gk: An R Package for the g-and-k and generalised g-and-h Distributions

by Dennis Prangle

Abstract The g-and-k and (generalised) g-and-h distributions are flexible univariate distributions which can model highly skewed or heavy tailed data through only four parameters: location and scale, and two shape parameters influencing the skewness and kurtosis. These distributions have the unusual property that they are defined through their quantile function (inverse cumulative distribution function) and their density is unavailable in closed form, which makes parameter inference complicated. This paper presents the **gk** R package to work with these distributions. It provides the usual distribution functions and several algorithms for inference of independent identically distributed data, including the finite difference stochastic approximation method, which has not been used before for this problem.

arXiv:1706.06889v1 [stat.CO] 21 Jun 2017

Introduction

Statisticians have long sought for a simple extension to the normal distribution which can model data subject to skew, heavy tails or both. One approach is to transform a standard normal random variable $Z \sim N(0,1)$ to

$$X = A + BG(Z)H(Z), \tag{1}$$

where A and B are location and scale parameters, $G(\cdot)$ introduces asymmetry, and $H(\cdot)$ elongates the tails of the distribution while having little effect near the mode. This paper considers two such distributions, the g-and-k and generalised g-and-h distributions. These distributions can model many types of behaviour through just a small number of parameters.

Defining random variables as transformations of Z is equivalent to specifying the distribution's quantile function (defined in the next section), and distributions of this type are known as *quantile distributions*. Work on quantile distributions goes back at least to Hastings et al. (1947). See Gilchrist (2000) for a book length treatment of their history and use in statistics. Tukey (1977) proposed the form (1) and a distribution using it: the original g-and-h distribution. Haynes et al. (1997) were the first to use the two distributions considered in this paper: the g-and-k distribution and a generalised form of the g-and-h distribution. For brevity henceforth "g-and-h distribution" will refer to their generalised form. See Peters et al. (2016) for a thorough review of these and other distributions based on (1).

Applications of the g-and-k and g-and-h distributions have included environmental data (Rayner and MacGillivray, 2002), financial returns (Drovandi and Pettitt, 2011) and insurance risk (Peters et al., 2016). There has also been considerable methodological work on inference for these distributions (e.g. Rayner and MacGillivray, 2002; Haynes and Mengersen, 2005; Allingham et al., 2009; Drovandi and Pettitt, 2011; Fearnhead and Prangle, 2012). This is because it is not possible to express the densities of quantile distributions in closed form beyond some special cases, which makes it difficult to apply standard likelihood-based inference methods.

This paper presents the **gk** R package to work with the g-and-k and g-and-h distributions. The remaining sections covering the following:

- A mathematical definition of the distributions.
- A description of the package's functions to perform standard distributional tasks and how they are implemented.
- An exploration of the range of valid parameters for these distributions, as this has a complicated form. We propose a novel rule giving "safe" parameter values for the g-and-k distribution.
- A desctiption of several methods for parameter inference and corresponding functions supplied by the package.
- An illustrative analysis of a real dataset.
- A summary.

Definitions

The cumulative distribution function (cdf) of a univariate random variable X, $F_X : \mathbb{R} \to [0,1]$, is defined as $\Pr(X \leq x)$. (Later we will often drop subscripts where they are clear from the context.) The cdf suffices to completely specify the probability distribution of X. It is often the case that the cdf is not available in closed form but is implicitly defined through its derivative, the probability density function (pdf). An example of this is the normal distribution.

For quantile distributions, the cdf is implicity defined through its inverse, the *quantile function* $F_X^{-1}(u)$ where $F_X^{-1} : [0,1] \to \mathbb{R}$. The g-and-k and g-and-h distributions use a quantile function of the form $F^{-1}(u;\theta) = Q(z(u);\theta)$ where $z(\cdot)$ is the $N(0,1)$ quantile function and θ is a vector of parameters. The Q functions are:

$$Q_{gk}(z; A, B, g, k, c) = A + B(1 + c \tanh[gz/2])z(1 + z^2)^k \tag{2}$$

$$Q_{gh}(z; A, B, g, h, c) = A + B(1 + c \tanh[gz/2])z \exp(hz^2/2). \tag{3}$$

It is possible to sample from the distributions using the *inversion method*, that is, by simulating $U \sim U(0,1)$ and substituting it into the quantile function. Equivalently one can sample $Z \sim N(0,1)$ and substitute it into Q_{gk} or Q_{gh} i.e. the process described in the introduction based on Equation (1). In terms of (1), $G(z) = 1 + c \tanh(gz/2)$ produces asymmetry and $H(z) = z(1 + z^2)^k$ or $z \exp(hz^2/2)$ elongates tails.

Each distribution has four main parameters: A (location), B (scale), g (shape parameter mainly affecting skewness), and k or h (shape parameter mainly affecting kurtosis). The remaining parameter c is discussed below. When both shape parameters are zero the distribution is simply $N(0,1)$. An illustration of the flexible shapes that the g-and-k density can take is given in Figure 1. The g-and-h can produce similar shapes, with the following exception. The g-and-k distribution allows negative values of k which can produce lighter tails than a normal distribution, but also bimodal distributions of potentially limited usefulness.

Well-defined continuous distributions result from parameter values producing strictly increasing quantile functions. Determining when this is true is complicated so discussion is postponed to a later section. For now note that it is standard to take $B > 0$ and fix $c = 0.8$ (which will be assumed throughout unless mentioned otherwise), and in this case $k \geq 0$ or $h \geq 0$ guarantees a valid distribution.

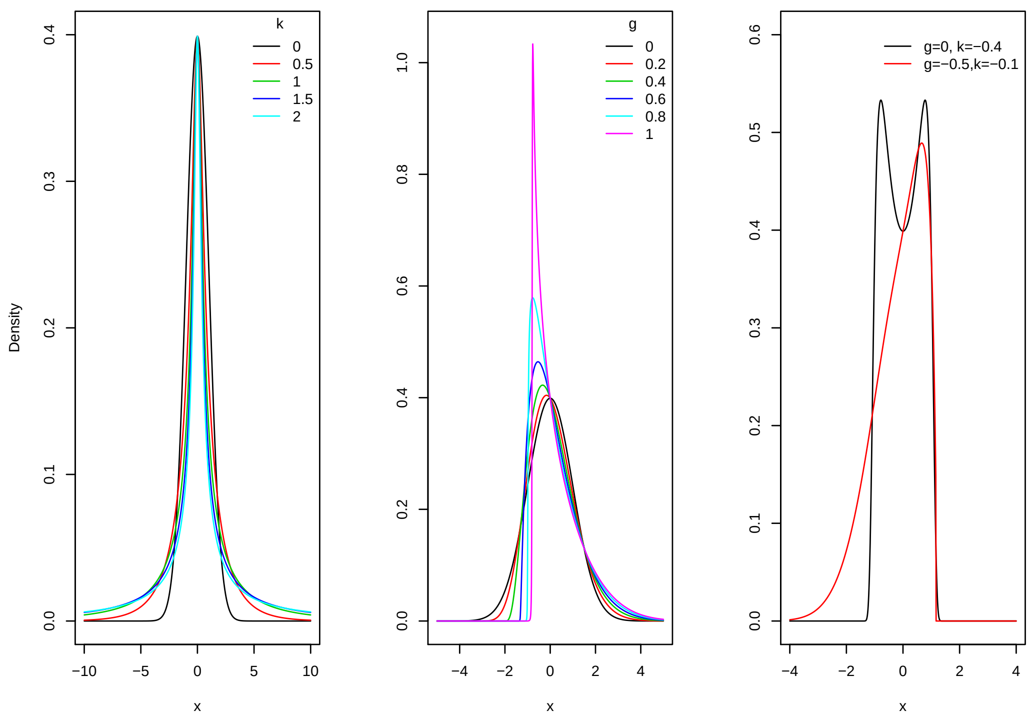

Figure 1: Example g-and-k densities. The first panel fixes $g = 0$ and varies k, mainly altering kurtosis. The second fixes $k = 0$ and varies g, mainly altering skewness. The third shows two examples with $k < 0$.

