GetDist: a Python package for analysing Monte Carlo samples

Antony Lewis[1, *]

[1] *University of Sussex*

(Dated: October 30, 2019)

Monte Carlo techniques, including MCMC and other methods, are widely used and generate sets of samples from a parameter space of interest that can be used to infer or plot quantities of interest. This note outlines methods used the Python `GetDist` package to calculate marginalized one and two dimensional densities, which can be used for confidence contours, limits and plotting. In general inferring properties of the true distribution from samples is difficult, and fully Bayesian methods can be numerically expensive. We use non-parametric Kernel Density Estimation (KDE) techniques for estimating 1D and 2D densities, which are fast enough for interactive use. Many Monte Carlo methods produce correlated and/or weighted samples, for example produced by MCMC, nested, or importance sampling, and there can be hard boundary priors. `GetDist`'s baseline method consists of applying a linear boundary kernel, and then using multiplicative bias correction. The smoothing bandwidth is selected automatically following Botev et al. [1], based on a mixture of heuristics and optimization results using the expected scaling with an effective number of samples (defined to account for MCMC correlations and weights). Two-dimensional KDE use an automatically-determined elliptical Gaussian kernel for correlated distributions. The package includes tools for producing a variety of publication-quality figures using a simple named-parameter interface, as well as a graphical user interface that can be used for interactive exploration. It can also calculate convergence diagnostics, produce tables of limits, and output in latex.

I. INTRODUCTION

Monte Carlo (MC) sampling methods are widely used, from simulations to Markov Chain Monte Carlo (MCMC) sampling for Bayesian inference [2–4]. Once samples have been generated, many (but not all) quantities of interest can easily be estimated from the samples, including parameter means, confidence limits and marginalized densities. `GetDist` is a Python package for computing these and making publication-quality figures[1] . An example is shown in Fig. 1, taken from the Planck satellite cosmological parameter analysis for which the package was originally developed [5]. The MC samples here are drawn from the posterior distribution of cosmological parameters using MCMC, and are shown (thinned) as coloured points in the figure. The full example parameter space is 27-dimensional, but since marginalization from samples simply corresponds to ignoring parameters, the marginalized 1D and 2D densities are just proportional to the local (weighted) sample density in the subspace of interest. The 1D curves show estimates of the marginalized density for some individual parameters, and the 2D contours show contours of equal marginalized probability that contain different fractions of the total probability. `GetDist` provides a simple Python and graphical interface for calculating the required densities and producing figures (using MATPLOTLIB [6]); for many more examples see the plot gallery[2]. This note is not intended to document the `GetDist` package (see the documentation), but to collect some notes and references for what the code is doing.

To calculate marginalized probabilities, or make 1D probability or confidence/exclusion plots we need a way to estimate the underlying marginalized probability density from the distribution of samples. This is easy to do approximately by just constructing histograms, however it may be unclear how wide to make the bins and the result is unlikely to be an accurate representation of the density. A Bayesian approach could attempt to solve for the distribution of the true density given the samples (and a model of how they were drawn), for example using a Gaussian process prior [7–9]. While conceptually appealing and potentially very accurate, solutions typically involve a further step of MC sampling and can have non-trivial computational cost. There are also practical difficulties to making it very rigorous, for example one rarely has a good model for the exact sampling distributions of realistic MCMC chains. Instead we focus on fast and relatively simple conventional kernel density estimates, which effectively amount to using intelligently smoothed histograms with appropriately chosen smoothing widths. The main difficulties are how to choose the widths and how to handle hard boundary priors. In general the smoothing width could vary spatially, but for simplicity it is sufficient for many distributions of practical interest to use a fixed smoothing for each marginalized

*URL: `http://cosmologist.info`

[1] `https://getdist.readthedocs.io/`, install using `pip install getdist`. Source code at `https://github.com/cmbant/getdist/`

[2] `https://getdist.readthedocs.io/en/latest/plot_gallery.html`

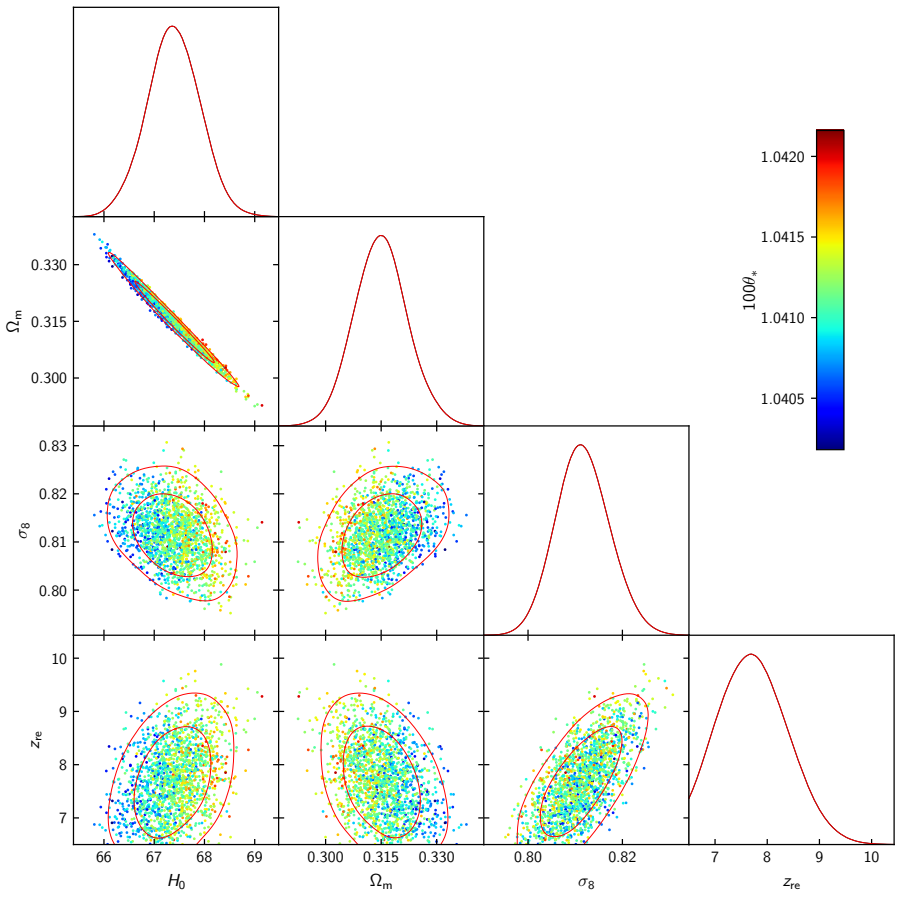

FIG. 1: Example `GetDist` 'triangle' plot of MCMC parameter samples, here taken from the Planck 2018 baseline cosmological parameter chains [5] (generated using the `CosmoMC` fast-parameter dragging sampler [13–15]). Thinned samples are shown as coloured points, where the colour corresponds to the θ_* parameter shown in the colour bar (which is marginalized out by projection in the 1 and 2D plots). The 1D plots and 2D density contours containing 68% and 95% of the probability are constructed from all of the samples using kernel density estimates as described in this note. Although relatively simple unimodal distributions, all the marginalized 2D distributions are somewhat non-Gaussian, there is a hard prior on the parameter $z_{\rm re} > 6.5$ which must be accounted for in the density estimates, and H_0 and $\Omega_{\rm m}$ are tightly correlated.

distribution as we assume. For MC samples, one option would be to generate so many samples that sampling noise is nearly negligible (so that a very narrow smoothing could be used), and this should of course be done when the sampling cost is low enough. However, using a good density estimate can dramatically reduce the number of samples that are required for a given target accuracy, potentially greatly reducing the computational cost of the MC needed to produce reliable results and nice figures.

`GetDist` can be used on independent single samples, but also has specific support for weighted samples and samples with substantial correlations as typically produced by MCMC algorithms. Weighted samples are the natural product of importance sampling, nested sampling [10–12], and various other sampling techniques. MCMC methods produce highly correlated samples, but once converged the chain of samples from standard Metropolis sampling and variants should be stationary. The MC sampling noise in general depends on both the correlations and the weights, so both need to be accounted for when estimating an appropriate kernel smoothing.

