Graphical tools for the examination of high-dimensional functions obtained as the result of Bayesian analysis

W K Kaye

A thesis submitted in partial fulfilment of the
requirements of Nottingham Trent University
for the degree of Doctor of Philosophy

August 2009

Contents

List of Tables

List of Figures

Acknowledgements

My thanks go to Serif UK. and their MD and founder Gwyn Jones who sponsored a large part of this work.

To my family Winifred Kaye, Pauline Smith, Vanessa Smith, Steff Smith, Samantha Downes, David Downes and Rebecca Rider.

To Judy, Ted and Richard Hunt for their help and encouragement.

To Thom Baguley (not least for the loan of a MAC G3 laptop), Paul Evans, Wayne Cranton, John Marriott and the other academic colleagues and friends who have put up with it and even encouraged me in the last 17 years.

Finally to John C. Naylor for his support and understanding.

Abstract

Bayesian statistics has a tendency to produce objects that are of many more than three dimensions, typically of the same dimensionality as the parameter set of the problem. This thesis takes the idea of visual, exploratory data analysis and attempts to apply it to those objects. In order to do this it examines several areas, Monte Carlo Markov Chains (MCMC), display graphics and methods – especially Projection Pursuit methods – and Kernel Density Estimation (KDE).

During the course of this work acceptable prior technology was found for MCMC and, once the decision for it had been made, Projection Pursuit. However, the current state of KDE gave rise to several objections. Not least among these came from the Bayesian background of the researcher, KDE had not been put in a suitable Bayesian framework and so clashed with the other technology. In addition it was felt that KDE needed too much user input and that is was somewhat ad hoc. This led to reformulating KDE in a Bayesian framework which had the added advantage of removing the need for a user to provide a bandwidth for each application. Chapter 6 of this thesis considers Bayesian theory and how it can be applied to KDE to produce a form more usable and satisfying in terms of Bayesian mathematics.

This is shown to provide a powerful and flexible statistical tool without the need for the ad hoc choices often associated with these methods. This formulation of the KDE as a Bayesian problem is believed to be unique.

As part of this work, software was produced in R to provide a usable visualisation of BKDE. A large number of examples is provided to demonstrate how this software can allow easy visualisation of a variety of types of dataset both with and without Kernel Density Estimation.

Chapter 1

Introduction

Bayesian statistics has a tendency to produce objects that are of many more than three dimensions. This thesis takes the idea of visual, exploratory, data analysis and attempts to apply it to those objects. In order to do this it examines several areas, Monte Carlo Markov Chains (MCMC), display graphics and methods, especially Projection Pursuit methods, and Kernel Density Estimation (KDE).

During the course of this work acceptable prior technology was found for MCMC and, once the decision for it had been made, Projection Pursuit. However, the current state of KDE gave rise to several objections, not least among these came from the Bayesian background of the researcher. KDE had not been put in a suitable Bayesian framework and so clashed with the other technology. In addition it was felt that KDE needed too much user input and that is was somewhat ad hoc. This led to a new formulation of KDE in a Bayesian framework which had the added advantage of removing the need for a user to provide a bandwidth for each application. Chapter 6 of this thesis considers Bayesian theory and how it can be applied to KDE to

produce a form more usable and satisfying in terms of Bayesian mathematics, ending with a Bayesian formulation that, at least, has the surety of selecting the best possible model for the data from the KDE family of models.

Bayesian Statistics is a branch of statistics that relies on the statement of Bayes' Theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{1.1}$$

originally published in Bayes (1763). Bayesian analysis tends to result in either a symbolically described posterior density (if the analysis is analytical) or as a sample from such a density (if the analysis is implemented via some form of simulation method such as MCMC).

Generally in statistics the first port of call is to some form of exploratory data analysis. Histograms, box plots, scatter plots, means and standard deviations are all tools that allow the statistician to perform an initial assessment of a problem. With the high dimension of the results produced, Bayesian analysis has no such tools without first summarising the results in some way. This thesis attempts to provide a methodology for the graphical examination of such a result.

Bayes' Theorem is often stated to proportionality as:

$$P(A|B) \propto P(B|A)P(A) \tag{1.2}$$

and the application of Bayes' Theorem to a problem depends on finding the constant of proportionality, usually through numerical integration or by using a prior that is amenable to mathematical analysis.

In the above:

$P(A)$	the prior probability of A. "Prior" in the sense that it does not take into account any information about B and is generally held to represent belief about the existing value of A.
$P(A\|B)$	the conditional probability of A, given B, also called the posterior probability because it is derived, using Bayes' Theorem, having observed B.
$P(B\|A)$	the conditional probability of B given A, the likelihood.
$P(B)$	the probability of B, acts as a normalising constant.

If the problem is the analysis of some data then

A	is the parameter set of the problem (i.e., A describes a density that is the current belief about whatever is being measured).
B	is the measured data.