Distribution functions

The **gk** package provides the standard suite of R functions for the g-and-k and g-and-h distributions i.e. random sampling and calculation of the pdf, cdf and quantile functions. This section describes how these functions are implemented. It is assumed that parameters have been chosen such that the quantile function is strictly increasing. No warning is given when this is not the case as checking validity is time consuming (see next section).

Quantile function The `qgk` and `qgh` functions calculate the quantile function $F^{-1}(u)$. Their implementation is straightforward. First $z(u)$ is calculated using `qnorm`, then this is passed to an internal function, `z2gk` or `z2gh`, which computes Q_{gk} or Q_{gh}.

Random sampling The `rgk` and `rgh` functions perform random sampling. This is done by the method described earlier of sampling $N(0,1)$ draws and substituting them into Q_{gk} or Q_{gh}, via the function `z2gk` or `z2gh`.

Cumulative distribution function The `pgk` and `pgh` functions calculate the cdf $F(x)$ given input x. They numerically solve $Q(z) - x = 0$, which is guaranteed to have a unique root for z. The required final output is then $u = \Phi^{-1}(z)$ where Φ is the $N(0,1)$ cdf. An alternative approach would be to directly solve $Q(z(u)) - x = 0$ for u. However we found this was less numerically stable for u close to 0 or 1.

Our code finds the root for z using R's `uniroot` command and `z2gk` or `z2gh` for $Q(z)$ evaluations. The need to run a root finding algorithm means this function is slow relative to cdf calculations of standard distributions - see Table 1.

The functions include an argument `zscale`. Setting this to `TRUE` outputs the z value which is found rather than u. This is used in the density functions below, and more generally is also useful to retain numerical precision when z has large magnitude.

Probability density function The `dgk` and `dgh` functions calculate the pdf $f(x)$, or the log pdf if the argument `log=TRUE` is supplied. The method is based on the standard probability result that if A has density $f_A(a)$ and $t(a)$ is a differentiable 1-1 transformation then the density of $B = t(A)$ is

$$f_B(b) = f_A(a)/t'(a) \qquad \text{where } a = t^{-1}(b),$$

and t' denotes the first derivative of t.

For quantile distributions we have $Z \sim N(0,1)$ and $X = Q(Z)$ for some Q function. So the pdf of X is

$$f(x) = \phi(z)/Q'(z) \qquad \text{where } z = Q^{-1}(x),$$

where $\phi(z)$ is the $N(0,1)$ pdf.

Our code to calculate the pdf first finds $z = Q^{-1}(x)$ using `pgk` or `pgh` with `zscale=TRUE`. Then the pdf or its log is calculated using formulae (4) and (5) (see Appendix A) for $Q'(u)$. The reliance on performing root finding within `pgk` and `pgh` means that `dgk` and `dgh` are slow relative to pdf calculations for standard distributions - see Table 1.

Note that an alternative representation of $f(x)$ is $1/q'(u)$ where $q(u)$ represents $F^{-1}(u)$ and $u = F(x)$. Density calculations based on this approach are described in Rayner and MacGillivray (2002). However we found that calculating the u values required for this approach was occasionally numerically unstable, as mentioned above.

Cost Table 1 compares the time to execute **gk**'s distributional functions to those for the normal distribution. It illustrates that random sampling and quantile function calculation are reasonably efficient, but calculating the cdf and pdf are expensive.

Range of valid parameters

Recall that a valid continuous distribution requires the quantile function to be strictly increasing. Clearly this property is unaffected by the choice of A and $B > 0$. This section discusses the effects of g, h, k and c.

Several theoretical results on valid parameters can be derived. It's convenient to concentrate on $c \geq 0$. In this case $h < 0$ or $k < -1/2$ is invalid. Taking $k \geq 0$ or $h \geq 0$ produces valid distributions

	Time (microseconds)			Ratio vs normal	
	Normal	g-and-k	g-and-h	g-and-k	g-and-h
Quantile function	175	972	445	5.56	2.55
Random sampling	150	921	436	6.15	2.91
cdf	313	143151	116928	457	374
pdf	369	138381	111279	375	302

Table 1: Mean times to perform various distributional operations, evaluated by the **microbenchmark** pacakge. For example the random sampling row compares `rnorm(N)`, `rgk(N,1,2,3,4)` and `rgh(N,1,2,3,4)` for $N = 100$. We also tried $N = 1$, which gave qualitatively similar results but slightly better relative efficiency of the **gk** functions.

when $0 \leq c < c^* \approx 0.83$. This is the reason for taking $c = 0.8$ as standard: it maintains this property while allowing the skewness factor in (2) and (3) to have a large effect. For justification of all these results, see Appendix B.

When $c = 0.8$, the above results completely characterise the range of valid parameters for the g-and-h distribution. For the g-and-k distribution, there is still some uncertainty for $-0.5 \leq k < 0$, which, as mentioned earlier, corresponds to light tails. For both distributions, the case where $c > c^*$ is less clear: even positive values of k or h do not guarantee validity. Therefore we provide the function `isValid` to test parameter validity numerically.

Validity can be checked by testing whether the minimum derivative of (2) or (3) is positive. Appendix A shows that it is equivalent to test whether the functions (6) or (7) are positive. `isValid` uses numerical optimisation to minimise these and returns whether the minimum value is positive. To reduce the possibility of finding local minima, multiple optimisation starting points can supplied as a vector to the argument `initial_z`. However it is still not guaranteed that the global minimum is found, so there remains a possibility that the function may produce false positives.

The function can be used as follows to illustrate the region of valid g-and-k parameter values for $c = 0.8$. The results are plotted as Figure 2.

```
gk_grid = expand.grid(g = seq(-10,10,0.1), k = seq(-0.6,0.1,0.01))
v = isValid(gk_grid$g, gk_grid$k)
```

We do not test validity automatically within the package's other functions. This is because `isValid` is relatively computationally expensive and not guaranteed to be correct. Therefore particular care should be taken for $k < 0$ or $c > c^*$, as the distribution functions will not provide warnings when invalid parameters are used. A reasonable region of g and k values to use in practice with $c = 0.8$ can be derived from Figure 2. It shows that for $|g| < 7$ some $-0.5 \leq k < 0$ values are invalid. Apart from a narrow strip near $g = 0$, this invalid region's boundary is roughly quadratic, as illustrated by the curve $\tilde{k}(g) = -0.045 - 0.01g^2$ on the figure. Based on this analysis, $k \geq \max(-0.5, \tilde{k}(g))$ seems a reasonable sufficient condition for parameter validity to use in practice.

Inference functions

The package provides three inference methods for data $x_1, x_2, \ldots, x_n$ which are assumed to be independent and identically distributed (IID) draws from a g-and-k or g-and-h distribution with unknown parameters. This section describes these methods. An illustration of their use is provided in the next section. See the **gk** help files for a full description of all the arguments available.