II. WEIGHTED SAMPLES

We present results for weighted samples $\boldsymbol{X}_i$ for generality, so each sample in the parameter space $\boldsymbol{x}$ of interest is associated with a weight w_i (which can be unity for unweighted samples). Estimators for the mean of a function $F(\boldsymbol{x})$

under the distribution $f(\boldsymbol{x})$ are then given by weighted sums over n sample points:

$$\hat{\bar{F}} = \frac{1}{N}\sum_{i=1}^{n} w_i F(\boldsymbol{X}_i), \tag{1}$$

where $N = \sum_{i=1}^{n} w_i$. Define $f_p(w_i, \boldsymbol{X}_i)$ as the probability of getting sample point $\boldsymbol{X}_i$ with weight w_i, taking points to be independent for now. The expected value of the estimator of the mean is then

$$\begin{aligned}\langle\hat{\bar{F}}\rangle &\approx \frac{1}{N}\sum_{i=1}^{n}\int \mathrm{d}\boldsymbol{X}_i\,\mathrm{d}w_i w_i F(\boldsymbol{X}_i) f_p(w_i,\boldsymbol{X}_i) = \frac{1}{N}\sum_{i=1}^{n}\int \mathrm{d}\boldsymbol{X}_i\, F(\boldsymbol{X}_i) f_p(\boldsymbol{X}_i)\int \mathrm{d}w_i w_i f_p(w_i|\boldsymbol{X}_i)\\ &= \frac{n}{N}\int \mathrm{d}\boldsymbol{X}_i\, F(\boldsymbol{X}_i) f_p(\boldsymbol{X}_i)\langle w(\boldsymbol{X}_i)\rangle_p,\end{aligned} \tag{2}$$

where here and below we neglect differences between $\langle N\rangle_p = n\langle w\rangle_p$ and N. The estimator will therefore be unbiased, $\langle\hat{\bar{F}}\rangle_p = \bar{F}$, if

$$\frac{\langle w(\boldsymbol{X}_i)\rangle_p f_p(\boldsymbol{X}_i)}{\langle w\rangle_p} = f(\boldsymbol{X}_i). \tag{3}$$

We shall assume weighted samples satisfy Eq. (3), which is true for importance weights where the weights are non-stochastic and given by $w(\boldsymbol{X}_i) = \alpha f(\boldsymbol{X}_i)/f_p(\boldsymbol{X}_i)$ for arbitrary constant α. It also holds for MCMC chains where $w_i \geq 1$ integer weights count the number of steps due to rejected proposals at each point so that $\langle w(\boldsymbol{X}_i)\rangle_p \propto f(\boldsymbol{X}_i)/f_p(\boldsymbol{X}_i)$, though in the case of MCMC the points are not independent (as discussed further below).

III. KERNEL DENSITY ESTIMATION (KDE)

Kernel Density Estimation (KDE) is the standard term for a wide class of non-parametric methods of estimating probability densities from samples, improving on simple histograms by making some weak assumptions about smoothness. There are many reviews of the basics (e.g. see Refs. [16–19]). For analysis of MCMC samples, we can continue to run the MCMC until we have 'enough' samples; for good convergence using standard criteria, typically $\mathcal{O}(1000)$ independent-equivalent samples (and even more KDE-equivalent samples, see Sect. III D). This is rather larger than many cases where when KDE is applied to small samples of data, and as such more accurate methods that can be unstable with smaller numbers of samples can be used successfully. I'll start by reviewing some of the basics, then discuss various complications due to boundaries and sample correlations, and then improved estimators using multiplicative bias correction.

The basic ingredient is an estimate $\hat{f}(x)$ of the form

$$\hat{f}(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^{n} K_h(\boldsymbol{x}-\boldsymbol{X}_i) \approx \frac{1}{n}\sum_{b} H_b(\boldsymbol{x}_b) K_h(\boldsymbol{x}-\boldsymbol{x}_b), \tag{4}$$

where $\{\boldsymbol{X}_i\}$ are the sampled points, with n samples in total. This is sometimes called the "Parzen–Rosenblatt" window estimator. The kernel K_h can be chosen in different ways; `GetDist` uses (slightly truncated) zero-centred Gaussians by default, with a width parameter h (or more generally a covariance). The width parameter h determines how broad the kernel is, and hence how smooth the estimated function is. In practice, assuming we are only interested in low-dimensional densities, to get good scaling with large number of samples, the samples $\{\boldsymbol{X}_i\}$ can be binned (finely compared to the scale of K_h), to give sample counts $H_b(\boldsymbol{x}_b)$ in a set of bins with centres $\boldsymbol{x}_b$ (with $n = \sum_b H_b$). Also evaluating $\hat{f}$ as a (finely) binned density we then have a simple convolution that is fast to evaluate using FFTs[3]:

$$\hat{f} \approx \frac{1}{n} H * K_h. \tag{5}$$

[3] Or directly if the number of points is relatively small. FFTs could also be replaced by fast gauss transforms (see e.g. `http://www.umiacs.umd.edu/~morariu/figtree/`)

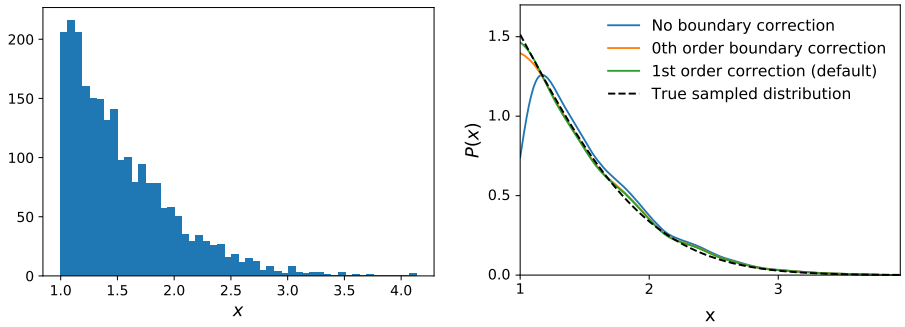

FIG. 2: Samples from a truncated Gaussian distribution with $x > 1$. The histogram is on the left and density estimates on the right using various different kernels (without multiplicative bias correction). Some form of boundary correction is essential in order not to severely underestimate the density near the boundary. The lowest-order correction removes the leading bias, but tends to underestimate any gradient at the boundary. In the case shown here the first-order correction works better than the zeroth order correction, but this is not guaranteed and higher-order methods can make the result less stable.

In general we have weighted samples, with each sample having a weight w_i each, in which case

$$\hat{f}(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{n} w_i K_h(\boldsymbol{x} - \boldsymbol{X}_i) \approx \frac{1}{N} H * K_h \tag{6}$$

where $N \equiv \sum_i w_i$, and H_b is now the weighted sum of the samples in each bin. In the continuum limit the histogram function is $H(\boldsymbol{x}) = \sum_i w_i \delta(\boldsymbol{x} - \boldsymbol{X}_i)$, and using Eq. (3) we have

$$\frac{1}{N}\langle H(\boldsymbol{x})\rangle = \frac{1}{N}\sum_i \int \mathrm{d}\boldsymbol{X}_i \mathrm{d}w_i f_p(w_i, \boldsymbol{X}_i) w_i \delta(\boldsymbol{x} - \boldsymbol{X}_i) = \frac{\langle w\rangle}{N}\sum_i \int \mathrm{d}\boldsymbol{X}_i f(\boldsymbol{X}_i)\delta(\boldsymbol{x} - \boldsymbol{X}_i) \approx f(\boldsymbol{x}),$$

where we drop the p subscript on the expectations $\langle\rangle$ where confusion should not arise. The KDE estimator of Eq. (6) therefore has expectation

$$\langle \hat{f}(\boldsymbol{x})\rangle = [K_h * f](\boldsymbol{x}), \tag{7}$$

which converges to $f(\boldsymbol{x})$ when K_h tends to a delta function as the kernel width goes to zero ($h \to 0$).

A. KDE bias and linear boundary kernels

Where there is a boundary, for example a prior on some parameter that it is positive, smoothing over the boundary will give biased results, since there are no samples on one side (see Fig. 2). Let's assume our function is of the form

$$f(\boldsymbol{x}) = B(\boldsymbol{x})\tilde{f}(\boldsymbol{x}) \tag{8}$$

where B is zero in the disallowed region, and one in the allowed region[4], and $\tilde{f}$ is a smooth function over the scale of the kernel (and equal to f where $B = 1$). Series expanding $\tilde{f}$ around $\boldsymbol{x}$ using its assumed smoothness $\tilde{f}(\boldsymbol{x} - \boldsymbol{\delta}) = \tilde{f}(\boldsymbol{x}) - \tilde{\boldsymbol{f}}^{(1)}(\boldsymbol{x}) \cdot \boldsymbol{\delta} + \ldots$, from Eq. (6) in the continuum limit we have

$$\begin{aligned}
\langle \hat{f}(\boldsymbol{x})\rangle &= \frac{1}{N}\int \langle H(\boldsymbol{x} - \boldsymbol{\delta})\rangle K_h(\boldsymbol{\delta}) d\boldsymbol{\delta} = \int B(\boldsymbol{x} - \boldsymbol{\delta})\tilde{f}(\boldsymbol{x} - \boldsymbol{\delta}) K_h(\boldsymbol{\delta}) d\boldsymbol{\delta} && (9)\\
&= \int B(\boldsymbol{x} - \boldsymbol{\delta})\left[\tilde{f}(\boldsymbol{x}) - \boldsymbol{\delta}\cdot\tilde{\boldsymbol{f}}^{(1)}(\boldsymbol{x}) + \frac{1}{2}\delta^i\delta^j\tilde{f}^{(2)}_{ij}(\boldsymbol{x})\ldots\right] K_h(\boldsymbol{\delta}) d\boldsymbol{\delta} && (10)\\
&= (K_h * B)\tilde{f}(\boldsymbol{x}) - (K^i_h * B)\tilde{f}^{(1)}_i(\boldsymbol{x}) + \frac{1}{2}(K^{ij}_h * B)\tilde{f}^{(2)}_{ij}(\boldsymbol{x}) + \ldots, && (11)
\end{aligned}$$

