mcmcplot Documentation

Release 1.0.1

Paul Miles

Mar 09, 2020

CONTENTS

The mcmcplot package is designed to assist in the analysis of sampling chains gathered during a Markov Chain Monte Carlo (MCMC) simulation. This package was designed with the MCMC code pymcmcstat in mind; however, the plotting routines are amenable to other data sets. The plotting routines use matplotlib and seaborn. User's are recommended to investigate other plotting routines available in seaborn as it is specifically designed for this sort of analysis. The routines available in mcmcplot serve as a useful wrapper function for several seaborn plots, but it is not an exhaustive demonstration.

CHAPTER

ONE

INSTALLATION

This code can be found on the Github project page. The package is available on the PyPI distribution site and the latest version can be installed via

```
pip install mcmcplot
```

The master branch typically matches the latest version on the PyPI distribution site. To install the master branch directly from Github,

```
pip install git+https://github.com/prmiles/mcmcplot.git
```

You can also clone the repository and run `python setup.py install`.

CHAPTER

TWO

GETTING STARTED

- Tutorial notebooks
- Documentation
- Release history

CHAPTER

THREE

LICENSE

MIT

CHAPTER

FOUR

CONTRIBUTORS

See the GitHub contributor page

CHAPTER

FIVE

CITING MCMCPLOT

Please see the mcmcplot homepage or follow the DOI badge above to find the appropriate citation information.

CHAPTER

SIX

FEEDBACK

- Feature Request
- Bug Report

CHAPTER

SEVEN

SPONSOR

This work was sponsored in part by the NNSA Office of Defense Nuclear Nonproliferation R&D through the Consortium for Nonproliferation Enabling Capabilities.

CHAPTER

EIGHT

CONTENTS:

8.1 mcmcplot package

8.1.1 mcmcplot.mcmatplot module

Created on Wed Jan 31 12:54:16 2018

@author: prmiles

mcmcplot.mcmatplot.**plot_chain_metrics**(*chain*, *name=None*, *settings=None*, *return_settings=False*)

Plot chain metrics for individual chain

- Scatter plot of chain
- Histogram of chain

Args:

- **chains** (ndarray): Sampling chain for specific parameter

Kwargs:

- **names** (str): Name of each parameter
- **settings** (dict): Settings for features of this method.
- **return_settings** (bool): Flag to return figure settings. Default: *False*

Returns:

- If *return_settings=True*, (tuple): (figure handle, settings actually used in program)
- Otherwise, figure handle

mcmcplot.mcmatplot.**plot_chain_panel**(*chains*, *names=None*, *settings=None*, *skip=1*, *maxpoints=500*, *return_settings=False*)

Plot sampling chain for each parameter

Args:

- **chains** (ndarray): Sampling chain for each parameter

Kwargs:

- **names** (list): List of strings - name of each parameter
- **settings** (dict): Settings for features of this method.
- **skip** (int): Indicates step size to be used when plotting elements from the chain

- **maxpoints** (int): Max number of display points - keeps scatter plot from becoming overcrowded
- **return_settings** (bool): Flag to return figure settings. Default: *False*

Returns:

- If *return_settings=True*, (tuple): (figure handle, settings actually used in program)
- Otherwise, figure handle

mcmcplot.mcmatplot.**plot_density_panel**(*chains*, *names=None*, *settings=None*, *return_kde=False*, *hist_on=False*, *return_settings=False*)

Plot marginal posterior densities

Args:

- **chains** (ndarray): Sampling chain for each parameter

Kwargs:

- **names** (list): List of strings - name of each parameter. Default: *None*
- **settings** (dict): Settings for features of this method. Default: *None*
- **return_kde** (bool): Flag to return handles of functions from KDE. Default: *False*
- **return_settings** (bool): Flag to return figure settings. Default: *False*
- **hist_on** (bool): Flag to include histogram on plot with marginal distribution.

Returns:

- If *return_settings=True* and *return_kde=True*, (tuple): (figure handle, settings used, kde handles)
- If *return_settings=True* and *return_kde=False*, (tuple): (figure handle, settings used)
- If *return_settings=False* and *return_kde=True*, (tuple): (figure handle, kde handles)
- Otherwise, figure handle

mcmcplot.mcmatplot.**plot_histogram_panel**(*chains*, *names=None*, *settings=None*, *return_settings=False*)

Plot histogram from each parameter's sampling history

Args:

- **chains** (ndarray): Sampling chain for each parameter

Kwargs:

- **names** (list): List of strings - name of each parameter. Default: *None*
- **settings** (dict): Settings for features of this method. Default: *None*
- **return_settings** (bool): Flag to return figure settings. Default: *False*