MCMC inference The `mcmc` function implements inference using *Markov chain Monte Carlo* (MCMC). This samples from a Markov chain whose stationary distribution is the Bayesian posterior of interest for the parameters θ. We use a Metropolis-Hastings algorithm, in which a proposed new state of the chain θ' is sampled by adding a $N(0, \Sigma)$ increment to the current state θ_t. A decision to accept or reject θ' is made based on the prior and likelihood values at θ_t and θ' and a random variable (see steps 3-4 of Algorithm 1.)

Tuning Σ can be difficult. Haynes and Mengersen (2005), who first used MCMC for the g-and-k distribution, did this manually. Instead we use the adaptive Metropolis (AM) algorithm of Haario et al. (2001) which tunes Σ automatically during its operation. The resulting θ_ts no longer form a Markov chain, but it has been proved (Saksman and Vihola, 2010) that, under suitable conditions, calculations using them still converge to posterior quantities as the length of the chain increases. The AM algorithm is presented as Algorithm 1. Step 1 states the proposal matrix used in terms of the empirical variance

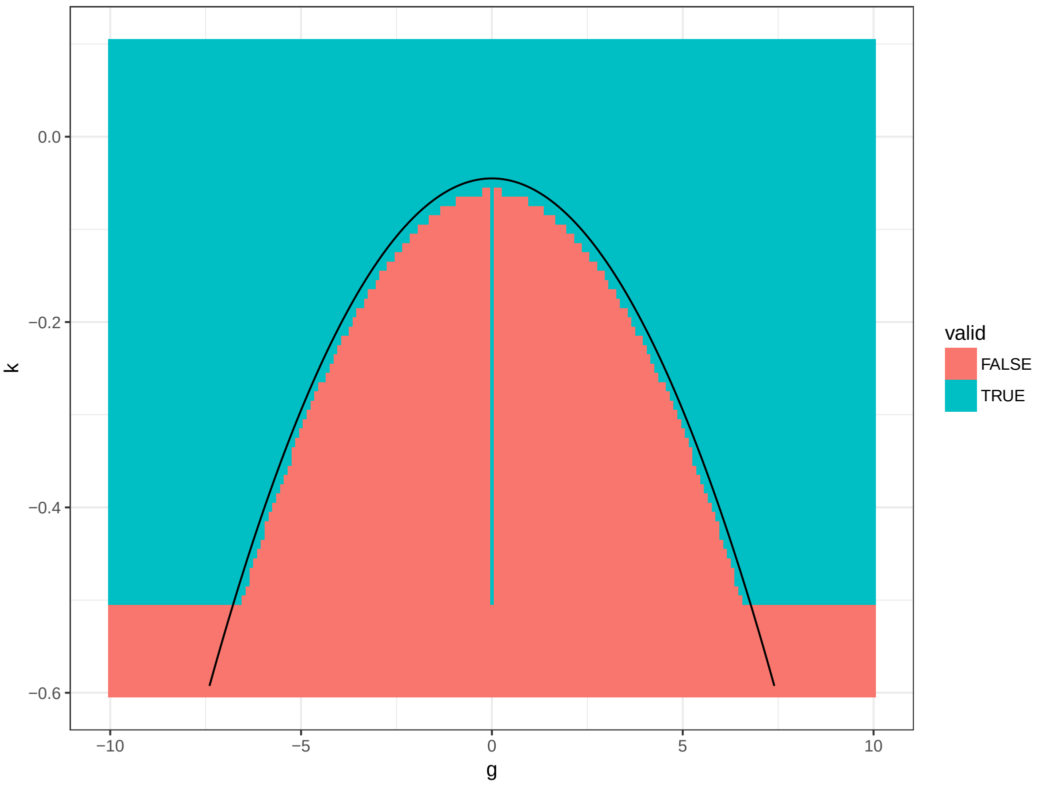

Figure 2: Validity of parameter values for the g-and-k distribution when $c = 0.8$, calculated using `isValid`. Also shown is a quadratic function $\tilde{k}(g)$ near the curved part of the boundary between the regions.

of the past MCMC states. To calculate this empirical variance efficiently, the code updates it each time a new state is observed. As a default we specify tuning choices $\epsilon = 10^{-6}$ and $t_0 = 100$.

Like other Bayesian methods, MCMC requires a prior density for the parameters, $\pi(\theta)$, to be specified. This must be supplied by the user. For computational convenience this should be supplied in the form of a function `get_log_prior` which takes a vector of parameters as input and returns the log prior density. We allow the user to reparameterise θ, using $\log B$ rather than B, via the `logB` argument. This can improve MCMC efficiency when the posterior for B is concentrated on values close to zero.

For IID data the likelihood is $L(\theta) = \prod_{i=1}^{n} f(x_i; \theta)$, the product each observation's pdf. Evaluating this for the g-and-k or g-and-h distributions using the `pgk` or `pgh` command requires n calls to numerical optimisation. Therefore MCMC becomes computationally expensive for even moderately large datasets.

ABC inference The `abc` function implements inference by *approximate Bayesian computation* (ABC). This is a method for approximate Bayesian inference which avoids evaluating the likelihood function. It is especially useful when the likelihood function is unavailable or, as for quantile distributions, is expensive to compute. ABC is based instead on finding parameter values which produce simulated data similar to the observations. The `abc` function implements a simple version of ABC, Algorithm 2. Here a simulation is accepted if it has one of the M smallest distances to the observations. Distance refers to a weighted version of Euclidean distance between vectors of simulated and observed *summary statistics*. Details of the weighting are given in the algorithm's description. (For n large, `abc` avoids high memory requirements by running several batches of Algorithm 2. Each batch uses $N = 10^4$ and returns the M best simulations. The overall best M best simulations are then found and returned. The v_j weights calculated in the first batch are reused in the others.)

Like `mcmc`, `abc` is a Bayesian method and requires a prior distribution for θ to be provided. It is convenient for this to be provided in a different form to the `mcmc` case. A function `rprior` should be supplied which a single numeric input and returns that many samples from the prior distribution as rows of a matrix.

ABC produces samples from an approximation to the Bayesian posterior distribution. The quality of the approximation depends in a complex way on the choice of summary statistics and the tuning

Algorithm 1 The Adaptive Metropolis MCMC algorithm

```
**Input**: observations $x$, prior density $\pi(\theta)$, number of iterations to perform $N$, initial state $\theta_0$, initial variance matrix $\Sigma_0$, pre-tuning period $t_0$, tuning parameter $\epsilon > 0$.

**Loop** over $1 \leq t \leq N$:
    1. If $t \leq t_0$ let $\Sigma_t = \Sigma_0$. Otherwise let $\Sigma_t = \frac{1}{4}(2.4)^2(\hat{\Sigma}_{t-1} + \epsilon I)$, where $\hat{\Sigma}_{t-1}$ is the variance of $\theta_1, \theta_2, \ldots, \theta_{t-1}$.
    2. Sample $\theta' \sim N(\theta_{t-1}, \Sigma_{t-1})$
    3. Sample $u \sim U(0,1)$ and let $r = \frac{\pi(\theta')L(\theta')}{\pi(\theta_{t-1})L(\theta_{t-1})}$.
    4. If $u < r$ let $\theta_t = \theta'$. Otherwise let $\theta_t = \theta_{t-1}$.

**Output**: sample $\theta_0, \theta_1, \ldots, \theta_N$.
```

Algorithm 2 Approximate Bayesian computation (ABC)

```
**Input**: observations $x$, prior distribution $\pi(\theta)$, summary statistic function $s(x)$, number of simulations to perform $N$, number of samples to output $M$.

1. Calculate observed summaries $s_0 = s(x)$.
2. For $1 \leq i \leq N$ sample parameters $\theta_i$ from the prior.
3. For $1 \leq i \leq N$ simulate summary statistics $s(x_i)$ given parameters $\theta_i$. Let $s_{ij}$ denote the $j$th component of $s(x_i)$.
4. For $1 \leq j \leq q$ (where $q = \dim(s_0)$) calculate the empirical variance $v_j$ of the $(s_{ij})_{1 \leq i \leq n}$ values.
5. For $1 \leq i \leq N$ let $d_i = \sum_{j=1}^{q} (s_{ij} - s_{0j})^2 / v_j$.
6. Find the $M$ smallest $d_i$ values and return the corresponding $\theta_i$s.
```

parameters N and M. For more background on ABC see the review paper by Marin et al. (2012) and the handbook of Sisson et al. (2017). Two general R packages for ABC which implement more advanced methods are **abc** (Csilléry et al., 2012) and **easyABC** (Jabot et al., 2013).

