_array**(*array*, *zero_flag*)

Zero an array of data with its initial values.

Parameters

- **array** (*list*) – List of data
- **zero_flags** (*bool*) – Boolean indicating if data needs zeroing

Returns **zerod** – Zero'd list

Return type list

CHAPTER

THREE

JSONPARSING

Convert model information to JSON for use with WeBCMD.

bayescmd.jsonParsing.modelJSON.**float_or_str**(*n*)
Determine if a string can be returned as a float.

Parameters **n** (str) – String to convert to float if possible

Returns **s** – *n* as float, or str if not a number.

Return type float

bayescmd.jsonParsing.modelJSON.**getDefaultFilePath**(*model_name*)
Given a model name, return the default path to the modeldef file.

Parameters **model_name** (*str*) – Name of model

Returns **modelPath** – Path to modeldef file

Return type str

bayescmd.jsonParsing.modelJSON.**get_model_name**(*fpath*)
Return model name from file path.

Parameters **fpath** (str) – Path to model def file.

Returns Name of model, as determined by modeldef file.

Return type str

bayescmd.jsonParsing.modelJSON.**json_writer**(*model_name*, *dictionary*)
Write a python dictionary to JSON.

Parameters

- **model_name** (str) – Name of BCMD model.
- **dictionary** (dict) – Dictionary to write to file.

Returns Writes to JSON file in data/ dir with name *model_name*.json

Return type None

bayescmd.jsonParsing.modelJSON.**modeldefParse**(*fpath*)
Process a modeldef file to extract information.

Function reads a modeldef file and extracts default model inputs, outputs and parameters.

Parameters **fpath** (str) – Path to modeldef file.

Returns

model_data – Dictionary of model information with form:

{model_name: name of model, obtained via *get_model_name()*,

input: `list` of `str` for each model input,

output: `list` of `str` for each model output,

parameters: `dict` of param_name (`str`): param_value(`str`)}

Return type `dict`

CHAPTER

FOUR

MISCELLANEOUS

Here you will find a number of useful functions that are used throughout the general BayesCMD package.

4.1 Utility Functions

Miscellaneous utility functions used throughout BayesCMD.

This module contains a number of utility functions that are used throughout the different BayesCMD subpackages.

bayescmd.util.**findBaseDir**(*basename*, *max_depth=5*, *verbose=False*)
Get relative path to a BASEDIR. :param basename: Name of the basedir to path to :type basename: str

Returns Relative path to base directory.

Return type StringIO

bayescmd.util.**round_sig**(*x*, *sig=1*)
Round a value to N sig fig.

Parameters

- **x** (*float*) – Value to round
- **sig** (*int, optional*) – Number of sig figs, default is 1

Returns Rounded value

Return type float

4.2 Processing Results

Process results obtained using BayesCMD.

Process the various results obtained using BayesCMD, such as the *parameters.csv* file. It is also possible to concatenate a number of different *parameters.csv* files obtained using parallel batch runs into a single parameters file.

bayescmd.results_handling.**BAYESCMD**
str – Absolute path to base directory. Found using *bayescmd.util.findBaseDir*

bayescmd.results_handling.**data_import**(*pfile*, *nan_sub=100000*, *chunk_size=10000*, *verbose=True*)
Import a parameters file produced by a batch process.

Parameters

- **pfile** (*str*) – Path to the file of parameters and distances

- **nan_sub** (*int or float, optional*) – Number to substitute for NaN distances/params. Default of 100000
- **chunk_size** (*int, optional*) – Size of chunks to load for dataframe. Default of 10000
- **verbose** (*bool, optional*) – Boolean as to whether include verbose information. Default of True

Returns **result** – Dataframe containing all the parameters and distances, with NaN swapped for nan_sub

Return type pd.DataFrame

bayescmd.results_handling.**data_merge**(*parent_directory*, *verbose=True*)

Merge a set of parameters.csv files into one.

Parameters

- **parent_directory** (list of str) – Parent directory to a set of directories each containing model runs and a parameters.csv file.
- **verbose** (boolean, optional) – Boolean indicator of whether to print extra information.

Returns Concatenated will be written to file in *parent_directory*

Return type None

bayescmd.results_handling.**diag_kde_plot**(*x*, *medians*, ***kws*)

Plot univariate KDE and barplot with median of distribution marked on.

Includes median of distribution as a line and as text.

Parameters

- **x** (*array-like*) – Array-like of data to plot.
- **medians** (dict) – Dictionary of parameter, median pairings.
- **kws** (*key, value pairings.*) – Other keyword arguments to pass to sns.distplot.

Returns **ax** – AxesSubplot object of univariate KDE and bar plot with median marked on as well as text.

Return type matplotlib.AxesSubplot

bayescmd.results_handling.**frac_calculator**(*df*, *frac*)

Calculate the number of lines for a given fraction.

Parameters

- **df** (*pd.DataFrame*) – Data frame to find fraction of. Normally the output of *data_import*
- **frac** (*float*) – The fraction of results to consider. Should be given as a percentage i.e. 1=1%, 0.1=0.1%

Returns Number of lines that make up the fraction.

Return type int

bayescmd.results_handling.**get_output**(*model_name*, *p*, *times*, *input_data*, *d0*, *targets*, *distance='euclidean'*, *zero_flag=None*)

Generate model output and distances.