Intuitively, Bayes' Theorem, in this form, describes the way in which beliefs about 'A' are updated by having observed 'B'.

If the problem is not amenable to a formulation that allows an analytic solution it is possible to use MCMC methods to derive an arbitrarily large sample from that density or, equivalently (and sometimes more usefully), from the Bayesian predictive density[1] derived from the solution.

Interest in carrying out Bayesian analysis completely by means of Monte

[1]Assume a probabilistic model $p(y|w)$, parametrised by w, with the prior distribution $p(w)$. Given an observed a set of data D consisting of N i.i.d.[2] observations $\{\mathbf{y}_n\}_{n=1}^N$. A standard approach is to approximate the posterior $p(w|D)$, and use this to make future predictions.

[2]In probability theory and statistics, a sequence or other collection of random variables is independent and identically distributed (i.i.d.) if each has the same probability distribution as the others and all are mutually independent.

Carlo Markov Chain (MCMC) simulation has grown as the practical means of carrying out such an analysis has become commonly available, see for example Smith and Roberts (1993); Gelfand and Smith (1990a); Dellaportas and Smith (1993); Chib and Greenberg (1994); Hastings (1970) and for a practical example Geman and Geman (1984). The application of these methods leads to the availability of samples, again arbitrarily large, from posterior densities.

However the sample is obtained, the interest here is in the application of, largely existing, technology to the problem of presenting a useful view of the posterior density.

The approach is to derive a representative sample from the density, either as an MCMC simulation from the symbolic density or as a random selection from the simulation output, and to use Projection Pursuit methods (Jones and Sibson, 1987) to obtain a series of views from that sample. In order to better view the projection a Kernel Density Estimation technique is used giving a more readily assessed summary of the information.

In summary the goal of this thesis is to examine possible methodologies for the examination of high dimensional functions, surveying existing theory, adding new work where needed and concluding with a working system. The following sections detail the software used in the examples given in the thesis and outline the content of each chapter.

1.1 Software

1.1.1 R and S-Plus

Throughout the thesis R refers to the R Project for Statistical Computing that provides " a free software environment for statistical computing and graphics." (CRAN, 2009)

R compiles and runs on a wide variety of UNIX platforms, Windows and MacOS and can be downloaded from http://www.r-project.org/. R in the thesis refers to version 2.8.1.

The software for this project was originally written on various versions of S-Plus 3.x installed on the computing network situated in the Department of Mathematics and Statistics of Nottingham Trent University between 1993 and 1998. R provides a public domain language that approximates that of S-Plus.

S is a high level language and S-Plus is a value-added version of S sold by Insightful Corporation. S-Plus provides advanced statistical analysis capabilities based on the S language. There are effectively three current implementations of S, the old S engine (S version before version 4), the new S engine (S version 4 and above), and R. R is, of course, distributed freely through the CRAN network.

1.1.2 Bayes4

The Bayes4 system is "primarily intended for the numerical implementation of integrals, for example

$$p(x) = \int_{\Omega} l(x;\theta)p(\theta)d\theta \tag{1.3}$$

$$E(g(\theta)|x) = \int_{\Omega} g(\theta)p(\theta|x)d\theta \tag{1.4}$$

or

$$p(\theta_1|x) = \int_{\Omega} p(\theta|x)d\theta_c \tag{1.5}$$

the evaluation of which is implicit in the practical implementation of the Bayesian paradigm." (Naylor and Shaw, 1983)

The original Bayes4 was implemented in Fortran 77 by John C Naylor and was derived from work in his PhD thesis (Naylor, 1982).

The system, which consists of a library of functions and classes, has been translated from the original Fortran to C++ and its use involves writing problem specific C++ code and building an executable image that incorporates the Bayes4 library[3].

1.1.3 WinBugs

"The BUGS (Bayesian inference Using Gibbs Sampling) project is concerned with flexible software for the Bayesian analysis of complex statistical models using Markov chain Monte Carlo (MCMC) methods. The project began in 1989 in the MRC Biostatistics Unit and led initially to the 'Classic' BUGS program, and then onto the WinBUGS software developed jointly with the

[3]I am indebted to Dr. J M Marriott for the most recent version of the Bayes4 code.

Imperial College School of Medicine at St Mary's, London. Development now also includes the OpenBUGS project in the University of Helsinki, Finland. There are now a number of versions of BUGS, which can be confusing." (WinBugs, 2009)

The version used to generate the examples in Chapter 8 was WinBUGS 1.4.3. More information about the background to the system can be found in Lunn et al. (2000).

1.1.4 C++

The C++ compiler used for the compilation of R was the gcc (Ubuntu 4.3.3-5ubuntu4) compiler 4.3.3 Copyright (C) 2008 Free Software Foundation, Inc.

That used for the compilation of the C++ version of the Bayes4 libraries and the BKDE code that uses them was that provided with the Yellow Dog operating system for MAC computers, gcc 4.1.2. Copyright (C) 2006 Free Software Foundation, Inc.