[4] B can be more general. Specifically, for the binned densities it can account for the fraction of the bin allowed by the prior (e.g. $B = 1/2$ for points where the prior cuts a bin in half). It could also account for other known locations of sharp features or structure [20]

where $K_h^{ijk\cdots}(\boldsymbol{x}) \equiv K_h(\boldsymbol{x})x^i x^j x^k \ldots$. Away from the boundary so that $B = 1$, we have $(K_h * B) = 1$ and $(K_h^i * B) = 0$ (for symmetric kernels), so the estimator is unbiased to linear order. The second order bias scales with the covariance of the kernel ($K_h^{ij} * B \to [\mathrm{cov}(K_h)]^{ij}$) and the local curvature of $\tilde{f}$, and describes the broadening of peaks by convolution (hence typically overestimation of the variance). In units of the width of f, the second order bias is $\mathcal{O}(h^2)$, and hence is small as long as the kernel is narrow enough compared to f.

With a boundary, the estimator is biased even at zeroth order. Normalizing by $(K_h * B)$ removes the leading bias, but leaves a linear bias if there is a non-zero gradient at the boundary (the convolution makes the shape at the boundary too flat). A simple solution to this is to use a linear boundary kernel [21]: using a non-symmetric kernel near the boundary to remove the bias. Starting with a simple symmetric kernel K_h, we can construct a more general kernel

$$K'_h(\boldsymbol{x}) = K_h(\boldsymbol{x})\left(A_0 + A_1^i x_i + \frac{1}{2}A_2^{ij}x_i x_j + \ldots\right), \tag{12}$$

and solve for coefficients $\{A\}$ to render the estimator unbiased. In one dimension this is straightforward to quadratic order[5], but it gets messy in more dimensions, and the multiplicative correction (described in Sect. III C) seems to be generally better at removing higher order biases. So here we restrict to linear kernels and set $A_{\geq 2} = 0$. We then have

$$\begin{aligned}
\langle\hat{f}(\boldsymbol{x})\rangle &= = \int B(\boldsymbol{x}-\boldsymbol{\delta})\tilde{f}(\boldsymbol{x}-\boldsymbol{\delta})K_h(\boldsymbol{\delta})\left(A_0 + A_1^i\delta_i + \ldots\right)d\boldsymbol{\delta} && (13)\\
&= \int B(\boldsymbol{x}-\boldsymbol{\delta})\left[\tilde{f}(\boldsymbol{x}) - \delta^i f_i^{(1)}(\boldsymbol{x}) + \ldots\right]K_h(\boldsymbol{\delta})\left(A_0 + A_1^j\delta_j + \ldots\right)d\boldsymbol{\delta} && (14)\\
&= \left[(K_h * B)A_0 + A_1^i(K_h^i * B)\right]\tilde{f}(\boldsymbol{x}) - \left[(K_h^i * B)A_0 + A_1^j(K_h^{ij} * B)\right]f_i^{(1)}(\boldsymbol{x}). && (15)
\end{aligned}$$

Solving for unit response to $\tilde{f}$ and zero gradient bias then gives

$$A_0 = \frac{1}{W_0 - W_1^i W_2^{ij} W_1^j} \qquad A_1^i = -[W_2^{-1}]^{ij}W_1^j A_0, \tag{16}$$

where $W_2^{ij} \equiv (K_h^{ij} * B)$, $W_1^i \equiv (K_h^i * B)$, $W_0 \equiv (K_h * B)$. The residual bias is then $\mathcal{O}(h^2)$, even approaching the boundary. Note that the correction kernel is only different from the starting kernel within a kernel width of the boundary, since $W_0 = 1, W_1 = 0$ for symmetric kernels where $B = 1$. However, for generality the terms can also be calculated by full convolutions.

One issue with the linear boundary kernel estimators is that they are not guaranteed to be positive. A simple fix is to impose positivity by using the positive estimate

$$\hat{f}_P \equiv \bar{f}\exp\left(\hat{f}/\bar{f} - 1\right), \tag{17}$$

where $\bar{f}$ is the simple de-biased kernel formed by normalizing by $(K_h * B)$ [23]. We also always renormalize so that the kernel density integrates to unity (or has peak normalized to one for convenient plotting). If $\hat{f}_P$ is only used as a pilot estimate for a later higher order estimator, accuracy of $\hat{f}_P$ near the boundary is in any case not critical.

B. Statistical and total error

To quantify the error in the kernel estimator, people often use the mean integrated squared error

$$\mathrm{MISE} \equiv \int \mathrm{d}\boldsymbol{x}\left\langle(\hat{f}(\boldsymbol{x}) - f(\boldsymbol{x}))^2\right\rangle, \tag{18}$$

largely because it is convenient to calculate analytically in simple cases. There are contributions from bias and statistical noise. Assume for simplicity there are no boundary priors here, so Eq. (11) gives the leading bias

$$\langle\hat{f}\rangle - f = \frac{1}{2}[\mathrm{Cov}(K_h)]^{ij}f_{ij}^{(2)} + \ldots. \tag{19}$$

---

[5] Giving a fourth order kernel, see e.g. [22]

To see the dependence on the smoothing scale h of the d-dimensional kernel, we can define the kernel as $K_h(\boldsymbol{x}) = \frac{1}{h^d}K(\boldsymbol{x}/h)$, and hence

$$[\mathrm{Cov}(K_h)]^{ij} = \frac{1}{h^d}\int \mathrm{d}^d\boldsymbol{x}x^i x^j K(\boldsymbol{x}/h) = h^2[\mathrm{cov}(K)]^{ij}. \tag{20}$$

The bias is independent of whether the samples are weighted or correlated. The statistical term is more tricky however. For now, just take the sample locations to be independent. Then (taking N to be non-stochastic)

$$\begin{aligned}\int \mathrm{d}\boldsymbol{x}\,\mathrm{var}\hat{f} &= \frac{1}{N^2}\int \mathrm{d}\boldsymbol{x}\sum_i\left\{\int \mathrm{d}\boldsymbol{X}_i\mathrm{d}w_i w_i^2 K_h^2(\boldsymbol{x}-\boldsymbol{X}_i)f_p(w_i,\boldsymbol{X}_i) - \left[\int \mathrm{d}\boldsymbol{X}_i\mathrm{d}w_i w_i f_p(w_i,\boldsymbol{X}_i)K_h(\boldsymbol{x}-\boldsymbol{X}_i)\right]^2\right\}\\ &= \frac{n\langle w^2\rangle}{N^2}\int \mathrm{d}\boldsymbol{x}' K_h^2(\boldsymbol{x}') - \frac{1}{N}\int \mathrm{d}\boldsymbol{x}\langle\hat{f}\rangle^2 = \frac{n\langle w^2\rangle}{N^2h^d}R(K) - \frac{1}{N}R(f) + \mathcal{O}(h^2/N),\end{aligned} \tag{21}$$

where $R(K) \equiv \int \mathrm{d}\boldsymbol{y}K^2(\boldsymbol{y})$. We can define an effective sample number

$$N_{\mathrm{eff}}^{\mathrm{indep}} \equiv \frac{N^2}{\sum_i w_i^2} = \frac{(\sum_i w_i)^2}{\sum_i w_i^2} \approx \frac{N^2}{n\langle w^2\rangle}, \tag{22}$$

so that the leading statistical variance scales $\propto 1/N_{\mathrm{eff}}^{\mathrm{indep}}$:

$$\int \mathrm{d}\boldsymbol{x}\,\mathrm{var}[\hat{f}(\boldsymbol{x})] \approx \frac{1}{N_{\mathrm{eff}}^{\mathrm{indep}}h^d}R(K) - \frac{1}{N}R(f) + \ldots. \tag{23}$$

For small h, the first term dominates.

