Superplot: a graphical interface for plotting and analysing MultiNest output

Andrew Fowlie\(^1\,a\) and Michael Hugh Bardsley\(^2\)

\(^1\) ARC Centre of Excellence for Particle Physics at the Tera-scale, School of Physics and Astronomy, Monash University, Melbourne, Victoria 3800, Australia
\(^2\) School of Physics and Astronomy, Monash University, Melbourne, Victoria 3800, Australia

Received: 14 July 2016 / Revised: 21 September 2016
Published online: 9 November 2016 – © Società Italiana di Fisica / Springer-Verlag 2016

Abstract. We present an application, Superplot, for calculating and plotting statistical quantities relevant to parameter inference from a “chain” of samples drawn from a parameter space, produced by, e.g., MultiNest. A simple graphical interface allows one to browse a chain of many variables quickly, and make publication quality plots of, \textit{inter alia}, one- and two-dimensional profile likelihood, posterior pdf (with kernel density estimation), confidence intervals and credible regions. In this short manual, we document installation and basic usage, and define all statistical quantities and conventions. The code is fully compatible with Linux and Windows. All functionality is available on Mac OS X, though it must be invoked by the command line rather than a graphical interface.

1 Introduction

Many branches of physics are utilising sophisticated numerical methods to infer the parameters of a model from data. This typically involves a numerical exploration of a parameter space with a Monte-Carlo algorithm, resulting in a collection of weighted samples drawn from the parameter space (henceforth referred to as a chain). A modern example of such an algorithm is nested sampling \cite{1,2}. The popular MultiNest \cite{3–5} implementation of nested sampling is utilised for parameter extraction in manifold areas of physics, including supersymmetric fits (see, e.g., refs. \cite{6–9}), cosmological fits (see, e.g., refs. \cite{10–16}) and X-ray analysis (see, e.g., refs. \cite{17,18}), and in a forthcoming general purpose fitting program, GAMBIT \cite{19}.

We present an application, Superplot, for plotting parameters extracted by MultiNest (or a similar code, such as PolyChord \cite{20,21}) with the matplotlib plotting library \cite{22}. This should simplify the final step in parameter extraction: calculating and plotting results from a chain, such as posterior density, profile likelihood, credible regions or confidence intervals. This is, of course, already possible with private scripts or pippi \cite{23}, SuperEGO \cite{24}/CosmoloGUI \cite{25}, modified versions of the programs GetPlots \cite{26}/GetDist \cite{27} and even ROOT \cite{28}. The advantage of Superplot is that a graphical interface allows one to quickly browse a chain of many variables and make publication quality plots, with control over binning or kernel density estimation (KDE), without writing any scripts or codes. We describe installation instructions in sect. 2 and usage in sect. 3. Definitions and conventions of all statistical quantities are provided in appendix A.

2 Installation

Superplot is a Python 2.7 code. The simplest method of installing Superplot is via the pip package manager\(^1\). The command

```
$ pip install superplot
```

\(^1\) pip is included in Python beginning in version 2.7.9. If your version of Python does not include the pip package manager, see https://pip.pypa.io/en/latest/installing/.
should install Superplot and dependencies, except matplotlib. Note, however, that on some operating systems, pip may not be able to automatically build dependencies. Thus matplotlib and other dependencies may have to be separately installed; see sect. 2.1.

Superplot was tested on Linux, Windows and Mac OSX. Mac OSX is supported at the command line only (see sect. 3.3), as unfortunately there are known issues with the graphical interface on Mac OSX. Linux and Windows are fully supported. All plotting functionality and summary statistics are available from the command line on Linux, Windows and Mac OSX, should you experience issues with the graphical interface on your system. In Linux, programs are placed in a platform-dependent directory, e.g., ~/.local/bin for Ubuntu and Mint. To invoke the programs directly, e.g., superplot_gui, this directory must be in the user’s path.

If installation via pip is problematic, clone the source code from GitHub,

```
git clone https://github.com/michaelhb/superplot.git
```

or download the source code from https://github.com/michaelhb/superplot/archive/master.zip. You may need to refer to sect. 2.1 for instructions on installing dependencies. Once all dependencies are satisfied, Superplot can be installed and run from the source directory,

```
python setup.py install
python ./superplot/super_gui.py
```

Finally, you may wish to place configuration and example files in a convenient location,

```
superplot_create_home_dir -d <path_to_directory>  # Linux (if ~/.local/bin in path) or
python -m superplot.create_home_dir -d <path_to_directory>  # Linux/Windows/Mac OSX
```

See sect. 3.1 for further details.