The C version of the BKDE code was compiled on a SUN system running SunOS with the gcc compiler current in 1993.

1.2 Data

Many of the examples in this thesis use one of the following data sets, chosen because they have interesting features. A brief description follows:

The clinical trial data. A 24 dimensional posterior density, being cell probabilities of a $2 \times 2 \times 6$ contingency table from a simulated clinical trial,

where the parameters are some function of the cell probabilities. These data are from a clinical trial simulation system created by Dr J C Naylor for Nottingham Clinical Trials Ltd..

The Archaeological problem. A sample of the posterior density given by a MCMC approach to the archaeological problem specified in Naylor and Smith (1988) and discussed in 6.2. The problem is a difficult one to describe and is an attempt to estimate the endpoints of a series of phases of pottery production at the Danebury, Iron Age fort. The data are a series of radio-carbon dates attached to pottery sherds recovered from the site. The dates are ordered and the pottery is attached to a style that corresponds to a change in production. The end dates of the phases overlap and the problem is to derive a reasonable date for the change. The data here is the output from a Metropolis-Hastings analysis that was flawed in several ways. Because of the quality of the data there were areas in the simulation that were not reached in this run and there was a limit placed on the generated values that created the straight edge observed in the projections in Figure 4.2. This particular simulation was not useful as part of the investigation of the problem, but gave data that were interesting visually.

The tooth data set. The largest canonical variable for 6 teeth from a range of anthropological sources Andrews (1972, Table 2). This data originally came from Ashton et al. (1957)

The Old Faithful data. A subset of the data from "Observations of eruptions of the Old Faithful geyser in Yellowstone National Park, USA" Weisberg (1980). These are from the eruption durations and two different size samples from the available data are used.

Five dimensional data set. This is a simulated data set the makeup of which varies depending on what is being demonstrated. If the data is anything other than a five dimensional data set consisting of five uncorrelated normal variates it's makeup is specified in the relevant section.

The Clinical trial data, the five dimensional data and the data from the archaeological problem are too large to sensibly include in the thesis, in addition some data is generated as needed, for example that in Figure 5.3. The tooth data are given in Table 4.1 and the Old Faithful data in Appendix E.

These and other data introduced in this thesis are used purely as data with no attempt at analysis, or even description, of the underlying statistics except where that is felt to be necessary in the context of the thesis.

1.3 Thesis overview

The work is divided into sections based around the three techniques used and the investigations made as follows:

Chapter 2 presents a sufficiently rigorous introduction to Bayesian statistical theory to allow the average reader to follow the remaining chapters.

Chapter 3 surveys the available forms of MCMC ("Monte Carlo Markov Chain sampling.") with a view to their use both in the analysis of intractable problems and in deriving a sample from a mathematical description of a density.

Chapter 4 surveys a range of multivariate viewing techniques, especially those

projection techniques that have proved appropriate for the current work. The requirements of minimal introduced distortion and the ability to apply some sort of density estimation to the projected data are discussed.

Chapter 5 introduces the technique of Kernel Density Estimation (KDE) from an ad hoc frequentist approach along with some of the problems that this approach introduces which make it inadequate here. The literature is surveyed with the aim of finding fast, automatic approaches to the problem.

Chapter 6 a Bayesian development of Kernel Density Estimation (KDE) is introduced that offers a novel solution to the bandwidth selection problem. This is shown to provide a powerful and flexible statistical tool without the need for the ad hoc choices often associated with these methods. The formulation of the KDE as a Bayes' problem is believed to be unique. Examples of the use of the BKDE are given including both real life (for example the Old Faithful data set) and contrived (examples from Berlinet and Devroye, 1994) data.

Chapter 7 introduces the idea of combining the Grand Tour with KDE and briefly introduces R routines that accomplish this.

The appendices contain code and data that is referred in the rest of the text.

Appendix A contains the implementations of the Grand Tour that were written, along with further explanations of interesting sections of the code and examples of how to use it within S-Plus 3.x or R.

Appendix B Presents S-Plus/R code for simple versions of the Grand Tour, three functions that implement Gauss Hermite numerical integration, written by J C Naylor, and BKDE functions that use that integration to compute the required bandwidth for the BKDE within the tours.

Appendix C has implementations of both the variable and non-variable kernel versions of BKDE in both the code written for the Fortran 77 libraries of the original Bayes4 and for the newer C++ version.

Appendix D has examples of BKDE applied to all 28 densities from Berlinet and Devroye (1994).

Appendix E has the shortened version of the Old Faithful data set and the Andrews (1972, Table 2) tooth data.

Chapter 2

The Bayesian paradigm

This chapter contains basic Bayesian results that are used in later sections. Some of the common methods used in Bayesian analysis and the problems that this thesis attempts to address are demonstrated by use of examples.