The total mean integrated error of Eq. (18) is the sum of the bias and statistical terms, and evaluating the leading terms to get the asymptotic mean integrated squared error (AMISE) gives

$$\mathrm{AMISE} = \int \mathrm{d}\boldsymbol{x}\,\mathrm{var}\hat{f} + \int \mathrm{d}\boldsymbol{x}\langle\hat{f}-f\rangle^2 = \frac{1}{N_{\mathrm{eff}}^{\mathrm{indep}}h^d}R(K) - \frac{1}{N}R(f) + \frac{h^4}{4}\int \mathrm{d}\boldsymbol{x}\left([\mathrm{Cov}(K)]^{ij}f_{ij}^{(2)}\right)^2 + \ldots, \tag{24}$$

Minimizing this with respect to h gives $h \propto [N_{\mathrm{eff}}^{\mathrm{indep}}]^{-1/(4+d)}$, or explicitly an asymptotically-optimal kernel smoothing scale of

$$h = \left(\frac{R(K)d}{N_{\mathrm{eff}}^{\mathrm{indep}}I}\right)^{\frac{1}{4+d}}, \tag{25}$$

where $I \equiv \int \mathrm{d}\boldsymbol{x}\left([\mathrm{Cov}(K)]^{ij}f_{ij}^{(2)}\right)^2$.

Apart from the scaling with the effective number of samples $N_{\mathrm{eff}}^{\mathrm{indep}}$, h also scales with the curvature of f via the dependence on I: the larger the average squared second derivative, the more structure the gets smoothed out, and hence the smaller h should be. But remember that this is specific to the simple linear kernel estimator of Eq. (4), assuming independent sample points.

C. Multiplicative bias correction

Using boundary kernels renders estimates that are unbiased to $\mathcal{O}(h^2)$. However, there is still a systematic broadening of peaks, which can lead to systematically overestimated errors unless there are sufficiently many samples that $h \ll 1$. We can do better (or save computing time by generating fewer samples), by using a higher-order estimator.

Note that the simple estimator is exactly unbiased if the density is flat (or linear). We can therefore try to *flatten* the density before performing the convolutions. Specifically, doing the *multiplicative bias correction* to form

$$\hat{\hat{f}} = g(K_h * [H/g]), \tag{26}$$

where g is an approximation to the shape of f, so that H/g is nearly flat. Absent any prior information about the shape, the simplest thing to do is use $g = \hat{f}$, where $\hat{f}$ is a standard linear kernel density estimate; the $\hat{\hat{f}}$ estimator

then has bias $\mathcal{O}(h^4)$ away from boundaries (assuming sufficient smoothness of f) [24]. To improve the flattening near boundaries, we can take $\hat{f}$ to be the linear boundary kernel estimate from the Sect. III A. In principle the flattening can also be iterated, but for good choice of smoothing widths usually little is to be gained (and iterations will not converge due to random fluctuations being magnified). The simple multiplicative bias correction method compares well with other higher-order kernel methods for many distributions [22] and seems to work well in practice as long as the density is indeed sufficiently smooth. In principle different bandwidths can be used for the pilot estimator g and the final estimate $\hat{f}$ (see e.g. [25] who recommend g is over-smoothed compared to $\hat{f}$), but for simplicity we take them to be the same. Other approaches to bias reduction are possible, including the 'data sharpening' [26, 27] method, which is a special case of a more general diffusion approach [1].

Using multiplicative bias correction often generates nice smooth densities even with relatively small numbers of samples. However, note that this can make the sampling error on the result less immediately visually apparent when plotted than when using lower-order estimates that result in more obviously unsmooth densities. `GetDist` allows the multiplicative correction order to be changed as desired, but is set to first order by default (doing multiplicative bias correction once).

D. Correlated samples

In reality, samples from MCMC are correlated. Expressed in weighted form (where weights count the rejections of the next proposal), there are non-trivial weights and correlations between chain positions. Correlations will increase the error. However, perhaps surprisingly, correlations don't have a dramatic effect on the optimal kernel bandwidth: the main impact of correlation on the variance does not scale with h, since correlated errors between nearby $\boldsymbol{x}$ cannot be lowered by more smoothing; see Refs. [28, 29].

Eq. (24) for the error using $N_{\rm eff}^{\rm indep}$ independent weighted samples (equivalent to Ref. [29]) cannot be the full story when correlated samples are used with finite h of practical interest. For example, proposals in orthogonal subdimensions could leave the parameter(s) of interest exactly unchanged between steps even though they appear as different points in the full-dimensional parameter space. This could be remedied by using a parameter-dependent $N_{\rm eff}^{\rm indep}$ in Eq.(24), where the weights now count all consecutive identical points in the parameter space of the kernel density. However, it is also clear that very small changes in a parameter, for example due to accepted proposals along very nearly orthogonal eigendirections, should contribute nearly the same as exactly identical points. In other words, whether or not the correlation matters when determining the bandwidth depends on the shape of the correlation function; e.g., whether there is high probability for $|X_i - X_{i+k}| < h\sigma_X$, or whether the distribution is broad compared to $h\sigma_X$.

In detail, we have

$$\int \mathrm{d}\boldsymbol{x}\, \hat{f}^2(\boldsymbol{x}) = \frac{1}{N^2}\int \mathrm{d}\boldsymbol{x} \sum_{i,j} w_i w_j K_h(\boldsymbol{x}-\boldsymbol{X}_i)K_h(\boldsymbol{x}-\boldsymbol{X}_j) = \frac{1}{N^2}\sum_{i,j}\; w_i w_j [K_h * K_h](\boldsymbol{X}_i - \boldsymbol{X}_j). \tag{27}$$

Assuming stationarity[6] leads to

$$\int \mathrm{d}\boldsymbol{x}\, \langle \hat{f}^2(\boldsymbol{x})\rangle = \frac{n\langle w^2\rangle}{N^2}\int \mathrm{d}\boldsymbol{x}' K_h^2(\boldsymbol{x}') + \frac{2}{N^2}\sum_{k=1}^{n-1}(n-k)\langle w_i w_{i+k}[K_h * K_h](\boldsymbol{X}_i - \boldsymbol{X}_{i+k})\rangle, \tag{28}$$

and transforming $K_h \to K$ then gives

$$\int \mathrm{d}\boldsymbol{x}\, \langle \hat{f}^2(\boldsymbol{x})\rangle = \frac{R(K)}{N_{\rm eff}^{\rm indep} h^d} + \frac{2}{N^2 h^d}\sum_{k=1}^{n-1}(n-k)\left\langle w_i w_{i+k}[K*K]\left(\frac{\boldsymbol{X}_i - \boldsymbol{X}_{i+k}}{h}\right)\right\rangle. \tag{29}$$

This makes it clear that the result depends on the number of sequences of points within distance h of each other, as determined by the local $K * K$ filter. Note that the last term in Eq. (29) contains a large contribution $\int\langle\hat{f}(\boldsymbol{x})\rangle^2$ from points that have close to the same value (a fraction $\mathcal{O}(h/n)$ of the terms in the sum), even in the absence of correlations. If points separated by $k \lesssim \Delta$ are strongly correlated with $P(\boldsymbol{X}_{i+k}|\boldsymbol{X}_i) \sim \delta(\boldsymbol{X}_{i+k} - \boldsymbol{X}_i)$, the second

[6] Note that this is not valid for the output of nested sampling and other dynamic sampling methods; in these cases `GetDist` currently simply treats the samples as independent, which could be improved in future.

term also has a contribution $\sim R(K)\Delta/Nh^d$ that is the same order as the first; this limit is what is considered in Ref. [29], and accounts for rejection steps that leave the parameter value exactly unchanged.[7] In general we define an effective number of samples, dependent on which parameter subspace is included in the dimensions of $\boldsymbol{X}$, estimated from the samples, as[8]

$$N_{\mathrm{eff},X}^{\mathrm{KDE}} \equiv \frac{N^2}{\sum_i w_i^2 + 2R(K)^{-1}\sum_k \sum_i \left(w_i w_{i+k}[K*K]([\boldsymbol{X}_i - \boldsymbol{X}_{i+k}]/h) - \hat{\mu}_K\right)}, \tag{32}$$

where the sum over k can be taken only up to order of the correlation length ($\mathcal{O}(L_X^s)$) where the terms are significantly non-zero (and hence is reasonably fast to evaluate), and $\mu_K \equiv \langle w_i w_j [K*K]([\boldsymbol{X}_i - \boldsymbol{X}_j]/h)\rangle$ takes out the $\sim \langle\hat{f}\rangle^2$ contribution expected for uncorrelated samples (estimated here roughly by a sum over widely separated small subset of samples). In the $h \to 0$ limit this definition therefore isolates the term that contributes to the total variance as

$$\int \mathrm{d}\boldsymbol{x}\, \mathrm{var}\hat{f}(\boldsymbol{x}) \approx \frac{1}{N_{\mathrm{eff},X}^{\mathrm{KDE}} h^d} R(K) + \mathcal{O}(1/N), \tag{33}$$

and hence includes the effect of exactly duplicated samples from MCMC rejection. The definition of Eq. (32) obeys consistency under sample-splitting, so it does not matter how samples are grouped up in to weighted samples or split up, and for uncorrelated samples reduces to Eq. (22). More generally, Eq. (32) very roughly includes other tight short-range correlation effects from MCMC sampling (but also some additional covariance that is actually mostly h-independent, which ideally should not affect the bandwidth choice). As defined $N_{\mathrm{eff},X}^{\mathrm{KDE}}$ does however itself depend on h. We take a fiducial value $h \approx 0.2\sigma$ for estimating $N_{\mathrm{eff},X}^{\mathrm{KDE}}$. Values of $N_{\mathrm{eff},X}^{\mathrm{KDE}}$ typically lie between $N_{\mathrm{eff}}^{\mathrm{indep}}$ and the $N_{\mathrm{eff},X}^{\mathrm{var}}$ defined in Eq. (49) below that determines the sampling errors on parameter means.