### 2.1 Dependencies

Superplot requires some common Python modules. The most obscure dependencies can be installed via pip,

```
pip install appdirs prettytable simpleyaml joblib
```

but should be installed automatically by pip install superplot. Other dependencies may require OS-specific installation, including matplotlib [22] version 1.4 or newer with gtk support, numpy [29], scipy [30] and pandas [31] from the SciPy stack², and PyGTK³. For, e.g., Ubuntu 16 users, dependencies may be installed via

```
apt-get install git python-pip python-numpy python-scipy python-pandas
libfreetype6-dev python-gtk2-dev python-matplotlib
```

For users of other Linux distributions, including Mint, it may be necessary to install missing dependencies python-setuptools, python-tk and python-wheel via, e.g.,

```
apt-get install python-setuptools python-tk python-wheel
```

Whereas for Windows users, dependencies may be installed by

1. Installing the Anaconda Python distribution from https://www.continuum.io/downloads
2. Installing the PyGTK all-in-one bundle from http://ftp.acc.umu.se/pub/GNOME/binaries/win32/pygtk/2.24
3. Upgrading matplotlib, pip install --force-reinstall --no-deps --upgrade matplotlib
4. Finally, installing Superplot, pip install superplot

### 3 Quickstart

The main component of Superplot is superplot_gui — a graphical interface for making plots from a chain. To start superplot_gui, from any directory run either

```
superplot_gui  # Linux (if ~/.local/bin in path) or
python -m superplot.super_gui  # Linux/Windows (issues with Mac OSX)
```
Fig. 1. Selecting data files for Superplot.

(a) First, open a *.txt file from MultiNest.
(b) Second, optionally, open a *.info file that labels columns in the *.txt file.
(c) Finally, select a plot, and click Make plot in the graphical interface in Superplot.

The latter is advised for Windows as it may avoid problems with stdout. You will be prompted to select a MultiNest chain ending in *.txt (fig. 1(a)). Select a chain of your choice, e.g., SB_MO_log_allpost.txt—a chain from SuperBayes [7,8], see sect. 3.1 for further details about its location—and click Open. You will be asked whether you wish to select an information file (fig. 1(b)). This optional file could contain labels and metadata associated with the chain (see sect. 3.5). Select, e.g., SB_MO_log_all.info and click Open. If you select No information file..., the variables in the chain will be assigned numerical labels based on column order.

After the information file is selected, the main graphical interface appears (fig. 1(c)). The left-hand side of the window contains controls for configuring the plot, including:

- Type of plot (Plot type); see fig. 2 for examples. The possibilities are:
  i) One-dimensional plot One-dimensional marginalised pdf, $p(x)$, and/or profile likelihood, $\mathcal{L}(x)$.
  ii) One-dimensional chi-squared plot One-dimensional chi-squared with an theoretical error band.
  iii) Two-dimensional posterior pdf, filled contours only Two-dimensional credible regions of posterior pdf, illustrated by filled contours.
  iv) Two-dimensional profile likelihood, filled contours only Two-dimensional confidence intervals, illustrated by filled contours.
  v) Two-dimensional posterior pdf Two-dimensional posterior pdf, $p(x,y)$, illustrated by shading on a two-dimensional plane.
  vi) Two-dimensional profile likelihood Two-dimensional profile likelihood, $\mathcal{L}(x,y)$, illustrated by shading on a two-dimensional plane.
  vii) Three-dimensional scatter plot Three-dimensional scatter plot—all samples scattered on a two-dimensional plane, shaded by the value of a third variable.
- Variables you wish to plot (e.g., x-axis variable).
- The axis labels, e.g., type the x-axis label in text-box below x-axis variable. Labels may include a \LaTeX math environment ($...$) and are, by default, rendered with pdflatex. Any pdflatex errors should be printed to the terminal.
- Whether a variable should be logged (e.g., Log x-data). If selected, data is logged and then binned. This differs from binning and plotting on a logarithmic scale.
- The number of bins per dimension and bin limits.
- The limits for the x- and y-axis.
- Plot title, legend title and legend position.
- Selection of optional plot elements, e.g., the best-fit point or posterior mean.
- Whether to use kernel density estimation (KDE) of pdfs, as described in appendix B.