2.1 Basic theory

Consider a model for data $\boldsymbol{x}$ assumed to be observed values of some random variable $\boldsymbol{X}$. This model defines a probability distribution for $\boldsymbol{X}$ in terms of parameters $\boldsymbol{\theta}$, from a parameter space $\boldsymbol{\Theta}$, by a density function $p(\boldsymbol{x}|\boldsymbol{\theta})$. The value of this density data $\boldsymbol{x}$ is often called the likelihood function, as it describes the likelihood of this particular sample $\boldsymbol{x}$ in terms of the parameters $\boldsymbol{\theta}$.

In a Bayesian model initial knowledge about $\boldsymbol{\theta}$ is represented as a prior distribution having density $p(\boldsymbol{\theta})$. This may come from some 'expert' assessment of the parameter value or from some previous measurement or experiment.

Statistical inference for $\boldsymbol{\theta}$ is obtained by using Bayes' Theorem to update knowledge about $\boldsymbol{\theta}$ in the light of the sample data $\boldsymbol{x}$.

The inference about $\boldsymbol{\theta}$, given data $\boldsymbol{x}$, is provided by the posterior density given by Bayes' Theorem as:

$$p(\boldsymbol{\theta}|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{\int_{\boldsymbol{\Theta}} p(\boldsymbol{x}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}} = \frac{p(\boldsymbol{x}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\boldsymbol{x})} \tag{2.1}$$

where $\boldsymbol{\Theta}$ is the entire parameter space.

It is often convenient and sufficient to express Bayes' theorem to proportionality[1] as

$$p(\boldsymbol{\theta}|\boldsymbol{x}) \propto p(\boldsymbol{x}|\boldsymbol{\theta})p(\boldsymbol{\theta}). \tag{2.2}$$

Note that

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta} \tag{2.3}$$

is obtainable as the constant needed to make $p(\boldsymbol{\theta}|\boldsymbol{x})$ a proper density, so that

$$\int p(\boldsymbol{\theta}|\boldsymbol{x}) = 1. \tag{2.4}$$

If the sample is large then the information contained in the prior is swamped by that in the data and the prior has little effect on the posterior density.

[1]The variable y is said to be proportional to the variable x, if there exists a non-zero number k such that $y = kx$, the relation is often denoted $y \propto x$ and the constant ratio $k = y/x$, is called the constant of proportionality.

If, on the other hand, the sample information is small the posterior will be dominated by the prior.

The Bayesian approach has several theoretical advantages over, for example, the more familiar frequentist methods. One such is that it does not violate the *likelihood principle*. This principle implies that all the information to be learned about $\boldsymbol{\theta}$ from the sample is captured in the likelihood (Lindley, 1965). Hence, two different samples having proportional likelihoods would have the same inference; this is true if Bayesian methods are used, see, for example, Savage (1962) and O'Hagan (1994). As a simple example of a statistic in common use that violates the likelihood principle consider the problem of estimating the variance (σ^2) from a sample:

$$s^2 = \frac{\sum(x_i - \bar{x})^2}{n-1} \tag{2.5}$$

where Given a sample mean $\bar{x}$ and sample size n, s^2 is an unbiased estimator for σ^2. The denominator has been chosen to remove bias. It therefore considers samples that have not been seen and, hence, information not in the observed data.

2.1.1 Marginal and conditional densities

Although the posterior density given by (2.1) provides all that is needed for inference about $\boldsymbol{\theta}$, there may be particular interest in a subset of $\boldsymbol{\theta}$, $\boldsymbol{\theta}_I$ of dimension[2] t. where

[2]A quantity $\boldsymbol{\theta}$ is said to be of dimension d if it may equally well be written $\{\theta_1, \theta_2, \ldots, \theta_d\}$, i.e. it consists of d distinct items.

$$\boldsymbol{\theta}_I = (\theta_{i_1}, \theta_{i_2} \ldots, \theta_{i_t})^T \tag{2.6}$$

and, if $\boldsymbol{\theta}$ is of dimension k,

$$I = (i_1, i_2, \ldots, i_t) \subset (1, 2, \ldots, k). \tag{2.7}$$

Denoting the complement of $\boldsymbol{\theta}_I$ with respect to $\boldsymbol{\theta}$ as $\boldsymbol{\theta}_I^c$, then the Bayesian paradigm gives inference for $\boldsymbol{\theta}_I$ as the *marginal posterior density*

$$p(\boldsymbol{\theta}_I|\boldsymbol{x}) = \int_{\boldsymbol{\Theta}_{k-t}} p(\boldsymbol{\theta}|\boldsymbol{x}) d\boldsymbol{\theta}_I^c, \tag{2.8}$$

where Θ_{k-t} is the parameter space supporting $\boldsymbol{\theta}_I^c$, i.e. the appropriate region of integration for the subset $\boldsymbol{\theta}_I^c$ of $\boldsymbol{\theta}$.