Parameters

- **model_name** (str) – Name of model
- **p** (dict) – Dict of form {'parameter': value} for which posteriors are being investigated.
- **times** (list of float) – List of times at which the data was collected.
- **input_data** (dict) – Dictionary of input data as generated by abc.inputParse.
- **d0** (dict) – Dictionary of real data, as generated by abc.import_actual_data.
- **targets** (list of str) – List of model outputs against which the model is being optimised.
- **distance** (str) – Distance measure. One of 'euclidean', 'manhattan', 'MAE', 'MSE'.
- **zero_flag** (*dict*) – Dictionary of form target(str): bool, where bool indicates whether to zero that target.

  Note: zero_flag keys should match targets list.

Returns A tuple of (p, model output data).

Return type tuple

bayescmd.results_handling.**histogram_plot**(*df*, *distance='euclidean'*, *fraction=1*, *n_bins=100*)

Plot histogram of distance values.

Plot a histogram showing the distribution of distance values for a given fraction of all distances in the dataframe. Distance values will have been calculated during the batch process.

Parameters

- **df** (pandas.DataFrame) – Dataframe of distances and parameters, generated using *data_import()*
- **distance** (str, optional) – Distance measure. One of 'euclidean', 'manhattan', 'MAE', 'MSE'. Default is 'euclidean'.
- **fraction** (float, optional) – Fraction of all distances to plot. Varies from 0 to 1. Default is 1.
- **n_bins** (int, optional) – Number of histogram bins. Default is 100.

Returns Matplotlib figure with histogram on.

Return type matplotlib.figure

bayescmd.results_handling.**kde_plot**(*df*, *params*, *frac*, *plot_param=1*, *n_ticks=6*, *d='euclidean'*, *verbose=False*)

Plot the model parameters pairwide as a KDE.

Parameters

- **df** (pandas.DataFrame) – Dataframe of distances and parameters, generated using *data_import()*
- **params** (dict of str: tuple) – Dict of model parameters to compare, with value tuple of the prior max and min.
- **frac** (float) – Fraction of results to consider. Should be given as a percentage i.e. 1=1%, 0.1=0.1%
- **plot_param** (int) – Which group to plot:

  0: Outside posterior 1: Inside posterior 2: Failed run

- **n_ticks** (int, optional) – Number of x-axis ticks. Useful when a large number of parameters are bring compared, as the axes can become crowded if the number of ticks is too high.
- **d** (str, optional) – Distance measure. One of 'euclidean', 'manhattan', 'MAE', 'MSE'.

  Note: Should be given as a raw string if latex is used i.e. *r'MAE'*.
- **verbose** (boolean, optional) – Boolean to indicate verbosity. Default is False.

Returns **g** – Seaborn pairgrid object is returned in case of further formatting.

Return type seaborn.PairGrid

bayescmd.results_handling.**plot_repeated_outputs**(*df*, *model_name*, *parameters*, *input_path*, *inputs*, *targets*, *n_repeats*, *frac*, *zero_flag*, *openopt_path=None*, *distance='euclidean'*)

Generate model output and distances multiple times.

Parameters

- **model_name** (str) – Name of model. Should match the modeldef file for model being generated i.e. model_name of 'model'' should have a modeldef file 'model1.modeldef'.
- **parameters** (dict of str: tuple) – Dict of model parameters to compare, with value tuple of the prior max and min.
- **input_path** (str) – Path to the true data file
- **inputs** (list of str) – List of model inputs.
- **targets** (list of str) – List of model outputs against which the model is being optimised.
- **n_repeats** – Number of times to generate output data
- **frac** (float) – Fraction of results to consider. Should be given as a percentage i.e. 1=1%, 0.1=0.1%
- **zero_flag** (*dict*) – Dictionary of form target(str): bool, where bool indicates whether to zero that target.

  Note: zero_flag keys should match targets list.
- **openopt_path** (str or None) – Path to the openopt data file if it exists. Default is None.
- **distance** (str, optional) – Distance measure. One of 'euclidean', 'manhattan', 'MAE', 'MSE'.

Returns **fig** – Figure containing all axes.

Return type matplotlib.figure

bayescmd.results_handling.**run_model**(*model*)

Run a BCMD Model.

Parameters **model** (*bayescmd.bcmdModel.ModelBCMD*) – An initialised instance of a ModelBCMD class.

Returns **output** – Dictionary of parsed model output.

Return type dict

bayescmd.results_handling.**scatter_dist_plot**(*df*, *params*, *frac*, *n_ticks=6*, *d='euclidean'*, *verbose=False*)

Plot distribution of parameters as a scatter PairPlot.

Parameters

- **df** (pandas.DataFrame) – Dataframe of distances and parameters, generated using *data_import()*
- **params** (dict of str: tuple) – Dict of model parameters to compare, with value tuple of the prior max and min.
- **frac** (float) – Fraction of results to consider. Should be given as a percentage i.e. 1=1%, 0.1=0.1%
- **n_ticks** (int, optional) – Number of x-axis ticks. Useful when a large number of parameters are bring compared, as the axes can become crowded if the number of ticks is too high.
- **d** (str, optional) – Distance measure. One of 'euclidean', 'manhattan', 'MAE', 'MSE'.

  Note: Should be given as a raw string if latex is used i.e. *r'MAE'*.
- **verbose** (boolean, optional) – Boolean to indicate verbosity. Default is False.

Returns **g** – Seaborn pairgrid object is returned in case of further formatting.

Return type seaborn.PairGrid

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

b

INDEX

Symbols

B

C

D

E

F

G

H

I