Using ABC for the g-and-k and g-and-h distributions was proposed by Allingham et al. (2009) and has been investigated in many subsequent papers. Following Drovandi and Pettitt (2011) we offer three choices of summary statistics which can be selected through the `sumstats` argument: (1) the full order statistics; (2) octiles of the observations, $E_1, E_2, \dots, E_7$; (3) robust estimates of the moments based on the octiles:

$$S_A = E_4, \quad S_B = E_6 - E_2, \quad S_g = (E_6 + E_2 - 2E_4)/S_b, \quad S_k = (E_7 - E_5 + E_3 - E_1)/S_b.$$

Many more sophisticated approaches to choosing ABC summary statistics have been proposed (Blum et al., 2013), but these are a simple starting point.

For summaries (2) or (3) we follow Fearnhead and Prangle (2012) and speed up step 3 of Algorithm 2 by using the fact that the octiles (or close approximations) can be simulated quickly without the need to simulate a full dataset. Suppose $X_1, X_2, \dots, X_N$ are g-and-k or g-and-h variables, and let $X_{(1)} < X_{(2)} \dots < X_{(N)}$ denote the order statistics. We replace E_i with $E_i' = X_{(r(iN/8))}$ where $r(\cdot)$ rounds to the nearest integer. Now we need to simulate 7 order statistics from the g-and-k or g-and-h distribution. To do so we simulate corresponding order statistics of the Uniform$(0,1)$ distribution using the exponential spacings method (Ripley, 1987). This is implemented by the `orderstats` function. The uniform order statistics are then substituted into $F^{-1}(u)$.

FDSA inference The `fdsa` function performs inference using *finite difference stochastic approximation* (FDSA). FDSA, originally due to Kiefer and Wolfowitz (1952), attempts to find θ^* minimising a loss function $\mathcal{L}(\theta)$ by iteratively calculating estimates $\theta_1, \theta_2, \dots$. Each iteration moves the estimate in the opposite direction to an estimate of the loss gradient, based on finite difference calculations.

We use FDSA for maximum likelihood estimation of IID observations. In this setting $\mathcal{L}(\theta)$ can be taken to be the negative log likelihood,

$$\mathcal{L}(\theta) = -\log L(\theta) = -\sum_{i=1}^{n} \log f(x_i; \theta).$$

The gradient of $\mathcal{L}(\theta)$ can be estimated using only a small subset of the data, so FDSA has the potential to scale up to large datasets better than MCMC, while avoiding the approximation error of ABC. Unlike ABC and MCMC, we are not aware of FDSA having previously been used for the g-and-k and g-and-h distributions.

The g-and-k and g-and-h distributions have some parameter constaints (e.g. $B > 0, h \geq 0$). Also we found setting further constaints from preliminary analyses sometimes helps FDSA behave well. Therefore we use a version of FDSA for bounded minimisation from L'Ecuyer and Glynn (1994), presented as Algorithm 3.

Algorithm 3 Finite difference stochastic approximation (FDSA)

```
Input: initial state $\theta_0$, choice of $a_t$ and $c_t$ sequences, function $\hat{\mathcal{L}}(\cdot)$ which calculates
an unbiased estimate of $\mathcal{L}(\cdot)$, number of iterations to perform $N$, vectors of (possibly
infinite) upper and lower parameter bounds $\theta^+, \theta^-$.

Loop over $0 \leq t \leq N-1$.

1. Calculate $\hat{g}_t$ by performing the following steps for $i \leq 1 \leq 4$.
    (a) Let $\Delta_i$ be a 4-dimensional vector whose $i$th component is 1 and others are zero.
    (b) Let $\phi^+ = P(\theta_t + c_t\Delta_i)$ and $\phi^- = P(\theta_t - c_t\Delta_i)$.
        (Here $P(\phi)$ is a projection operator. Its output is $\phi'$ such that $\phi_i'$ is the closest value
        to $\phi_i$ in $[\theta_i^-, \theta_i^+]$. The $i$ subscripts represent $i$th components.)
    (c) Let $\hat{g}_{it} = \frac{1}{|\phi_i^+ - \phi_i^-|}[\hat{\mathcal{L}}(\phi^+) - \hat{\mathcal{L}}(\phi^-)]$.
2. Let $\theta_{t+1} = P(\theta_t - a_t\hat{g}_t)$.

Output: Final estimate $\theta_N$.
```

The unbiased estimate of $\mathcal{L}(\theta)$ required by Algorithm 3, $\hat{\mathcal{L}}(\theta)$, can be taken to be the sum of a random sample of m negative log likelihood terms multiplied by n/m. Hence for a vector y containing a random subsample of m observations (sometimes referred to as a *batch*), $\hat{\mathcal{L}}(\theta)$ can be calculated using `-sum(dgk(y,A,B,g,k,log=TRUE))*n/m` (or similar for the g-and-h distribution). Variance reduction in step 1c of Algorithm 3 is possible by coupling the two estimates (Kushner and Yin, 2003). Hence we use the same random subsample of data for all $\hat{\mathcal{L}}$ calculations in an iteration of step 1.

FDSA convergence requires that the *gain sequences* a_t and c_t must satisfy certain conditions. Following Spall (1998) we take $a_t = a_0(A+t+1)^{-\alpha}$ and $c_t = c_0(t+1)^{-\gamma}$. This leaves several tuning choices, which can be selected by the user, or left at default values which we provide. Following Kleinman et al. (1999) we use default values $\alpha = 1$ and $\gamma = 0.49$. Following Spall (1998) our default for c_0 is an estimate of the standard deviation of $\hat{\mathcal{L}}(\theta_0)$ using some preliminary simulations. We provide defaults $a_0 = 1$ and $A = 100$ but it is recommended to manually tune these to produce rapid convergence. This may require several short pilot runs of the algorithm. The `fdsa` function allows a_0 and c_0 to be vectors, in which case operations in Algorithm 3 are interpreted as elementwise where necessary. This allows the user to tune gain sequences differently for each parameter. As for `mcmc`, we allow the user to reparameterise θ, using $\log B$ rather than B, via the `logB` argument, which can improve FDSA efficiency when the MLE value of B is close to zero.

Under weak assumptions, FDSA converges to a local minimum of $\mathcal{L}(\theta)$ (Kushner and Yin, 2003). In our experience the likelihood for the g-and-k and g-and-h distributions is usually unimodal, so there is little danger of converging to an incorrect mode. Nonetheless it may be a useful check on the results to rerun the algorithm from various starting points or compare with the output of another algorithm.