Once you have selected the plot you wish, click the Make plot button, located below the plot options. The desired plot should appear in the central window, as in fig. 1(c). If no plot appears, check whether any, e.g., \LaTeX errors were printed to the terminal. If so, fix any malformed \LaTeX and click Make plot.

The controls for saving plots are located below the central window. The option Save image creates a high-quality PDF version of the displayed plot, Save statistics in plot writes a text file containing summary statistics and metadata about the plot, and Save pickle of plot writes a serialised copy of the matplotlib plot object\(^4\). After selecting the desired outputs, press the Save plot button and select a location on the disk.

### 3.1 Configuration and example files

Beyond the control panel in the graphical interface, the appearance of a plot can be customised by editing configuration files included in Superplot. If Superplot was installed via pip, configuration and example files are located in the platform-dependent user data directory. To place them in a directory of your choice,

```
superplot_create_home_dir -d <path_to_directory>  # Linux (if ~/.local/bin in path) or
python -m superplot.create_home_dir -d <path_to_directory>  # Linux/Windows/Mac OSX
```

These files are used in preference to any copies installed with the source code. If they are deleted, Superplot will revert to copies installed with the source code. If the script is run more than once, Superplot will use the configuration files in the most recently created directory.

If the source code was downloaded, configuration files are included in superplot/config.yml and superplot/plotlib/styles, and example files are included in superplot/example. For further documentation of the code itself, see the API\(^5\).

The yaml file config.yml contains “schemes” which control the sizes, colours, symbols and labelling of individual plot elements, e.g., the symbol for the best-fit point. These are specified using matplotlib conventions —see the file header for further information. The styles/ directory contains a set of matplotlib style sheets\(^6\). Options such as line thickness, grid lines and fonts can be fine-tuned by editing these files. The file default.mplstyle contains options which apply to all plots. There are also style sheets specific to each plot type. The individual style sheets take precedence over default.mplstyle, allowing plot-specific customisation. The example/ directory contains, inter alia, *.txt and *.info example files for Superplot.

### 3.2 Summary statistics for chain

Superplot also includes a command that generates a table of statistics for a chain. This can be launched by running either

```
superplot_summary --data_file DATA_FILE [--info_file INFO_FILE]  # Linux (if
  ~/.local/bin in path) or
python -m superplot.summary --data_file DATA_FILE [--info_file INFO_FILE]  #
  Linux/Windows/Mac OSX
```

You should specify a chain and an (optional) information file by the command-line arguments. A table containing the label, best-fit point, posterior mean and 1σ credible region for every variable in the chain is printed to the terminal.

---

4 See http://matplotlib.org/users/whatsnew.html#figures-are-picklable. A pickle is a serialised copy of the plot object which can be loaded in a separate Python session. This may be useful if a plot needs to be customised in a manner not otherwise possible. For an example, see example/load_pkl.py.

5 See http://superplot.readthedocs.org.

6 See http://matplotlib.org/users/style_sheets.html.
3.3 Making Superplot plot via the command line

It is possible to make a Superplot plot from the command line, bypassing the GUI. For all possible options and usage, see

```
python -m superplot.super_command --help  # Linux/Windows/Mac OSX
```

The command-line interface may be necessary on Mac OSX or systems that fail PyGTK dependencies. Examples of simple and more complicated usage are

```
# Produce a 1D plot with many default options
# You may need to alter the path for the *.txt file
python -m superplot.super_command ~/.local/share/superplot/example/SB_MO_log_allpost.txt --xindex=4

# Produce a 2D plot, with many options specified at the command line
# You may need to alter the path for the *.txt file
python -m superplot.super_command ~/.local/share/superplot/example/SB_MO_log_allpost.txt --xindex=4 --yindex=5 --xlabel='$x$-label' --ylabel='y$-label' --logy=True --kde=True --plot_title='Example plot' --output_file='example.pdf'
```
The code saves a Superplot plot to the disk. In the first case, a descriptive name for the plot with .pdf extension is chosen by the code and printed to the screen. In the second case, the file name and extension example.pdf are specified at the command line. There is one compulsory positional argument —the name of the *.txt file—and one compulsory named argument —the index of the variable on the x-axis, --xindex=. All other arguments are optional and are specified explicitly by name.

The command line accepts a full set of plot options (including options otherwise only specified in the yaml file config.yml), and invokes the same plotting and statistical libraries as the GUI.