Returns:

- If *return_settings=True*, (tuple): (figure handle, settings actually used in program)
- Otherwise, figure handle

mcmcplot.mcmatplot.**plot_pairwise_correlation_panel**(*chains*, *names=None*, *settings=None*, *skip=1*, *maxpoints=500*, *return_settings=False*)

Plot pairwise correlation for each parameter

Args:

- **chains** (ndarray): Sampling chain for each parameter

Kwargs:

- **names** (list): List of strings - name of each parameter
- **settings** (dict): Settings for figure features made by this method.
- **skip** (int): Indicates step size to be used when plotting elements from the chain
- **maxpoints** (py:class:*int*): Maximum allowable number of points in plot.
- **return_settings** (bool): Flag to return figure settings. Default: *False*

Returns:

- If *return_settings=True*, (tuple): (figure handle, settings actually used in program)
- Otherwise, figure handle

8.1.2 mcmcplot.mcseaborn module

Created on August 5, 2018

@author: prmiles

mcmcplot.mcseaborn.**plot_joint_distributions**(*chains*, *names=None*, *settings=None*, *maxpoints=500*, *skip=1*, *return_settings=False*)

Plot joint distribution for each parameter set.

https://seaborn.pydata.org/generated/seaborn.jointplot.html

Args:

- **chains** (ndarray): Sampling chain for each parameter

Kwargs:

- **names** (list): List of strings - name of each parameter. Default: *None*
- **settings** (dict): Settings for features of this method. Default: *None*
- **skip** (int): Indicates step size to be used when sampling elements from the chain. Default: *1*
- **maxpoints** (int): Max number of sample points - keeps generation of KDE shorter. Default: *500*
- **return_settings** (bool): Flag to return figure settings. Default: *False*

Returns:

- If *return_settings=True*, (tuple): (figure handle, settings actually used in program)
- Otherwise, figure handle

mcmcplot.mcseaborn.**plot_paired_density_matrix**(*chains*, *names=None*, *index=None*, *settings=None*, *return_settings=False*)

Plot paired density matrix.

https://seaborn.pydata.org/generated/seaborn.PairGrid.html

Args:

- **chains** (ndarray): Sampling chain for each parameter

Kwargs:

- **names** (list): List of strings - name of each parameter. Default: *None*
- **settings** (dict): Settings for features of this method. Default: *None*
- **index** (list): Category for each row of chain. Default: *None*
- **return_settings** (bool): Flag to return figure settings. Default: *False*

Returns:

- If *return_settings=True*, (tuple): (figure handle, settings actually used in program)
- Otherwise, figure handle

8.1.3 mcmcplot.utilities module

Created on Mon May 14 06:24:12 2018

@author: prmiles

mcmcplot.utilities.**append_to_nrow_ncol_based_on_shape**(*sh*, *nrow*, *ncol*)

Append to list based on shape of array

Args:

- **sh** (tuple): Shape of array.
- **nrow** (list): List of number of rows
- **ncol** (list): List of number of columns

Returns:

- **nrow** (list): List of number of rows
- **ncol** (list): List of number of columns

mcmcplot.utilities.**check_settings**(*default_settings*, *user_settings=None*)

Check user settings with default.

Recursively checks elements of user settings against the defaults and updates settings as it goes. If a user setting does not exist in the default, then the user setting is added to the settings. If the setting is defined in both the user and default settings, then the user setting overrides the default. Otherwise, the default settings persist.

Args:

- **default_settings** (dict): Default settings for particular method.

Kwargs:

- **user_settings** (dict): User defined settings. Default: *None*

Returns:

- (dict): Updated settings.

mcmcplot.utilities.**check_symmetric**(*a*, *tol=1e-08*)

Check if array is symmetric by comparing with transpose.

Args:

- **a** (ndarray): Array to test.

Kwargs:

- **tol** (float): Tolerance for testing equality. Default: *1e-8*

Returns:

- (bool): True -> symmetric, False -> not symmetric.

mcmcplot.utilities.**extend_names_to_match_nparam**(*names*, *nparam*)

Append names to list using default convention until length of names matches number of parameters.

For example, if *names = ['name_1', 'name_2']* and *nparam = 4*, then two additional names will be appended to the *names* list. E.g.,:

```
names = ['name_1', 'name_2', '$p_{2}$', '$p_{3}$']
```

Args:

- **names** (list): Names of parameters provided by user
- **nparam** (int): Number of parameter names to generate

Returns:

- **names** (list): List of strings - extended list of parameter names

mcmcplot.utilities.**gaussian_density_function**(*x*, *mu=0*, *sigma2=1*)

Standard normal/Gaussian density function.