In a similar way, inference about $\boldsymbol{\theta}_I$, when $\boldsymbol{\theta}_I^c$ are known, is given by the *conditional posterior density*

$$p(\boldsymbol{\theta}_I|\boldsymbol{\theta}_I^c, \boldsymbol{x}) = \frac{p(\boldsymbol{\theta}|\boldsymbol{x})}{p(\boldsymbol{\theta}_I^c|\boldsymbol{x})} \tag{2.9}$$

2.1.2 Predictive densities

Similarly, inference about future observations $\boldsymbol{y}$, having observed $\boldsymbol{x}$ is given by the posterior predictive density

$$p(\boldsymbol{y}|\boldsymbol{x}) = \int_{\boldsymbol{\Theta}} p(\boldsymbol{y}|\boldsymbol{\theta}) p(\boldsymbol{\theta}|\boldsymbol{x}) d\boldsymbol{\theta} \tag{2.10}$$

where $p(\boldsymbol{\theta}|\boldsymbol{x})$ is the full posterior distribution. Note that the predictive density $p(\boldsymbol{y}|\boldsymbol{x})$ provides inference about $\boldsymbol{y}$ conditional only on the observed data

$\boldsymbol{x}$, without reference to any specific value of $\boldsymbol{\theta}$. This is in contrast to an estimative approach which might give $p(\boldsymbol{y}|\hat{\boldsymbol{\theta}})$ where $\hat{\boldsymbol{\theta}}$ is a point estimate (e.g. maximum likelihood).

2.1.3 Analytic solutions

The range of problems for which an Bayesian, analytic solution is possible is limited and depends on the choice of the model (leading to a tractable likelihood function) and prior.

A simple model assumes that the data $\boldsymbol{x} = (x_1, \ldots, n_n)$ is a random sample, of size n, from some distribution having density function $p(\boldsymbol{x}|\boldsymbol{\theta})$ with

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \prod_{i=1}^{n} p(x_i|\boldsymbol{\theta}). \tag{2.11}$$

If a prior $p(\boldsymbol{\theta})$ is chosen from some family $\mathcal{F}$, say $f(\boldsymbol{\theta}|\boldsymbol{\alpha})$, the choice of $\boldsymbol{\alpha}$ being regarded as part of the model specification, the posterior is given by

$$p(\boldsymbol{\theta}|\boldsymbol{x}) = \frac{\prod_{i=1}^{n} p(x_i|\boldsymbol{\theta}) f(\boldsymbol{\theta}|\boldsymbol{\alpha})}{\int_{\boldsymbol{\Theta}} \prod_{i=1}^{n} p(x_i|\boldsymbol{\theta}) f(\boldsymbol{\theta}|\boldsymbol{\alpha}) d\boldsymbol{\theta}}. \tag{2.12}$$

If $p(\boldsymbol{\theta}|\boldsymbol{x})$ is also a member of $\mathcal{F}$, say

$$p(\boldsymbol{\theta}|\boldsymbol{x}) = f(\boldsymbol{\theta}|\boldsymbol{\beta}) \tag{2.13}$$

where the parameter $\boldsymbol{\beta}$ is a function of only $\boldsymbol{\alpha}$ and $\boldsymbol{x}$ then the family $\mathcal{F}$ is said to be *closed under sampling*, with respect to the density $p(x|\boldsymbol{\theta})$ (Barnard, 1949). The prior $p(\boldsymbol{\theta}|\boldsymbol{\alpha})$ is called a *conjugate prior* for $p(x|\boldsymbol{\theta})$ (see Smith and Bernardo, 1994, p. 265).

Example

As an illustration, consider inference about the mean θ of an exponential distribution with density

$$p(x|\theta) = \theta^{-1}e^{-x/\theta} \qquad (x > 0,\ \theta > 0). \tag{2.14}$$

The likelihood of a random sample $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^T$ is

$$p(\boldsymbol{x}|\theta) = \theta^{-n}e^{-n\bar{x}/\theta} \tag{2.15}$$

in terms of the sufficient statistic[3]

$$\bar{x} = \sum_{i=1}^{n} \frac{x_i}{n}. \tag{2.16}$$

Take as prior a density of the form

$$\begin{aligned} p(\theta) &= f(\theta|\alpha_1, \alpha_2) && (2.17)\\ &= \frac{\alpha_2^{\alpha_1 - 1}}{\Gamma(\alpha_1 - 1)}\theta^{-\alpha_1}e^{-\alpha_2/\theta} && (2.18)\\ &\propto \theta^{-\alpha_1}e^{-\alpha_2/\theta} && (2.19) \end{aligned}$$

then the corresponding posterior is

[3]Let $\boldsymbol{x} = (x_1, \ldots, x_n)^T$ be a vector of observations from a distribution with parameters $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_k)$. Let $\boldsymbol{t} = t_1, \ldots, t_q)$ be q functions of $\boldsymbol{x}$. Then the set of statistics $\boldsymbol{t}$ is said to be sufficient for $\boldsymbol{\theta}$ if the likelihood function $l(\boldsymbol{\theta}|\boldsymbol{x})$ can be expressed in the form