3.4 Use with programs other than MultiNest

Superplot reads data from the MultiNest text file (*.txt), which is a plain-text array of floats separated by at least one space. The first and second columns are the posterior weight and $-2\ln L$ or chi-squared of each sample, respectively. Subsequent columns are parameter values associated with each sample. Superplot would work with any data in this format. If you consider only frequentist statistics and calculate only a chi-squared for each sample, the first column could be a place-holder—the frequentist quantities could still be plotted.

3.5 Information file

We inherited the format of an information file from getdist. The information file (*.info) provides labels for the parameters in the chain, beginning with the third column (i.e. ignoring posterior weight and $-2\ln L$). The *.info file is optional. If provided, Superplot automatically labels columns of the chain with the *.info file. The format of an entry in the *.info file is, e.g.,

```
lab1 = $m_0$ (GeV)
```

This would label the first parameter (i.e. third column) “$m_0$ (GeV)”. You do not have to provide labels for all columns. The label may include a \LaTeX math environment ($...$) and should not be enclosed in, e.g., quotes. Each label should be on a new line. All other text in the *.info file is ignored.

4 Summary and bug reporting

We detailed a new application, Superplot, for plotting a chain from, e.g., MultiNest. The application should be easy to install and use, and simplify the final step of parameter inference from a chain. To keep track of developments, report any bugs or ask for help, please see https://github.com/michaelhb/superplot/issues.

Appendix A. Statistical functions

All statistical quantities in Superplot are calculated numerically. With the exception of the best-fit point, an error of about half a bin width is introduced in all quantities by binning. We advise that a user chooses the number of bins carefully, compromising between unnecessarily discarding information by coarse binning and noisy, comb-like distributions. Out of the box, Superplot includes options for plotting many Bayesian and frequentist statistical quantities. By default, confidence intervals and credible regions are calculated at 1$\sigma$ and 2$\sigma$ two-tail significances, i.e. for probabilities

$$\beta = 1 - \alpha = 2\Phi(z) - 1$$

for $z = 1$ and $z = 2$, where $\Phi$ is the cdf for the standard normal distribution. This results in $\alpha \approx 0.32$ and $\alpha \approx 0.046$. All statistical functions were tested against data drawn from Gaussian distributions with known properties.

Appendix A.1. Bayesian

We briefly describe the Bayesian statistical quantities in Superplot, including our conventions and ordering rules. See, e.g., ref. [32] for further discussion. Typically, quantities based upon the posterior pdf are not parameterisation invariant. We note only exceptions to this rule.

Marginalised posterior pdf. Defined

$$p(x) = \int p(x, y, z, \cdots) \, dy \, dz \, d\cdots$$

(A.2)

$$p(x, y) = \int p(x, y, z, \cdots) \, dz \, d\cdots,$$

(A.3)
in the one- and two-dimensional cases, respectively. The pdf is calculated with a weighted histogram of samples in a chain with user-defined binning or by KDE as described in appendix B.

**One-dimensional credible regions.** Defined by a symmetric ordering rule, *i.e.* an equal probability is contained in each tail

\[
\int_{-\infty}^{x_a} p(x) \, dx = \int_{x_b}^{\infty} p(x) \, dx = \frac{1}{2} \alpha, \tag{A.4}
\]

\[
\int_{x_a}^{x_b} p(x) \, dx = 1 - \alpha, \tag{A.5}
\]

with a user-defined probability \(\alpha\). This is invariant for monotonic transformations. One-dimensional credible regions are calculated by summing one-dimensional marginalised pdf and basic linear interpolation or by KDE as described in appendix B.

**Two-dimensional credible regions.** Defined by an ordering rule which is such that the credible region is the smallest region containing a given fraction of the total probability, *i.e.* a credible region is the region \(R\) such that \(\int_{\mathcal{R}} p(x, y) \, dx \, dy\) is minimised subject to the constraint that

\[
\int_{\mathcal{R}} p(x, y) \, dx \, dy = 1 - \alpha \tag{A.6}
\]

with a user-defined probability \(\alpha\). Two-dimensional credible regions are calculated by finding the posterior density \(p_{\text{crit}}\) such that

\[
\int_{p(x, y) \geq p_{\text{crit}}} p(x, y) \, dx \, dy = 1 - \alpha \tag{A.7}
\]

is satisfied.