Args:

- **x** (float): Value of which to calculate probability.

Kwargs:

- **mu** (float): Mean of Gaussian distribution. Default: *0*
- **sigma2** (float): Variance of Gaussian distribution. Default: *1*

Returns:

- **y** (float): Likelihood of *x*.

mcmcplot.utilities.**generate_default_names**(*nparam*)

Generate generic parameter name set.

For example, if *nparam* = 4, then the generated names are:

```
names = ['$p_{0}$', '$p_{1}$', '$p_{2}$', '$p_{3}$']
```

Args:

- **nparam** (int): Number of parameter names to generate

Returns:

- **names** (list): List of strings - parameter names

mcmcplot.utilities.**generate_ellipse**(*mu*, *cmat*, *ndp=100*)

Generates points for a probability contour ellipse

Args:

- **mu** (ndarray): Mean values
- **cmat** (ndarray): Covariance matrix

Kwargs:

- **npd** (int): Number of points to generate. Default: *100*

Returns:

- **x** (ndarray): x-points
- **y** (ndarray): y-points

mcmcplot.utilities.**generate_ellipse_plot_points**(*x*, *y*, *ndp=100*)
Generates points for a probability contour ellipse for 2 columns of chain

Args:

- **x** (ndarray): chain 1
- **y** (ndarray): chain 2

Kwargs:

- **npd** (int): Number of points to generate. Default: *100*

Returns:

- (dict): 50% and 95% probability contours.

mcmcplot.utilities.**generate_names**(*nparam*, *names*)
Generate parameter name set.

For example, if *nparam* = 4, then the generated names are:

```
names = ['p_{0}', 'p_{1}', 'p_{2}', 'p_{3}']
```

Args:

- **nparam** (int): Number of parameter names to generate
- **names** (list): Names of parameters provided by user

Returns:

- **names** (list): List of strings - parameter names

mcmcplot.utilities.**generate_subplot_grid**(*nparam=2*)
Generate subplot grid.

For example, if *nparam* = 2, then the subplot will have 2 rows and 1 column.

Kwargs:

- **nparam** (int): Number of parameters. Default: *2*

Returns:

- **ns1** (int): Number of rows in subplot
- **ns2** (int): Number of columns in subplot

mcmcplot.utilities.**iqrange**(*x*)
Interquantile range of each column of x.

Args:

- **x** (ndarray): Array of points.

Returns:

- (ndarray): Interquantile range - single element array, *q3 - q1*.

mcmcplot.utilities.**is_semi_pos_def_chol**(*x*)
Check if matrix is semi positive definite by calculating Cholesky decomposition.

Args:

- **x** (ndarray): Matrix to check

Returns:

- If matrix is *not* semi positive definite return `False, None`
- If matrix is semi positive definite return `True` and the Upper triangular form of the Cholesky decomposition matrix.

mcmcplot.utilities.**make_x_grid**(*x*, *npts=100*)

Generate x grid based on extrema.

1. If *len(x) > 200*, then generates grid based on difference between the max and min values in the array.

2. Otherwise, the grid is defined with respect to the array mean plus or minus four standard deviations.

Args:

- **x** (ndarray): Array of points

Kwargs:

- **npts** (int): Number of points to use in generated grid. Default: *100*

Returns:

- (ndarray): Uniformly spaced array of points with shape `(npts,1)`.

mcmcplot.utilities.**scale_bandwidth**(*x*)

Scale bandwidth of array.

Args:

- **x** (ndarray): Array of points - column of chain.

Returns:

- **s** (ndarray): Scaled bandwidth - single element array.

mcmcplot.utilities.**setup_subsample**(*skip*, *maxpoints*, *nsimu*)

Setup subsampling from posterior.

When plotting the sampling chain, it is often beneficial to subsample in order to avoid to dense of plots. This routine determines the appropriate step size based on the size of the chain (nsimu) and maximum points allowed to plot (maxpoints). The function checks if the size of the chain exceeds the maximum number of points allowed in the plot. If yes, skip is defined such that every the max number of points are used and sampled evenly from the start to end of the chain. Otherwise the value of skip is return as defined by the user. A subsample index is then generated based on the value of skip and the number of simulations.

Args:

- **skip** (int): User defined skip value.
- **maxpoints** (int): Maximum points allowed in each plot.

Returns:

- (int): Skip value.

8.1.4 Module contents

CHAPTER

NINE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

m

INDEX