$$l(\boldsymbol{\theta}|\boldsymbol{x}) \propto g(\boldsymbol{\theta}|\boldsymbol{t})$$

$$p(\theta|\boldsymbol{x}) \propto \theta^{-\alpha_1} e^{-\alpha_2/\theta} \theta^{-n} e^{-n\bar{x}/\theta} \quad (2.20)$$

$$= \theta^{-(\alpha_1+n)} e^{-(\alpha_2+n\bar{x})/\theta} \quad (2.21)$$

which is clearly also of the form (2.17) and so may be written as

$$p(\theta|\boldsymbol{x}) = f(\theta|\alpha_1', \alpha_2') \quad (2.22)$$

where

$$\alpha_1' = \alpha_1 + n \text{ , and } \alpha_2' = \alpha_2 + n\bar{x} \quad (2.23)$$

Hence (2.14) is closed under sampling, with (2.17) as a convenient natural conjugate prior family for inference about the mean θ. In such a case a complete analytic solution is available.

2.2 Implementation Example

The cases in which a convenient conjugate prior is available are few, and ideally the choice of model and prior should not be limited to those that allow an analytical solution. Perhaps the best way to allow the reader to appreciate the range of alternative approaches, and the difficulties that arise, is to examine a simple example. A small deviation from that example (replacing the Normal prior with a Teachers prior with $\nu = 4$) makes it impossible to treat the example analytically but makes little difference to numerical methods.

Consider the manufacture of some component in which a particular measurement, x, is of interest. This may be modeled as being a value of a

random variable X having a Normal distribution with mean μ and variance σ^2, $N(\mu, \sigma^2)$. In this context the mean μ represents the 'setting' of the manufacturing process, while the variance σ^2 represents the 'process variability'.

The parameter μ is unknown, but prior knowledge about it may be represented by a $N(\mu_0, \sigma_0^2)$ density, and for this example values of $\mu_0 = 3.5,\ \sigma_0^2 = 1$ are suitable. The choice of σ_0^2 is in fact largely uninformative as a wide range (0.5 to 6.5 say) of values of μ are all quite likely.

Assume the parameter σ is known: $\sigma = 0.01$.

The sample data to hand

3.51 3.50 3.52 3.50 3.51 3.50 3.52 3.50 3.51 3.51

can be summarised as $n = 10,$ and $\sum x_i = 35.08$.

For this example both analytic and numerical solutions are readily available.

Analytic approach

The interest is in inference about $\boldsymbol{\theta} = (\mu)$ given $\sigma = 0.01$.

With prior $\mu \sim N(3.5, 1)$

$$p(\mu) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(\mu - 3.5)^2} \tag{2.24}$$

$\bar{x}$, the sample mean, is a sufficient statistic for μ, as the likelihood, $l(\mu|\boldsymbol{x})$, depends only on $\boldsymbol{x}$ and σ (fixed), through the sampling distribution of $\bar{x}$ which is $N(\mu, \frac{\sigma^2}{n})$

$$l(\mu|\boldsymbol{x}) \propto p(\bar{x}|\mu,\sigma^2) \tag{2.25}$$
$$\propto \exp\left(\frac{-n}{2\sigma^2}(\mu-\bar{x})^2\right) \tag{2.26}$$

so the posterior for μ is

$$p(\mu|\boldsymbol{x}) \propto p(\mu)l(\mu|\boldsymbol{x}) \tag{2.27}$$
$$\propto p(\mu)p(\bar{x}|\mu,\sigma^2). \tag{2.28}$$

So, by inspection

$$p(\mu|\boldsymbol{x}) = \left(\frac{n}{2\pi\sigma^2}\right)^2 \exp\left(\frac{-n}{2\sigma^2}(\mu-\bar{x})^2\right) \qquad -\infty < \mu < \infty \tag{2.29}$$

$\boldsymbol{x}$ is a random sample of size n from $N(\mu,\sigma^2)$ where σ is known and the prior for μ is $N(\mu_0,\sigma_0^2)$. The posterior is $\mu \sim N(\mu_n,\sigma_n^2)$ where

$$\mu_n = \frac{n\bar{x}/\sigma^2 + \mu_0/\sigma_0^2}{n/\sigma^2 + 1/\sigma_0^2} \tag{2.30}$$

and

$$\sigma_n^{-2} = n\sigma^{-2} + \sigma_0^{-2}. \tag{2.31}$$

The prior for μ is $N(3.5,1)$, $\sigma = 0.01$ and $\bar{x} = 3.508$ so posterior belief about μ is as if μ had a $N(3.508, 0.00316^2)$ distribution.