**Posterior mean.** Defined

\[
\langle x \rangle = \int x \cdot p(x) \, dx. \tag{A.8}
\]

**Posterior median.** Defined such that

\[
\int_{-\infty}^{x'} p(x) \, dx = \int_{x'}^{\infty} p(x) \, dx = 0.5. \tag{A.9}
\]

This is invariant for monotonic transformations. The posterior median is calculated in a similar manner to a one-dimensional credible region.

**Posterior modes.** Parameter values (*i.e.* bin centres in marginalised pdf) such that \(p(x)\) or \(p(x, y)\) is maximised. Note that the modes of \(p(x)\) and \(p(y)\) may not coincide with the mode of \(p(x, y)\).

**Appendix A.2. Frequentist**

We briefly describe the frequentist statistical quantities in Superplot. Quantities based upon the profile likelihood are parameterisation invariant. See, *e.g.*, ref. [33] for further discussion.

**Profile likelihood.** Defined

\[
\mathcal{L}(x) = \max_{y, z, \ldots} \mathcal{L}(x, y, z, \ldots), \tag{A.10}
\]

\[
\mathcal{L}(x, y) = \max_{z, \ldots} \mathcal{L}(x, y, z, \ldots), \tag{A.11}
\]

in the one- and two-dimensional cases, respectively. Profile likelihood is calculated by binning the chain and finding the greatest likelihood within each bin.

**One-dimensional \(\Delta \chi^2\).** Defined

\[
\Delta \chi^2(x) \equiv -2 \ln \frac{\mathcal{L}(x)}{\max \mathcal{L}(x, y, z, \ldots)}. \tag{A.12}
\]

This may be chi-squared distributed by Wilks’ theorem [34].
Confidence intervals. Defined via Wilks’ theorem in the one-dimensional case as the region such that
\[ -2 \ln \frac{\mathcal{L}(x)}{\max \mathcal{L}(x, y, z, \ldots)} \leq F^{-1}_\chi^2(1 - \alpha) \]  
(A.13)
and in the two-dimensional case, as the region such that
\[ -2 \ln \frac{\mathcal{L}(x, y)}{\max \mathcal{L}(x, y, z, \ldots)} \leq F^{-1}_\chi^2(1 - \alpha) \]  
(A.14)
where \( F^{-1}_\chi^2 \) is the inverse cdf for a chi-squared distribution with \( n \) degrees of freedom and \( \alpha \) is a user-defined confidence level. We calculate whether each bin is excluded. Confidence intervals are not contiguous.

Best-fit point. Defined as the parameter value (i.e. not the bin centre) for the sample for which the likelihood is maximised.

Appendix B. Kernel density estimation

There are drawbacks to estimating an unknown pdf by histogramming samples. Histogramming typically results in a jagged distribution that may be aesthetically unappealing as well as a poor estimate of the unknown pdf. One must, furthermore, judiciously pick the number of bins and the bin limits, both of which could affect the reconstructed pdf. Kernel density estimation (KDE) is an alternative to histogramming (see, e.g., ref. [35]). One replaces every sample with a kernel function and estimates the pdf by a weighted sum of the kernels over all samples:
\[ p(y) = \sum_i w_i K \left( \frac{y - x_i}{h} \right) \]  
where \( \int K \left( \frac{y - x_i}{h} \right) dy = 1 \) and \( \sum_i w_i = 1 \).  
(B.1)
The function \( K \) is a kernel function and \( h \) is the so-called bandwidth. Superplot includes only a Gaussian kernel function in one or two dimensions, with a variance or covariance matrix calculated from the samples.

The bandwidth, which multiplies the standard deviation of Gaussian kernels, significantly impacts the estimate of the pdf. Indeed, picking the bandwidth is crucial: it could over-smooth, destroying genuine features of the unknown pdf, or under-smooth, and retain spurious noise in the estimate of the pdf. By default, Superplot uses Scott’s rule of thumb [36] for the bandwidth,
\[ h_{\text{Scott}} = \left( \frac{d + 2}{4} \right)^{-1/(d+4)} \]  
where \( n_{\text{eff}} = \frac{1}{\sum_i w_i^2} \),  
(B.2)
for \( d \)-dimensional KDE with an effective sample number \( n_{\text{eff}} \). This is optimal for Gaussian distributions but may perform poorly for non-Gaussian, multi-modal distributions. Furthermore, whilst we insure that the KDE cannot over-spill outside a user-specified range (the bin range), there is no special treatment of boundaries, which may result in underestimates at the edges of unphysical regions. The bandwidth can be altered via the \texttt{bw.method} variable in the \texttt{yaml} file config.yml, as described in sect. 3.1. Scott’s rule (\texttt{scott}), Silverman’s rule [37] (\texttt{silverman}),
\[ h_{\text{Silverman}} = \left( \frac{d + 2}{4} \right)^{-1/(d+4)} n_{\text{eff}} \approx \begin{cases} 1.06 \cdot h_{\text{Scott}} & d = 1 \\ h_{\text{Scott}} & d = 2 \end{cases} \]  
(B.3)
and user-defined bandwidths (e.g., 0.01) are supported.