An alternative to FDSA is *simultaneous perturbation stochastic approximation* (SPSA) (Spall, 1998). Here each iteration makes a finite difference estimate of the derivative of the loss function when moving in a random direction from θ_t. An update moves θ_t a distance (negatively) proportional to this estimated derivative in the selected direction. Each SPSA iteration requires fewer likelihood estimates than FDSA, and it is asymptotically more efficient (Kushner and Yin, 2003). However we found in exploratory work that for our application the SPSA updates were dominated by improving A and B estimates, and the remaining parameters were learned very slowly.

Illustration

We illustrate **gk**'s inference methods on the `Garch` exchange rate dataset from the **Ecdat** package. This consists of 1967 daily US dollar exchange rates against other currencies from 1980 to 1987. We concentrate on the exchange rate with Canadian Dollars. Let x_t denote the exchange rate on day t. The *log return* is defined as $\log(x_{t+1}/x_t)$. Figure 3 is a time series plot of the log returns. Figure 7 shows a histogram and a quantile-quantile plot indicating that the tails are heavier than those of a normal density.

We focus on using the g-and-k distribution to model the log returns under an IID assumption. For models also including time series structure see for example Drovandi and Pettitt (2011). The full code for the analysis below can be run via the `fx` function.

The ABC and MCMC analyses which follow are Bayesian and require specification of a prior. We use a uniform prior for ease of comparison to the maximum likelihood results from FDSA. For MCMC we are able to use an improper uniform prior. For ABC a proper prior is required so we bound the parameters as follows $-1 < A < 1$, $0 < B < 1$, $-5 < g < 5$, $0 < k < 10$. We restrict A and B to magnitude 1 at most, as we believe log returns of this magnitude are highly unlikely. The g and k parameters are given wider support which can capture a broad range of distributional shapes.

ABC We ran ABC as follows:

```
rprior = function(i) {cbind(runif(i,-1,1), runif(i,0,1), runif(i,-5,5), runif(i,0,10))}
abc_out = abc(log_return, N=1E7, rprior=rprior, M=200, sumstats='moment estimates')
```

This simulated 10^7 parameter vectors and accepting the best 200. We used moment estimator summary statistics, described earlier, which can be simulated quickly without the need to simulate an entire dataset. As a result this analysis took only 6 minutes.

The resulting approximate posterior samples are shown in Figure 6. Figure 7 shows density and quantile-quantile plots under the mean parameter values. These reveal a very poor fit to the data. However this short ABC analysis does provide reasonable tuning choices for the other methods.

FDSA We ran FDSA as follows:

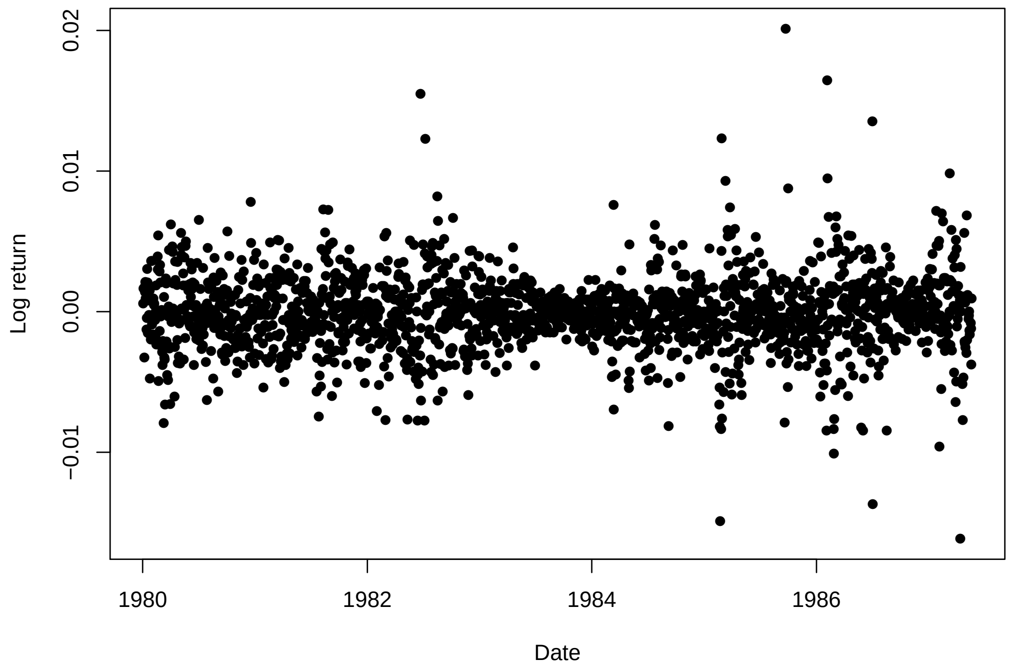

Figure 3: Log returns for US dollar / Canadian Dollar exchange rates.

```
abc_out_tf = abc_out[,1:4]
abc_out_tf[,2] = log(abc_out_tf[,2])
abc_est_tf = colMeans(abc_out_tf)
fdsa_out_pilot = fdsa(log_return, N=1E4, logB=TRUE, theta0=abc_est_tf, batch_size=100,
                      a0=2E-4)
a0 = c(1E-6, 1E-2, 1E-2, 1E-2)
fdsa_out = fdsa(log_return, N=1E4, logB=TRUE, theta0=abc_est_tf, batch_size=100, a0=a0)
```

We found that using the original parameterisation caused high variance in our gradient estimates. This is because the log-likelihood surface becomes extremely steep for B close to 0. Therefore we reparameterised B to $\log B$. The initial FDSA state was set to equal the ABC means. The FDSA steps sizes a_0 were tuned by trial-and-error.

Figure 4 shows a trace plot of the FDSA algorithm output. A pilot run with $a_0 = 2 \times 10^{-4}$ is shown in black. Parameters $\log B, g$ and k do not converge over 10,000 iterations. However they have smooth curves, indicating that there is relatively little noise in their gradient estimates and so larger steps could be taken. In contrast A converges quickly and then oscillates noisily. This indicates that a smaller step size could be used to average out this noise more effectively without endangering convergence. Therefore for the final run we used $a_0 = (10^{-6}, 10^{-2}, 10^{-2}, 10^{-2})$.

The final FDSA analysis took 17 minutes. The final states were $A = 9.1 \times 10^{-5}$, $B = 1.7 \times 10^{-3}$, $g = 2.0 \times 10^{-2}$ and $k = 0.35$. Figure 7 shows density and quantile-quantile plots under these parameter values. These are a much better fit to the data than the ABC results.

Next we use the FDSA results to help tune an MCMC algorithm, which quantifies the uncertainty in the parameter values.

MCMC We ran MCMC as follows:

```
fdsa_est_tf = fdsa_out[1E5,1:4]
Sigma0 = var(fdsa_out[1E5 + (-1000:0),1:4])
log_prior = function(theta) {
    if (theta[4]<0) return(-Inf)
    return(theta[2])
}
mcmcout_tf = mcmc(log_return, N=1E4, logB=TRUE, get_log_prior=log_prior,
```