E. Choice of kernel bandwidth

A good choice of kernel width is import to get good results: too broad, and features are washed out; too narrow, and sampling noise shows up. Recall from Eq. (24) that the Parzen—Rosenblatt estimator has bias $\mathcal{O}(h^2)$, and the statistical variance goes as $\mathcal{O}([Nh]^{-1})$. Minimizing with respect to h gave $h \propto N^{-1/5}$ (1D case of Eq. (25), corresponding to an overall convergence rate $\propto N^{-4/5}$). The constant in the optimal width depends on the distribution (and kernel); assuming one dimension and Gaussians gives the rule of thumb for parameter X ('normal scale rule'):

$$h = 1.06\hat{\sigma}_X (N_{\mathrm{eff},X}^{\mathrm{KDE}})^{-1/5}, \tag{34}$$

where h is the standard deviation of the Gaussian smoothing Kernel to use and $\hat{\sigma}_X$ is an estimate of the standard deviation of parameter X. In practice, for potentially non-Gaussian densities, $\hat{\sigma}_X$ can be set from a variety of scale measures, for example a width based on central quantiles to avoid over-estimation due to broad tails or a more refined method based on order statistics [30]. However, simple scale rules can be quite suboptimal for many non-Gaussian densities. We only use a scale rule as a fallback when other methods fail and for choosing a fiducial scale for evaluating Eq. (32). To estimate $\hat{\sigma}_X$ we follow a simplified version of Ref. [30] (taking $\hat{\sigma}_X = \min[\sigma_X, R_{0.4}/1.048]$, where R_x is a

---

[7] Note that if the (integer) weights are from chain rejections during MCMC (but we neglect correlations between accepted points), then $P(w) = (1-a)^{w-1}a$, where a is the acceptance probability. Evaluating expectations gives

$$\langle w\rangle = \frac{1}{a}, \qquad \langle w^2\rangle = \frac{2-a}{a^2}. \tag{30}$$

So for raw chains, neglecting correlations between accepted points, we have

$$N_{\mathrm{eff}}^{\mathrm{indep}} \approx \frac{n\langle w\rangle^2}{\langle w^2\rangle} \approx \frac{N\langle w\rangle}{\langle w^2\rangle} \approx N\frac{a}{2-a}. \tag{31}$$

This relates results in terms of weights to results in the literature terms of acceptance probability (e.g. Ref. [29]).

[8] It is often a good approximation to estimate the 2D result from the separate 1D results; in `GetDist` there is an option whether to use the 2D expression or not (`use_effective_samples_2D`). Dependence of the optimal smoothing scale on $N_{\mathrm{eff},X}^{\mathrm{KDE}}$ is quite weak, so a ballpark number is sufficient in most cases. A more optimal bandwidth estimator would not use a single $N_{\mathrm{eff},X}^{\mathrm{KDE}}$ but account for anisotropy in the sampling statistics for sampling methods where different parameters are treated qualitatively differently or have different diffusion rates.

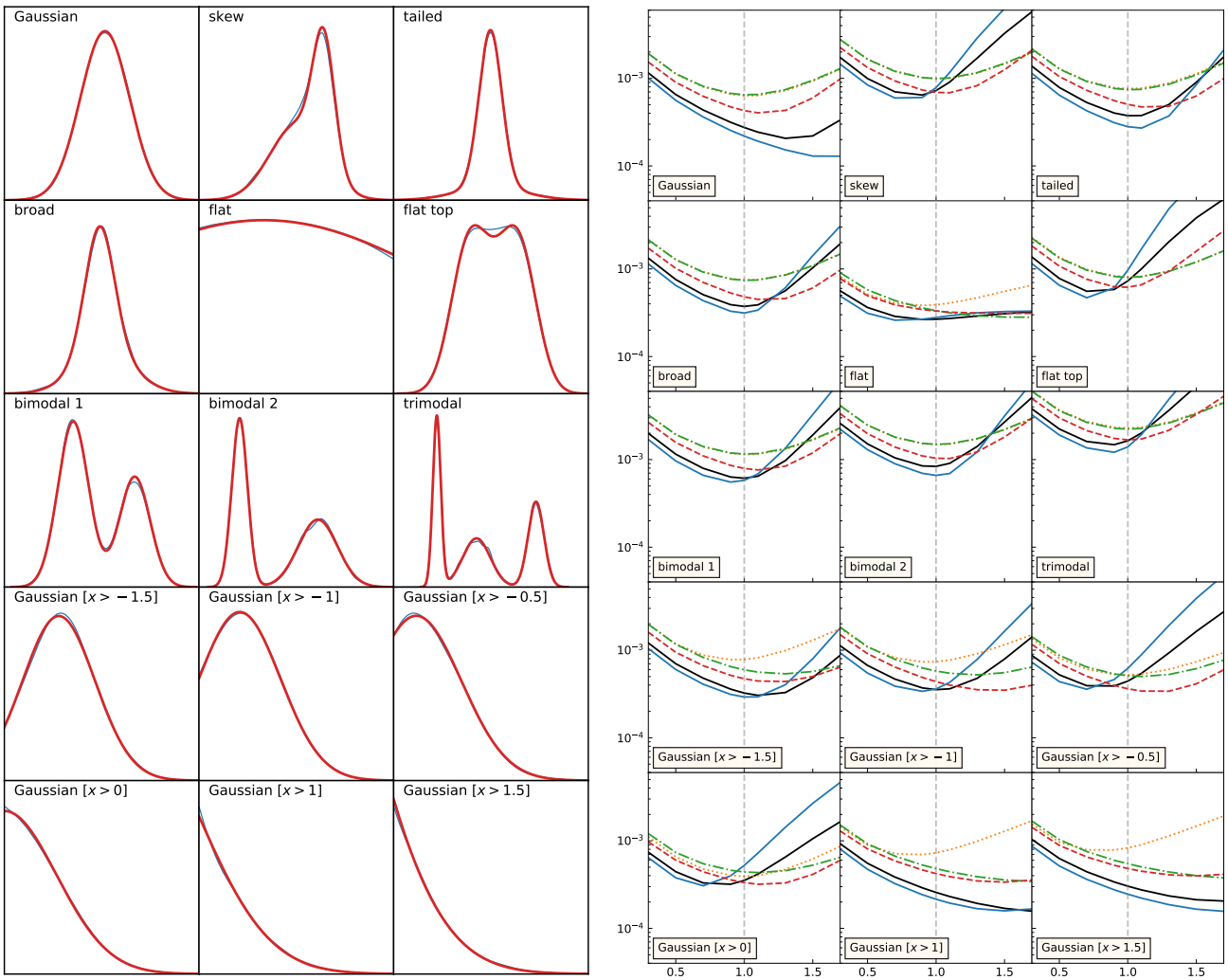

FIG. 3: *Left*: a set of test Gaussian-mixture distributions, comparing the true distribution (red) with the density estimate using 10000 independent samples (blue) using multiplicative bias correction and a linear boundary kernel. The Gaussian panels at the bottoms are truncated Gaussian distributions, and all distributions are normalized by the maximum value. *Right*: scaling of the average integrated squared error $\langle \int \mathrm{d}x(\Delta f(x))^2 \rangle / \int \mathrm{d}x(f(x))^2$ of the density estimate, where the average is estimated using 1000 sets of 10000 samples for each distribution. The x-axis is a scaling away from the automatically chosen kernel width (e.g. by Eq. (35)), so that one corresponds to the performance with default settings. Lines compare different kernel estimates: solid lines use a multiplicative bias correction (MBC) and linear boundary kernel (black: default, blue: next-order multiplicative bias correction). Dotted is the basic Parzen kernel (for which the kernel-width estimator is optimizing), dot-dashed is with linear boundary correction, and dashed is using a second-order boundary-corrected kernel. The MBC kernel width is sub-optimally chosen for Gaussian, where the leading bias term happens to be zero, but about right in many other cases.

the smallest parameter range enclosing x of the probability ($\min R_{0.4} = 1.048$ for a unit Normal) and searching over ranges starting at $p = 0, 0.1, 0.2...0.6$).