KDE is implemented via a modified version of \texttt{scipy.gaussian.kde}\footnote{The modifications support weighted samples and utilise discrete Fourier transforms. Development began from ref. [38].}. We finely bin samples in evenly spaced bins such that eq. (B.1) becomes a discrete convolution,
\[ p(y) = \sum_b w(x_b) K \left( \frac{y - x_b}{h} \right) = w \ast K, \]  
(B.4)
where we sum over bins \( b \) and \( \cdot \ast \cdot \) denotes a discrete convolution. We evaluate the convolution by the convolution theorem,
\[ w \ast K = F^{-1} \{ F\{w\} \cdot F\{K\} \}, \]  
(B.5)
where \( F\{\cdot\} \) applies a discrete-time Fourier transformation (DTFT). This well-known trick (see, e.g., ref. [39]) reduces the complexity of the calculation, making it tractable for large numbers of samples, and is implemented in \texttt{scipy.signal.fftconvolve}.
Fig. 3. Weighted histograms with 30 bins per dimension contrasted with Gaussian kernel density estimation (KDE) with Scott’s rule of thumb. All plots were produced by Superplot from a publicly released chain from SuperBayeS [7,8].

To illustrate the pros and cons, in fig. 3 we plot posterior pdf with and without KDE. For a simple one-dimensional pdf in fig. 3(a) and fig. 3(b), KDE appears successful; the distribution is smoothed without distortion. The two-dimensional pdf in fig. 3(c) and fig. 3(d), however, is problematic: genuine small features, such as the $1\sigma$ credible regions, appear over-smoothed and exaggerated in fig. 3(d) and the boundaries at $m_0 = 0$ and $m_{1/2} = 0$ appear over-smoothed. On the other hand, spurious small features, such as the noise in the $2\sigma$ credible region, appear to be correctly smoothed away. In fig. 3(e) and fig. 3(f), however, the pdf from KDE appears satisfactory as the contours are smoothed without obliterating or exaggerating genuine details. We advise that KDE is used with care, and disable it by default.
References