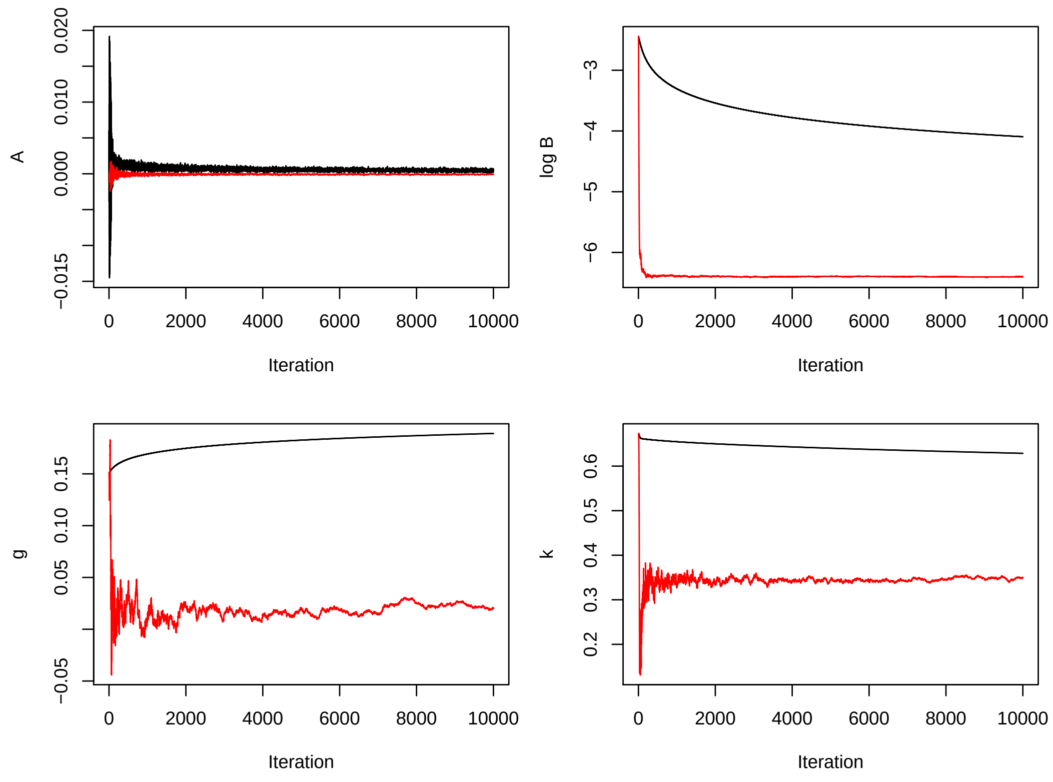

Figure 4: Output from the FDSA algorithm to infer g-and-k parameters for exchange rate log returns. Black shows output from a pilot run with $a_0 = 2 \times 10^{-4}$. Red shows output from the final run with $a_0 = (10^{-6}, 10^{-2}, 10^{-2}, 10^{-2})$.

```
                    theta0=fdsa_est, Sigma0=Sigma0)
```

Again we used a log reparameterisation for B. To achieve an improper uniform prior on the original parameterisation, we used a prior density proportional of $B\mathbb{1}(k > 0)$ on $(A, \log B, g, k)$ (where $\mathbb{1}$ represents an indicator function). Our initial parameter vector was the final FDSA state. We use the variance matrix of the last 1000 FDSA states to select the initial MCMC proposal variance.

Figure 5 shows a trace plot of the MCMC algorithm output. For the first few hundred iterations small proposals are made, at least for $\log B$, g and k, but the proposal variance quickly adapts and the remainder of the output appears to have converged. Exploratory work showed that taking a poor initial state meant MCMC is very slow to converge, because the variance matrix adapts to the transient state of the algorithm. Hence tuning based on FDSA output is very useful.

The MCMC analysis took 39 minutes. Figure 6 parameter histograms and figure 7 shows density and quantile-quantile plots. These are similar to the FDSA fit. Note that the density plot is based on mean parameter values from the MCMC output (after discarding the first half of the output as burn-in and transforming $\log B$ values back to the original parameterisation).

Summary The ABC analysis is quick but produces a poor fit. However it helps tune the FDSA method which finds a good estimate of the MLE in a reasonable time. Further computational effort using MCMC provides a Bayesian fit. Figure 7 shows that the g-and-k distribution fits the data better than a normal distribution, but still does not fit the most extreme observations. Further improvements might be possible by using more flexible distributions, for example allowing different k parameters for the upper and lower tails (Peters et al., 2016).

Discussion

This paper has reviewed the g-and-k and g-and-h distributions, and introduced the **gk** package to work with them. The package includes the usual distributional functions, although the pdf and cdf functions are slow due to relying on numerical root-finding. Another function tests the validity of different parameter combinations, and this was used to produce a novel result on which parameters are valid for the g-and-k distribution (i.e. it is appears to be sufficient that $k \geq \max(-0.5, -0.045 - 0.01g^2)$.)

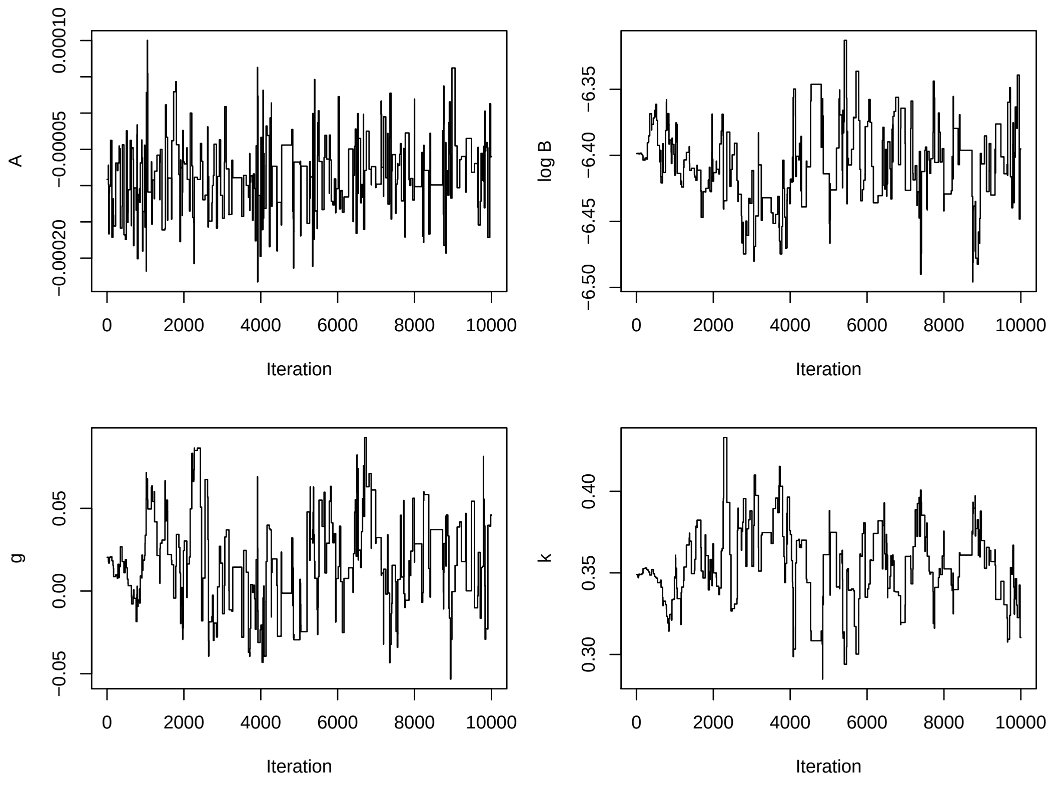

Figure 5: States of an MCMC algorithm to infer g-and-k parameters for exchange rate log returns.