An optimal bandwidth choice can be derived using Eq. (25). The only problem here is that the optimal bandwidth depends on second derivatives (I) of the (unknown) density f. Replacing the derivative term with an estimator gives so-called 'plug-in' methods, which can perform much better especially for multi-modal distributions. For reviews and variations of methods see e.g. [1, 31–33]. The main problem is that to estimate the second derivative you need to use a bandwidth, which gives you a recursive unknown bandwidth problem. Ref. [1] present a neat solution, where the optimal bandwidth is obtained as an equation fixed point that can be found numerically called the "Improved Sheather-Jones" (ISJ) estimate. Using a Discrete Cosine Transform (DCT), this can also efficiently handle leading-order boundary effects, so that boundaries are not mistaken for large derivatives [1]. The method only requires one DCT of the binned data and some binned array dot products, and hence is fast; we adopt it as our auto-bandwidth

selector[9]. The DCT imposes even symmetry about boundaries, so we only use it for the bandwidth choice, not the actual KDE (the linear boundary kernel gives better accuracy by allowing general gradients at the boundaries).

With multiplicative bias correction the bias is higher order, with bias $\mathcal{O}(h^4)$ away from boundaries, so the total error scales as $Ah^8 + B/(Nh)$. Optimization now gives $h \propto N^{-1/9}$ and overall convergence $\propto N^{-8/9}$. Again the proportionality constant will depend on the distributions, various examples are given in Ref. [34]. As a first guess we take the one-dimensional[10] rule of thumb

$$h = h_{\rm ISJ}(N_{{\rm eff},X}^{\rm KDE})^{1/5-1/9}. \tag{35}$$

These smoothing widths are larger than for the basic Parzen–Rosenblatt estimator, and have lower statistical noise since the basic estimator is forced to have smaller widths to avoid significant bias. For $N_{{\rm eff},X}^{\rm KDE} \sim \mathcal{O}(1000)$, the smoothing width is about twice as broad as the basic estimator. A more refined estimate could be made analogously to the ISJ method using the asymptotic error for the higher order method, but we have not attempted to implement this. Eq. (35) is somewhat too small for normal distribution (which happens to give zero leading bias for this estimator [24]), but somewhat too large for some truncated Gaussian shapes. See Fig. 3 for test results on various test distributions[11]. Higher order bias correction can perform better, but starts to be more sensitive to having the bandwidth chosen optimally; as a default we use multiplicative boundary correction without iteration, which (in the test distributions) is almost always better than the Parzen estimator even when the auto-selected bandwidth is not optimal. In some cases using second order (once-iterated) multiplicative bias correction can give additional improvement. The `GetDist` package has settings options to tune exactly which method is use if required.

Multivariate bandwidth matrix

For two-dimensional densities we define the Kernel (following e.g. Ref. [35]) in terms of an isotropic Gaussian kernel $K(\boldsymbol{x}) = K(|\boldsymbol{x}|)$ and a kernel matrix $\boldsymbol{M}$, with $K_M(\boldsymbol{x}) = |\boldsymbol{M}|^{-1/2}K(\boldsymbol{M}^{-1/2}\boldsymbol{x})$, so that

$$\int {\rm d}\boldsymbol{x}\, {\rm var}\hat{f}(\boldsymbol{x}) \approx \frac{1}{N_{{\rm eff},X}^{\rm KDE}}|\boldsymbol{M}|^{-1/2}\int {\rm d}\boldsymbol{y}K^2(\boldsymbol{y}) + \cdots \equiv \frac{1}{N_{{\rm eff},X}^{\rm KDE}}|\boldsymbol{M}|^{-1/2}R(K) + \ldots. \tag{36}$$

If $K(\boldsymbol{x})$ has identity covariance, ${\rm cov}(K) = \boldsymbol{I}$, then $\boldsymbol{M}$ is just the covariance of $K_M(\boldsymbol{x})$ and hence

$${\rm AMISE} \approx \frac{1}{N_{{\rm eff},X}^{\rm KDE}}|\boldsymbol{M}|^{-1/2}R(K) + \frac{1}{4}\int {\rm d}\boldsymbol{x}\left(M^{ij}f_{ij}^{(2)}\right)^2 + \ldots \tag{37}$$

If we parameterize the Gaussian kernel covariance as $\boldsymbol{M} = \begin{pmatrix} h_x^2 & ch_xh_y \\ ch_xh_y & h_y^2 \end{pmatrix}$, Eq. (37) becomes

$${\rm AMISE} \approx \frac{1}{4N_{{\rm eff},X}^{\rm KDE}\pi h_xh_y\sqrt{1-c^2}} + \frac{1}{4}\left[h_x^4\psi_{4,0} + h_y^4\psi_{0,4} + 2h_x^2h_y^2(2c^2+1)\psi_{2,2} + 4ch_xh_y(h_x^2\psi_{3,1} + h_y^2\psi_{1,3})\right], \tag{38}$$

where we defined ψ_{m_1,m_2} as

$$\psi_{m_1,m_2} \equiv (-1)^{i+j}\int {\rm d}\boldsymbol{x}\left(\frac{\partial^{i+j}}{\partial x_1^i\partial x_2^j}f(\boldsymbol{x})\right)\left(\frac{\partial^{p+q}}{\partial x_1^p\partial x_2^q}f(\boldsymbol{x})\right) = \int {\rm d}\boldsymbol{x}f(\boldsymbol{x})\left(\frac{\partial^{p+q+i+j}}{\partial x_1^{p+i}\partial x_2^{q+j}}f(\boldsymbol{x})\right) \tag{39}$$

assuming no boundary terms, where $m_1 = p + i$ and $m_2 = q + j$ and $m_1 + m_2$ is even. For m_1 and m_2 both even, ψ_{m_1,m_2} (and corresponding bandwidths) can be estimated following the fixed-point method[12] of Ref. [1], where we

---

[9] There can be multiple or no solutions to the fixed-point equation, esp. with some very flat bounded distributions. When multiple solution we take the larger one, and where no solutions we use the fallback of Eq. (34).

[10] In general we can replace 1/9 with $1/(4p+1)$ for a higher order estimator where the leading bias goes as h^{2p}.

[11] The code describing the exact distributions and for reproducing the figures is at `https://github.com/cmbant/getdist/blob/master/getdist/tests/test_distributions.py`,

[12] When there is no solution for the fixed point, we instead use a plugin estimate for the bandwidth used for estimating ψ_{m_1,m_2}.