1. J. Skilling, AIP Conf. Proc. 735, 395 (2004).
2. J. Skilling, Bayesian Anal. 1, 833 (2006).
3. F. Feroz, M.P. Hobson, Mon. Not. R. Astron. Soc. 384, 449 (2008) arXiv:0704.3704 [astro-ph].
4. F. Feroz, M.P. Hobson, M. Bridges, Mon. Not. R. Astron. Soc. 398, 1601 (2009) arXiv:0809.3437 [astro-ph].
5. F. Feroz, M.P. Hobson, E. Cameron, A.N. Pettitt, Importance Nested Sampling and the MultiNest Algorithm, arXiv:1306.2144 [astro-ph.IM].
6. A. Fowlie, Phys. Rev. D 90, 015010 (2014) arXiv:1403.3407 [hep-ph].
7. R. Ruiz de Austri, R. Trotta, L. Roszkowski, JHEP 05, 002 (2006) arXiv:hep-ph/0602028 [hep-ph].
8. R. Ruiz de Austri, R. Trotta, F. Feroz, SuperBayes: Supersymmetry Parameters Extraction Routines for Bayesian Statistics, http://www.ft.uam.es/personal/rruiz/superbayes/index.php?page=main.html, accessed September 2016.
9. K.J. de Vries et al., Eur. Phys. J. C 75, 422 (2015) arXiv:1504.03260 [hep-ph].
10. A. Lewis, S. Bridle, Phys. Rev. D 66, 103511 (2002) astro-ph/0205436.
11. A. Lewis, Phys. Rev. D 87, 103529 (2013), arXiv:1304.4473 [astro-ph.CO].
12. J. Zuntz, M. Paterno, E. Jennings, D. Rudd, A. Manzotti, S. Dodelson, S. Bridle, S. Sehrish, J. Kowalkowski, CosmoSIS: Modular Cosmological Parameter Estimation, arXiv:1409.3409 [astro-ph.CO].
13. G. Aslanyan, Comput. Phys. Commun. 185, 3215 (2014) arXiv:1312.4961 [astro-ph.IM].
14. M.J. Mortonson, H.V. Peiris, R. Easther, Phys. Rev. D 83, 043505 (2011) arXiv:1007.4205 [astro-ph.CO].
15. R. Easther, H.V. Peiris, Phys. Rev. D 85, 103533 (2012) arXiv:1112.0326 [astro-ph.CO].
16. J. Norena, C. Wagner, L. Verde, H.V. Peiris, R. Easther, Phys. Rev. D 86, 023505 (2012) arXiv:1202.0304 [astro-ph.CO].
17. M. Olamaie, F. Feroz, K.J.B. Grainge, M.P. Hobson, J.S. Sanders, R.D.E. Saunders, Mon. Not. R. Astron. Soc. 446, 1799 (2015) arXiv:1310.1885 [astro-ph.CO].
18. J. Buchner, A. Georgakakis, K. Nandra, L. Hsu, C. Rangel, M. Brightman, A. Merloni, M. Salvato, J. Donley, D. Kocevski, Astron. Astrophys. 564, A125 (2014) arXiv:1402.0004 [astro-ph.HE].
19. P. Scott et al., GAMBIT: The Global And Modular BSM Inference Tool, http://gambit.hepforge.org, accessed September 2016.
20. W.J. Handley, M.P. Hobson, A.N. Lasenby, Mon. Not. R. Astron. Soc. 450, L61 (2015) arXiv:1502.01856 [astro-ph.CO].
21. W.J. Handley, M.P. Hobson, A.N. Lasenby, Mon. Not. R. Astron. Soc. 453, 4384 (2015) arXiv:1506.00171 [astro-ph.IM].
22. J.D. Hunter, Comput. Sci. Eng. 9, 90 (2007).
23. P. Scott, Eur. Phys. J. Plus 127, 138 (2012) arXiv:1206.2245 [physics.data-an].
24. R. Lemrani, SuperEGO: SuperBayes Enhanced Graphical Output, http://www.ft.uam.es/personal/rruiz/superbayes/index.php?page=html/gui.htm, accessed September 2016.
25. S. Bridle, CosmoLog GUI, http://www.sarahbridle.net/cosmologui/, accessed September 2016.
26. R. Ruiz de Austri, R. Trotta, F. Feroz, GetPlots. http://www.ft.uam.es/personal/rruiz/superbayes/index.php?page=html/run.htm, accessed September 2016.
27. A. Lewis, S. Bridle, GetDist, http://cosmologist.info/cosmomc/doc/programs/GetDist.htm and http://cosmologist.info/cosmomc/readme_gui.html, accessed September 2016.
28. R. Brun, F. Rademakers, Nucl. Instrum. Methods A 389, 81 (1997).
29. S.v.d. Walt, S.C. Colbert, G. Varoquaux, Comput. Sci. Eng. 13, 22 (2011).
30. E. Jones, T. Oliphant, P. Peterson et al., SciPy: Open source scientific tools for Python, (2001), http://www.scipy.org/.
31. W. McKinney, Data structures for statistical computing in Python, in Proceedings of the 9th Python in Science Conference, edited by S. van der Walt, J. Millman (2010) pp. 51–56.
32. P. Gregory, Bayesian Logical Data Analysis for the Physical Sciences (Cambridge University Press, 2005).
33. F. James, Statistical Methods in Experimental Physics (World Scientific, 2006).
34. S.S. Wilks, Ann. Math. Stat. 9, 60 (1938).
35. J. Thompson, R. Tapia, Nonparametric Function Estimation, Modeling, and Simulation (SIAM, 1990).
36. D.W. Scott, Multivariate density estimation: Theory, practice, and visualization (John Wiley & Sons, 1992).
37. B. Silverman, Density Estimation for Statistics and Data Analysis, in Chapman & Hall/CRC Monographs on Statistics & Applied Probability (Taylor & Francis, 1986).
38. Weighted Gaussian kernel density estimation in Python, Stack Overflow, http://stackoverflow.com/q/27623919/.
39. G. Arfken, H. Weber, F. Harris, Mathematical Methods for Physicists: A Comprehensive Guide (Elsevier, 2012).