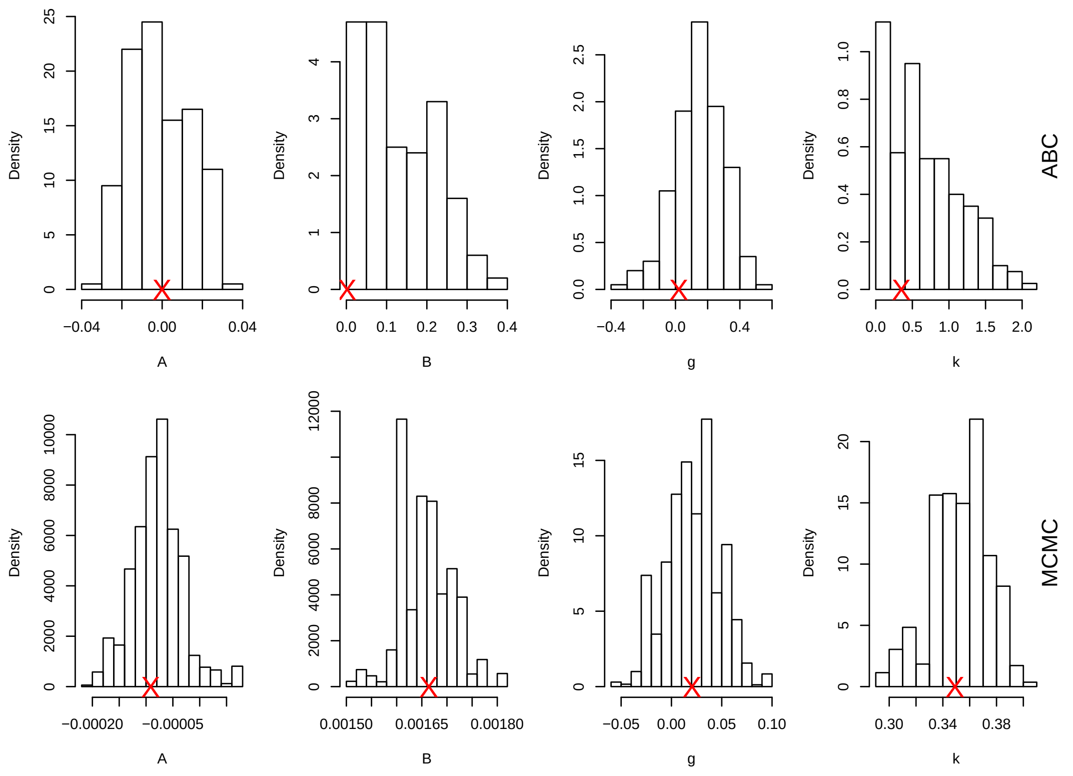

Figure 6: Parameter inference for fitting the g-and-k distribution to exchange rate log returns. The top row shows the ABC posterior sample and the bottom row the MCMC posterior sample, which requires much more concentrated parameter scales. FDSA estimates of the MLEs are shown by crosses on the x-axis.

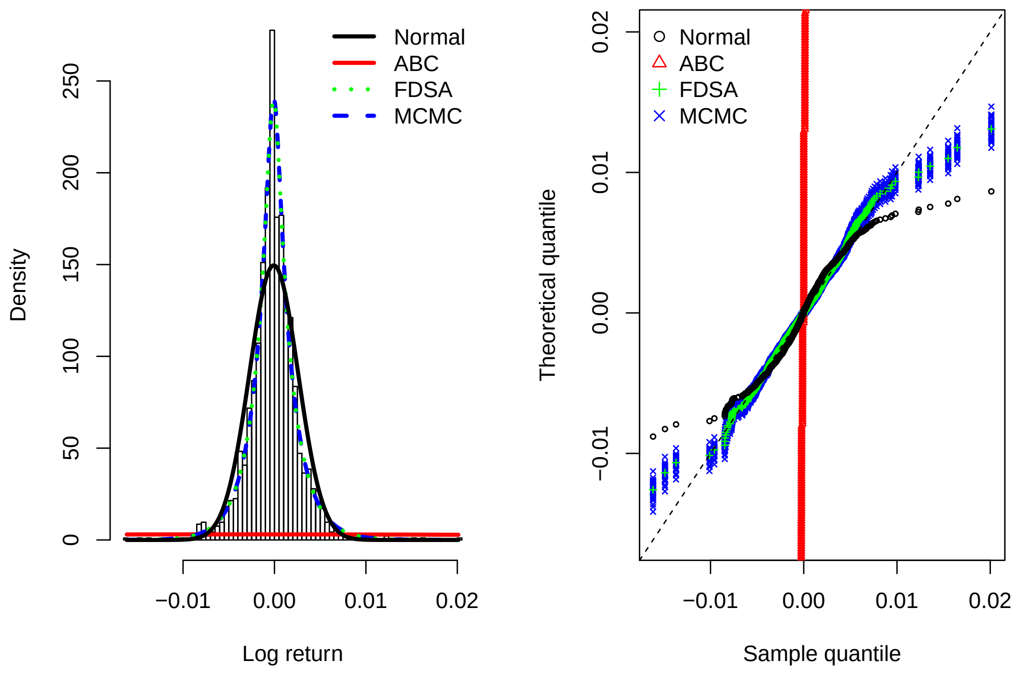

Figure 7: (Left) Histogram of exchange rate log returns, and fitted g-and-k densities. (Right) Quantile-quantile (QQ) plots of fitted g-and-k densities. QQ plots are shown for 30 vectors of parameters sampled from the second half of the MCMC output.

The package also provides several methods for inference of IID data under these distributions, and their use has been illustrated above. The methods include a FPSA algorithm which can find MLEs for large datasets in a reasonable time and has not been applied to this problem before.

Appendix A: Formulae

The derivatives of the Q functions are as follows:

$$Q'_{gk}(z;A,B,g,k,c) = B(1+z^2)^k R_{gk}(z;g,k), \tag{4}$$

$$Q'_{gh}(z;A,B,g,h,c) = B\exp(hz^2/2) R_{gh}(z;g,h), \tag{5}$$

where

$$R_{gk}(z;g,k,c) = [1+c\tanh(gz/2)]\,\frac{1+(2k+1)z^2}{1+z^2} + \frac{cgz}{2\cosh^2(gz/2)}, \tag{6}$$

$$R_{gh}(z;g,h,c) = [1+c\tanh(gz/2)]\,(1+hz^2) + \frac{cgz}{2\cosh^2(gz/2)}. \tag{7}$$

Observe that each Q' function has the same sign and roots as the corresponding R function.

Appendix B: Range of valid parameters - theory

This appendix proves theoretical results quoted earlier about which parameter values produce valid g-and-k and g-and-h distributions.

First note that the defining functions in (2) and (3) both have the property that $Q(z;A,B,-g,k,c) = Q(z;A,B,g,k,-c)$. Therefore any behaviour produced by $c<0$ can be replicated with $c>0$ and a different choice of g. So for simplicity it suffices to concentrate on $c \geq 0$.

For the remainder of this appendix, distributional validity will correspond to a strictly increasing quantile function. This property is generally violated if $c > 1$, as there are two solutions to $Q(z) = A$: $z = 0$ and a solution to $1 + c \tanh(gz/2) = 0$ (The only exception is the special case of $g = 0$.) Also taking $h < 0$ or $k < -1/2$ is invalid, as in either case Q, which is continuous, has a positive gradient at $z = 0$ but limits of zero.