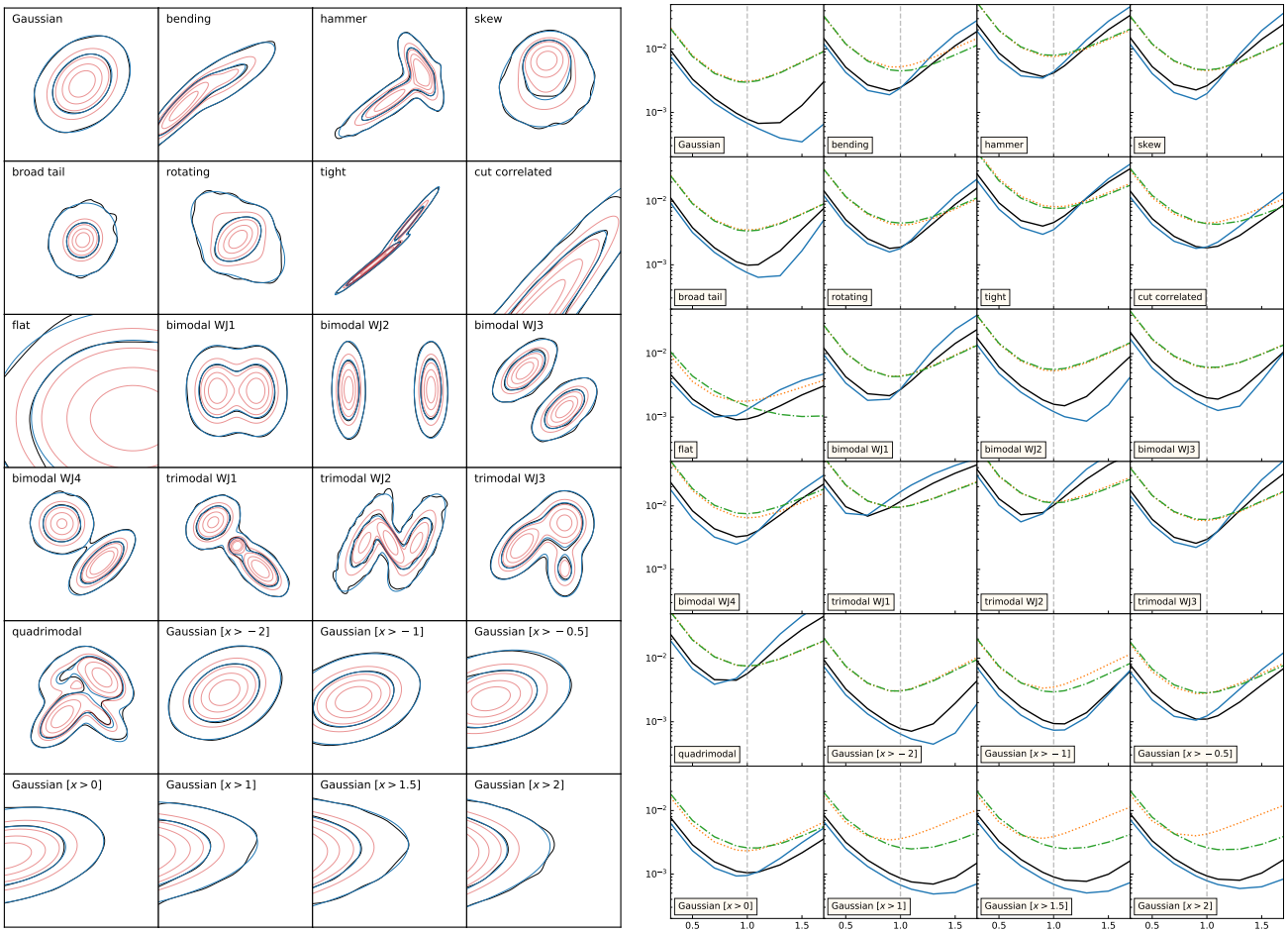

FIG. 4: *Left*: A set of 2D Gaussian mixture distributions (WJx labels are the same test distributions as Ref. [35]), comparing true density contours (enclosing the 68% and 95% of the probability) with contours from density estimation using one set of 10000 samples. *Right*: normalized average integrated squared error $\langle \int \mathrm{d}x(\Delta f(x))^2 \rangle / \int \mathrm{d}x(f(x))^2$ from 500 simulations of 10000 samples, as Fig. 3. In all but two of the trimodal examples the black lines (default is scale width one, with multiplicative bias correction and linear boundary kernel) give substantially lower error than the basic Parzen estimator (red dotted) and are more stable than the higher order bias correction (blue).

assume an isotropic Gaussian kernel for evaluating ψ_{m_1,m_2}. For the odd elements, the analogous argument to Ref. [1] (Appendix E) using Eq. 3.2 from [36] gives an equation for the bandwidth for estimating ψ_{m_1,m_2} as

$$h_{m_1,m_2} = \left(\frac{8(1-2^{-m_1-m_2-1})}{3(N_{\mathrm{eff},X}^{\mathrm{KDE}})^2} \frac{\psi_{0,0} R(K^{(m_1,m_2)})}{(\psi_{m_1,m_2+2} + \psi_{m_1+2,m_2})^2} \right)^{1/(2m_1+2m_2+6)} \tag{40}$$

where

$$R(K^{(m_1,m_2)}) = \frac{(2m_1-1)!!(2m_2-1)!!}{2^{m_1+m_2+2}\pi} \tag{41}$$

and $\psi_{0,0}$ can be estimated using the method for even elements.

In the case that the correlation c is zero, Eq. (38) can be optimized analytically to give [36]

$$h_x = \left[\frac{\psi_{0,4}^{3/4} R(K)}{\psi_{4,0}^{3/4}(\psi_{4,0}^{1/2}\psi_{0,4}^{1/2} + \psi_{2,2}) N_{\mathrm{eff},X}^{\mathrm{KDE}}} \right]^{1/6} \qquad h_y = (\psi_{4,0}/\psi_{0,4})^{1/4} h_x. \tag{42}$$

In the general correlated case the minimum must be found numerically. In the specific case that the target distribution

is Gaussian, the optimal Gaussian bandwidth matrix covariance is [35]

$$\boldsymbol{M} = \boldsymbol{C}(N_{\mathrm{eff},X}^{\mathrm{KDE}})^{-1/3}, \tag{43}$$

where $\boldsymbol{C}$ is the sample covariance. This can be used to define a rule of thumb for Gaussian-like distributions, but in general (especially in the multi-modal case) can be very bad.

There are several other issues here

- The bias term in Eq. (38) is not guaranteed to be positive if $c \neq 0$, so numerical minimization can fail. (see Ref. [37] for a possible alternative solution)
- With boundaries, the even ψ_{m_1,m_2} derivative terms can be approximated by imposing reflection boundary conditions (i.e. evaluating using DCT), but with m_1 or m_2 odd, ψ_{m_1,m_2} cannot be evaluated from the DCT transform (which assumes symmetry by construction). They can be evaluated by FFT if there are no sharp boundaries, but there is no easy way to approximately account for boundaries in this case.
- Since the ψ_{m_1,m_2} are evaluated using isotropic Gaussian kernels, they may be rather inaccurate if the optimal kernel is strongly elliptical.

We therefore adopt the following strategy:

- Assuming there are no boundaries, or a boundary in only one of the x or y directions (but not both), use the sample covariance to perform a Cholesky parameter rotation to define uncorrelated transformed variables. The Cholesky rotation is chosen so that if x or y has a boundary it remains unchanged, so the boundary in the transformed parameters remains parallel to the edge of the DCT box. The transformed samples are scaled (so roughly isotropic) and binned, so that evaluation of ψ_{m_1,m_2} using an isotropic kernel is not too suboptimal.
- If there is a boundary the even derivatives are evaluated following Ref. [1] by DCT, and the optimal diagonal bandwidth matrix evaluated from Eq. (42). This is then rotated back to the original coordinates.
- If there are no boundaries, the even and odd ψ_{m_1,m_2} derivatives are estimated, and (38) is minimized numerically. If this fails, Eq. (42) is used as a fall back. The bandwidth matrix is then rotated back to the original coordinates.
- If there are boundaries in both the x and y directions, a Cholesky rotation cannot preserve both boundaries, so the samples are not transformed. The diagonal form of Eq. (42) is evaluated on the untransformed samples, unless the sample correlation is very high, in which case a Gaussian rule of thumb bandwidth is assumed using the sample covariance. When there are boundaries the fixed-point solution for the moment bandwidth can give solutions that are substantially too large, in which case we fall back to a rule of thumb for the moment bandwidth.

The expected asymptotic scaling of the optimal bandwidths are $h \propto N^{-1/6}$ and $h \propto N^{-1/10}$ respectively for methods with quadratic and quartic bias. With multiplicative bias correction we therefore scale the elements of the bandwidth matrix determined above to give

$$h_{x,y} = 1.1 h_{x,y}^{\mathrm{ISJ}} (N_{\mathrm{eff}})^{1/6-1/10}, \tag{44}$$

where the 1.1 factor is empirically chosen. (In general we can replace $1/10$ with $1/(2p+2)$ for a higher order estimator where the leading bias goes as h^{2p}). See Fig. 3 for performance on typical distributions, showing that Eq. (44) slightly underestimates the bandwidth for a Gaussian distribution (and tail-truncated Gaussians), but is a reasonable compromise for most other cases and gives significant performance gains compared to the basic Parzen estimator.