Simulation approach

It is possible to utilise Monte Carlo integration using Markov Chains (MCMC) to obtain inference about population statistics in many problems. An MCMC solution to a problem consists of an MCMC constructed so that its stationary distribution is the distribution about which it is wished to draw inference, (see, for example, Gilks et al., 1996). The problem outlined in section 2.2 is amenable to this approach. Specifics of the method used are discussed in Chapter 3.

As a simple example of the use of MCMC to solve the above problem choose a Metropolis-Hastings, random walk MCMC. This has the advantage of allowing the use of a symmetrical distribution from which to choose the candidate.[4] Once the process has entered the state space of the target distribution[5] it will not leave it.

Here

$$\begin{aligned} p(\theta) &\sim N(3.5, 1) \\ l(\theta|\boldsymbol{x}) &\propto \exp\left(\tfrac{-n}{2\sigma^2}(\theta - \bar{x})^2\right) \\ &\propto p(\bar{x}|\theta, \sigma) \end{aligned} \tag{2.32}$$

so the probability of transition from θ_1 the current state to θ_2 the candidate state is

[4] If there exists a generated sequence of values and the current value is X_i, to generate X_{i+1}, the process is to generate a possible value for X_{i+1}, for example y, and then decide whether the chain moves to that value or stays in the current value. The value y is called the candidate.

[5] The current value of the chain is called the state and the state space of the distribution is the space consisting of all possible values the distribution might take.

$$p(\theta_1, \theta_2) \quad = \quad \min\left(\frac{p(\theta_2)p(\bar{x}|\theta_2,\sigma)}{p(\theta_1)p(\bar{x}|\theta_1,\sigma)}, 1\right) \tag{2.33}$$

The code shown in figure 2.2 is for running a Metropolis-Hastings, random walk MCMC sampler. With the transition probability above, and allowing 1000 samples as a burn in period, the output is a sample of 1000 values with $\hat{\mu} = 3.507999$ and $sd = 0.00304$. (Note that this result would, of course, be slightly different if run again, due to the small sample used and random differences. However, if a large enough sample is taken, run differences tend to zero with probability 1).

It is possible to draw a predictive sample from this density with the S-Plus command `y <- rnorm(1000, samp, 0.01)`[6], where `samp` is the output from the sampler, this gives the histogram shown in Figure 2.1.

Asymptotic approach

As above the likelihood is

$$l(\theta|\boldsymbol{x}) \propto \exp\left(\frac{-n}{2\sigma^2}(\mu - \bar{x})^2\right) \tag{2.34}$$

so the log-likelihood is

$$L(\theta|\boldsymbol{x}) = C - \frac{n}{2\sigma^2}(\mu - \bar{x})^2 \tag{2.35}$$

[6] `rnorm(n, m, s)` is the S-Plus command that generates n random numbers mean m, standard deviation s. So this generates 1000 samples using the MCMC sample from the posterior distribution of the mean as a vector of means (`samp`) and the known parameter σ as standard deviation.

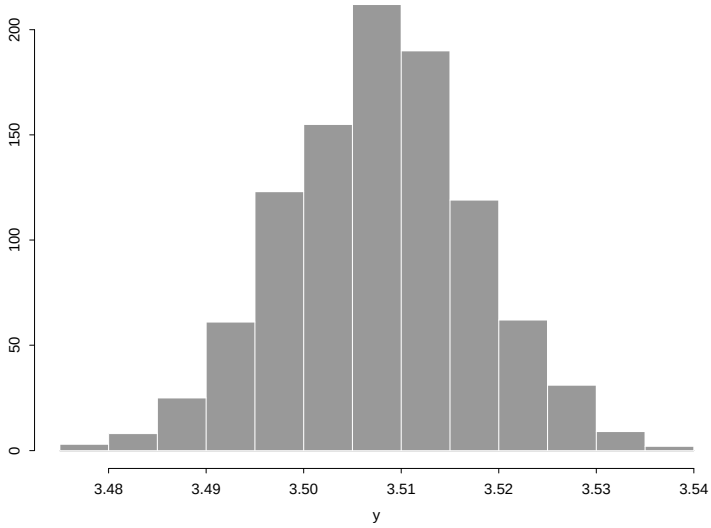


Figure 2.1: *Histogram of a predictive sample,* $n = 1000$*, generated using the R code* `y <- rnorm(1000, samp, 0.01)`*.* `samp` *was generated from the density* $\mu \sim N(3.507999, 0.00304^2)$ *using the code in figure 2.2. Axes are values (x) and bin counts (y).*

where C is a constant. Differentiating and equating to zero gives

$$\frac{n}{\sigma^2}(\theta - \bar{x}) = 0 \tag{2.36}$$

so that

$$\hat{\theta} = \bar{x} \tag{2.37}$$

and a second differentiation gives

$$\sigma_n^{-2} = \frac{n}{\sigma^2} \tag{2.38}$$

```
function(theta, w = 1000, mu = 3.508, sigma = 0.01, n = 10)
{
#simple mh sampler for the example with mean 3.05
#sigma 0.01
    f <- function(theta, mu = 3.508, sigma = 0.01, n = 10)
    {
        z <- exp( - n/2/sigma^2 * (theta - mu)^2)
        z
    }
    z <- rep(0, n)
    p <- rep(0, w + n)
    theta <- 3.5
    pmu <- 3.5
    psigma <- 1
    for(i in 1:(w + n)) {
        new <- theta + rnorm(1, 0, 0.01)
        prob<-(dnorm(new,pmu,psigma)*f(new))
            /(dnorm(theta,pmu,psigma)*f(theta))
        alpha <- min(prob, 1)
        p[i] <- alpha
        if(runif(1) < alpha) theta <- new
        if(i > w) z[i - w] <- theta
    }z
}
```