Finally it is shown that non-negative values of k or h produce valid distributions provided that $0 \leq c < c^* \approx 0.83$ (Rayner and MacGillivray, 2002). From Appendix A it suffices to derive the values of c such that $R(z)$ – representing either $R_{gk}(z; g, k, c)$ or $R_{gh}(z; g, h, c)$ – is guaranteed to be positive for $k \geq 0$ or $h \geq 0$. Note that $R(z)$ is a continuous function of z, and $R(0) > 0$. So a sufficient condition for validity is that no solution to $R(z) = 0$ exists. Rearranging $R(z) = 0$ using (6) and (7) gives

$$1/c = uv \operatorname{sech}^2 u + \tanh u, \tag{8}$$

$$\text{where} \quad u = -gz/2,$$

$$\text{and} \quad v = \begin{cases} \frac{1+z^2}{1+(2k+1)z^2} & (g\text{-and-}k) \\ \frac{1}{1+hz^2} & (g\text{-and-}h) \end{cases}$$

For $k \geq 0$ or $h \geq 0$, v can only take values in $(0, 1]$ with 1 attained by $z = 0$. Hence (8) gives $c > 0$ if and only if $u > 0$, and we concentrate on this case from now on. We wish to find the minimum positive solution for c. Since $1/c$ is increasing in v it suffices to concentrate on its largest value, $v = 1$. The problem reduces to minimising $(u \operatorname{sech} u + \tanh u)^{-1}$ for $u > 0$. Numerically this gives $c^* \approx 0.83$, as shown in Figure 8.

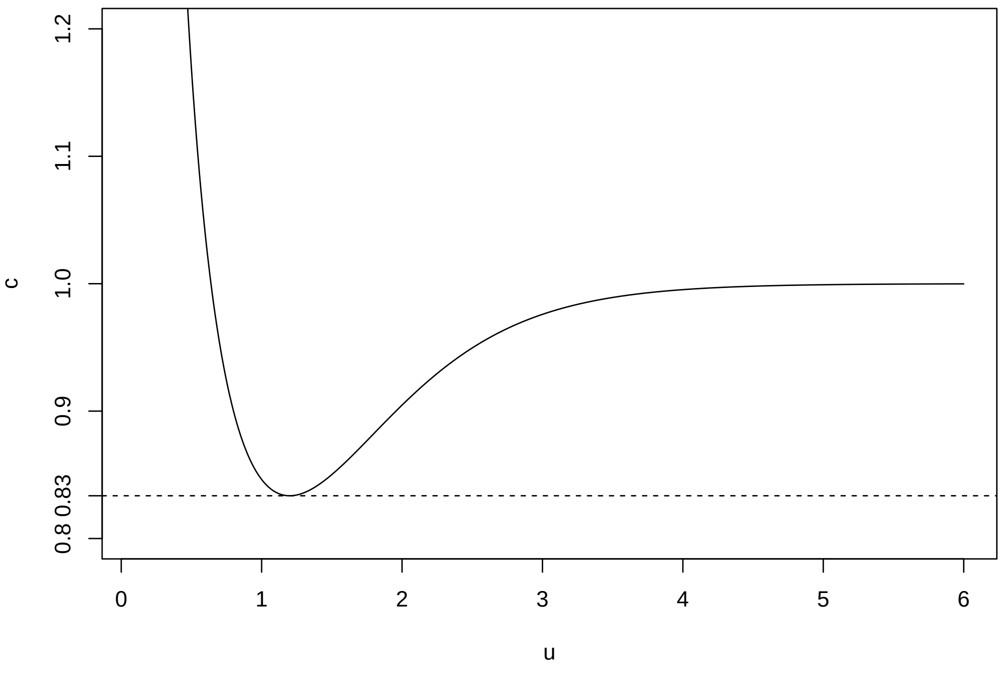

Figure 8: Solutions to (8) for $v = 1$ and $u > 0$.

Acknowledgements

Thanks to Kieran Peel who wrote a helpful undergraduate dissertation on this topic.

Bibliography

D. Allingham, R. A. R. King, and K. L. Mengersen. Bayesian estimation of quantile distributions. *Statistics and Computing*, 19(2):189–201, 2009. [p]

M. G. B. Blum, M. A. Nunes, D. Prangle, and S. A. Sisson. A comparative review of dimension reduction methods in approximate bayesian computation. *Statistical Science*, 28(2):189–208, 2013. [p]

K. Csilléry, O. François, and M. G. B. Blum. abc: an R package for approximate Bayesian computation (ABC). *Methods in ecology and evolution*, 3(3):475–479, 2012. [p]

C. C. Drovandi and A. N. Pettitt. Likelihood-free Bayesian estimation of multivariate quantile distributions. *Computational Statistics & Data Analysis*, 55(9):2541–2556, 2011. [p]

P. Fearnhead and D. Prangle. Constructing summary statistics for approximate Bayesian computation: Semi-automatic ABC. *Journal of the Royal Statistical Society, Series B*, 74:419–474, 2012. [p]

W. Gilchrist. *Statistical modelling with quantile functions*. CRC Press, 2000. [p]

H. Haario, E. Saksman, and J. Tamminen. An adaptive Metropolis algorithm. *Bernoulli*, pages 223–242, 2001. [p]

C. Hastings, Jr., F. Mosteller, J. W. Tukey, and C. P. Winsor. Low moments for small samples: a comparative study of order statistics. *The Annals of Mathematical Statistics*, pages 413–426, 1947. [p]

M. Haynes and K. Mengersen. Bayesian estimation of g-and-k distributions using MCMC. *Computational Statistics*, 20(1):7–30, 2005. [p]

M. A. Haynes, H. L. MacGillivray, and K. L. Mengersen. Robustness of ranking and selection rules using generalised *g*-and-*k* distributions. *Journal of Statistical Planning and Inference*, 65(1):45–66, 1997. [p]

F. Jabot, T. Faure, and N. Dumoulin. EasyABC: performing efficient approximate Bayesian computation sampling schemes using R. *Methods in Ecology and Evolution*, 4(7):684–687, 2013. [p]

J. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 23(3):462–466, 1952. [p]

N. L. Kleinman, J. C. Spall, and D. Q. Naiman. Simulation-based optimization with stochastic approximation using common random numbers. *Management Science*, 45(11):1570–1578, 1999. [p]

H. J. Kushner and G. G. Yin. *Stochastic approximation and recursive algorithms and applications*. Springer, 2003. [p]

P. L'Ecuyer and P. W. Glynn. Stochastic optimization by simulation: Convergence proofs for the GI/G/1 queue in steady-state. *Management Science*, 40(11):1562–1578, 1994. [p]

J.-M. Marin, P. Pudlo, C. P. Robert, and R. J. Ryder. Approximate Bayesian computational methods. *Statistics and Computing*, 22(6):1167–1180, 2012. [p]

G. W. Peters, W. Y. Chen, and R. H. Gerlach. Estimating quantile families of loss distributions for non-life insurance modelling via L-moments. *Risks*, 4(2):14, 2016. [p]

G. D. Rayner and H. L. MacGillivray. Numerical maximum likelihood estimation for the g-and-k and generalized g-and-h distributions. *Statistics and Computing*, 12(1):57–75, 2002. [p]

B. Ripley. *Stochastic Simulation*. Wiley, 1987. [p]

E. Saksman and M. Vihola. On the ergodicity of the adaptive Metropolis algorithm on unbounded domains. *The Annals of Applied Probability*, 20(6):2178–2203, 2010. [p]

S. A. Sisson, Y. Fan, and M. Beaumont, editors. *Handbook of Approximate Bayesian Computation*. Chapman & Hall/CRC, 2017. [p]

J. C. Spall. Implementation of the simultaneous perturbation algorithm for stochastic optimization. *IEEE Transactions on aerospace and electronic systems*, 34(3):817–823, 1998. [p]

J. W. Tukey. Modern techniques in data analysis. In *Proceedings of the NSF-Sponsored Regional Research Conference*. Southern Massachusetts University, 1977. [p]

Dennis Prangle
Department of Mathematics and Statistics
Newcastle University
NE1 7RU
UK
dennis.prangle@newcastle.ac.uk