IV. CORRELATION LENGTHS AND SAMPLING ERROR ON PARAMETER MEANS

From MCMC, possibly with post importance-sampling, the samples generally have non-trivial weights and non-trivial correlations. Consider a sample estimate for the mean $\bar{X}$ of a parameter X, given by

$$\hat{X} = \frac{1}{N}\sum_{i=1}^{n} w_i X_i. \tag{45}$$

From independent unit-weight samples, the variance of the mean estimator is σ_X^2/N; we can use this to define an effective $N_{\rm eff,X}^{\rm var}$ for the correlated weighted samples. The variance of $\hat{X}$ is given by

$$\langle(\hat{X}-\bar{X})^2\rangle = \frac{1}{N^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\langle w_i(X_i-\bar{X})w_j(X_j-\bar{X})\rangle. \tag{46}$$

Defining $d_i \equiv w_i(X_i-\bar{X})$, for chains in equilibrium we should have $\langle d_i d_j\rangle = C_d(|i-j|)$, where $C_d(k)$ is the autocorrelation function at lag k. Using this

$$\langle(\hat{X}-\bar{X})^2\rangle \;=\; \frac{1}{N^2}\left[nC_d(0)+2\sum_{k=1}^{n-1}(n-k)C_d(k)\right]. \tag{47}$$

If we assume that the correlation length is much shorter than the chain length[13], so $k \ll n$ for terms which matter, this is

$$\langle(\hat{X}-\bar{X})^2\rangle \approx \frac{n}{N^2}\left[C_d(0)+2\sum_{k=1}^{\infty}C_d(k)\right]. \tag{48}$$

We define this to be equal to $\sigma_X^2/N_{\rm eff,X}^{\rm var}$ so that

$$N_{\rm eff,X}^{\rm var} \approx \frac{N^2\sigma_X^2}{n[C_d(0)+2\sum_{k=1}^{\infty}C_d(k)]} \tag{49}$$

We can also define a correlation length by

$$L_d^w \equiv \frac{n}{N\sigma_X^2}\left[C_d(0)+2\sum_{k=1}^{\infty}C_d(k)\right], \tag{50}$$

so that $N_{\rm eff,X}^{\rm var} = N/L_X^w$. For unweighted samples L_X^w corresponds to the standard definition of the correlation length. For importance sampled chains, it is the length in 'weight units' (it scales with the arbitrary normalization of the importance weights). We can also define a correlation length L_X^s in 'sample units', so that $N_{\rm eff,X}^{\rm var} = n/L_X^s$, which gives an idea of how independent the different points are. In practice, to avoid sampling noise the upper limit for the sum is taken to be lag the lag at which the correlation has fallen to below some value (e.g. 0.05).

To estimate the error on Monte Carlo means, Eq. (49) can be estimated quickly using weighted sample convolutions and allows for both correlations and importance weights. In general there are correlations between parameters, so this is just an estimate for a single parameter, and will in general be optimistic (an upper limit).

V. DISCUSSION

Although `GetDist` 1.0 seems to work well for many simple cases, there are several assumptions and caveats. Future work could develop a kernel estimator optimized for other more-general non-stationary sampling distributions (e.g. the results of nested sampling). The boundary corrections as currently implemented require hard boundaries to be aligned with the parameter coordinates, which could be generalized to allow for an arbitrary prior mask; the leading-order correction would be a relatively straightforward extension, but optimization would require more work. The kernel optimization also currently does not account for the additional errors due to boundary correction, and may be suboptimal. More generally there may be non-trivial topology of the likelihood surface, for example highly multi-modal distributions with different characteristic scales on different modes. Although the basic methods described here will still work at some level, a global smoothing kernel is likely to be very suboptimal and there may be significant gains from implementing a more general method. The current implementation also only optimizes kernel widths for estimation of the density; for computation of integrals, such as tail confidence limits, this could be generalized. The code is open source on GitHub so contributions are welcome.

---

[13] Actually we don't need to do this, the finite estimator for the autocorrelation from the samples follows the original expression.

VI. ACKNOWLEDGEMENTS

I thank Jesus Torrado for work on the Cobaya interface and him and other github users for contributions. I acknowledge support from the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement No. [616170] and support by the UK STFC grant ST/P000525/1.

[1] Z. I. Botev, J. F. Grotowski, and D. P. Kroese, Ann. Statist. **38**, 2916 (2010), 1011.2602.
[2] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, J. Chem. Phys. **21**, 1087 (1953).
[3] W. Hastings, Biometrika **57**, 97 (1970).
[4] R. M. Neal (1993), http://cosmologist.info/Neal93.
[5] Planck Collaboration VI (Planck) (2018), 1807.06209.
[6] J. D. Hunter, Computing in Science & Engineering **9**, 90 (2007).
[7] R. Prescott Adams, I. Murray, and D. J. C. MacKay, arXiv e-prints (2009), 0912.4896.
[8] R. P. Adams, I. Murray, and D. J. C. MacKay, in *Advances in Neural Information Processing Systems 21*, edited by D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou (2009), pp. 9–16, https://papers.nips.cc/paper/3410-the-gaussian-process-density-sampler.pdf.
[9] C. Donner and M. Opper, arXiv e-prints (2018), 1805.11494.
[10] J. Skilling, in *American Institute of Physics Conference Series*, edited by R. Fischer, R. Preuss, and U. V. Toussaint (2004), pp. 395–405, URL http://www.inference.phy.cam.ac.uk/bayesys/.
[11] W. J. Handley, M. P. Hobson, and A. N. Lasenby, MNRAS **453**, 4384 (2015), 1506.00171, URL https://arxiv.org/abs/1506.00171.
[12] F. Feroz and M. P. Hobson, Mon. Not. Roy. Astron. Soc. **384**, 449 (2008), 0704.3704.
[13] A. Lewis and S. Bridle, Phys. Rev. D **66**, 103511 (2002), astro-ph/0205436.
[14] R. M. Neal (2005), math.ST/0502099.
[15] A. Lewis, Phys. Rev. D **87**, 103529 (2013), 1304.4473.
[16] M. Wand and M. Jones, *Kernel Smoothing* (Chapman and Hall, 1994), ISBN 0412552701, http://cosmologist.info/ISBN/0412552701.
[17] S. J. Sheather, Statist. Sci. **19**, 588 (2004), URL http://dx.doi.org/10.1214/088342304000000297.
[18] B. Hansen (2009), httt://www.ssc.wisc.edu/~bhansen/718/NonParametrics1.pdf.
[19] A. Z. Zambom and R. Dias, ArXiv e-prints (2012), 1212.2812.
[20] A. Poluektov, JINST **10**, 02011 (2015), 1411.5528.
[21] M. Jones, Stat. & Comput. **3**, 135 (1993), http://link.springer.com/content/pdf/10.1007/BF00147776.pdf.
[22] M. Jones and D. F. Signorini, J. Amer. Stat. Assoc. **92**, 1063 (1997), http://www.jstor.org/stable/2965571.
[23] M. Jones and P. J. Foster, Stat. Sinica **6**, 1005 (1996), http://www3.stat.sinica.edu.tw/statistica/oldpdf/A6n414.pdf.
[24] M. Jones, O. Linton, and J. Nielsen, Biometrika **82**, 327 (1995), http://www.jstor.org/stable/2337411.
[25] N. W. Hengartnera and E. Matzner-Lober, ESAIM: Prob. & Stat. **13**, 1 (2009), http://dx.doi.org/10.1051/ps:2007055.
[26] E. Choi and P. Hall, Biometrika **86**, 941 (1999), URL http://biomet.oxfordjournals.org/content/86/4/941.abstract.
[27] P. Hall and M. C. Minnotte, J. Roy. Stat. Soc. Series B (Stat. Meth.) **64**, 141 (2002), URL http://www.jstor.org/stable/3088854.
[28] P. Hall, S. N. Lahiri, and Y. K. Truong, Ann. Statist. **23**, 2241 (1995), http://projecteuclid.org/euclid.aos/1034713655.
[29] M. Sköld and G. Roberts, Scand. J. Stat. **30**, 699 (2003), http://www.jstor.org/stable/4616797.
[30] P. Janssen, J. S. Marron, N. Veraverbeke, and W. Sarle, J. Nonparam. Stat. **5**, 359 (1995), http://dx.doi.org/10.1080/10485259508832654.
[31] M. Jones, J. Marron, and S. J. Sheather, J. Amer. Stat. Assoc. **91**, 401 (1996), http://www.stat.washington.edu/courses/stat527/s14/readings/Jones_etal_JASA_1996.pdf.
[32] O. M. Eidous, M. A. A. S. Marie, and M. H. Ebrahem, J. Mod. Appl. Stat. Meth. **9**, 26 (2010), http://digitalcommons.wayne.edu/jmasm/vol9/iss1/26/.
[33] N.-B. Heidenreich, A. Schindler, and S. Sperlich, AStA **97**, 403 (2013), http://doi.org/10.1007/s10182-013-0216-y.
[34] M. D. Marzio and C. C. Taylor, J. Nonparam. Stat. **21**, 229 (2009), http://eprints.whiterose.ac.uk/42950/.
[35] M. P. Wand and M. C. Jones, J. Amer. Stat. Assoc. **88**, 520 (1993), http://www.jstor.org/stable/2290332.
[36] M. Wand and M. Jones, Comput. Stat. **9**, 97 (1994), URL ftp://math-libshare.math.ntu.edu.tw/131212-7377034.pdf.
[37] T. Duong and M. Hazelton, J. Nonparam. Stat. **15**, 17 (2003), URL http://www.mvstat.net/tduong/research/publications/duong-hazelton-2003-jns.pdf.