Figure 2.2: *R code for the MCMC example.*

giving, in this case, $\mu = 3.508$ and $\sigma_n = 0.00316$.

A histogram of 1000 samples from $N(3.508, 0.00316^2)$ is shown in figure 2.3. While not a good density estimator, a histogram is useful for the gross comparison of two samples needed here. Note that this histogram appears to come from a distribution with a smaller standard deviation than that shown in Figure 2.1; the predictive sample is a safer estimate as it allows for the uncertainty in the point density estimation.

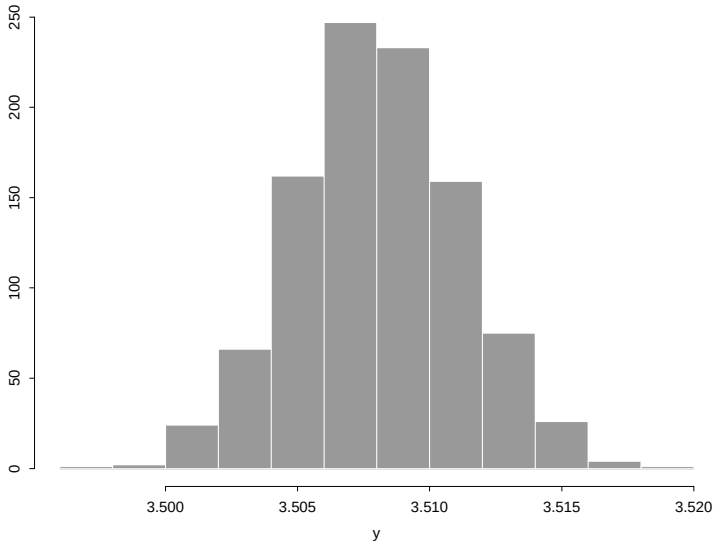


Figure 2.3: *Histogram of sample drawn from* $\mu \sim N(3.507999, 0.00304^2)$ *(Maximum likelihood approximation). Axes are values (x) and bin counts (y).*

Numerical approach

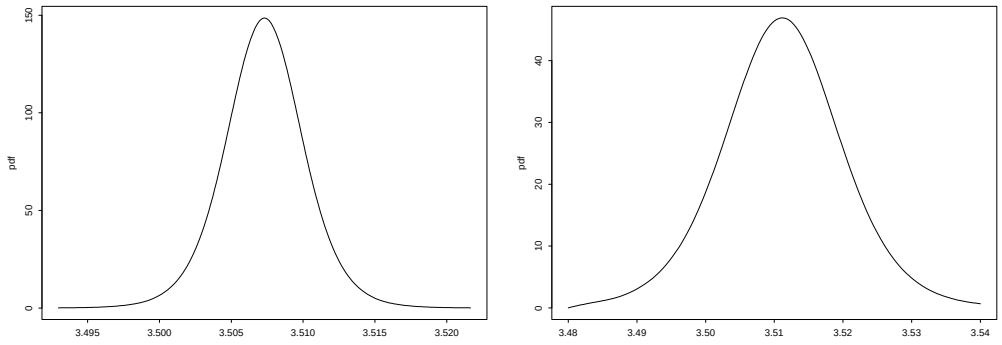
The interest is in what may be inferred about $\boldsymbol{\theta} = (\mu)$ from a random sample $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^T$. Each component of the random sample is modelled as a value of X, being independently and identically distributed as $N(\mu, \sigma^2)$. Hence the probability density function (pdf) for the whole sample, the likelihood function, can be written as

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x_i - \mu}{\sigma}\right)^2\right). \tag{2.39}$$

The prior is as given above, which may, in practice, be a setting up value (μ) and a measure of variation derived from system history (σ^2).

It is possible to numerically apply Bayes' Theorem. Of the many approaches

to numerically applying Bayes' Theorem, the Bayes4 suite of software developed by Dr J C Naylor (see Naylor, 1982; Naylor and Shaw, 1983; Naylor and Smith, 1982) is chosen here. Bayes4 uses Gauss-Hermite quadrature to obtain values for posterior density parameters. Applying Bayes4 gives the posterior densities shown in Figure 2.4 and also a numerical approximation to the posterior mean for μ of 3.508.

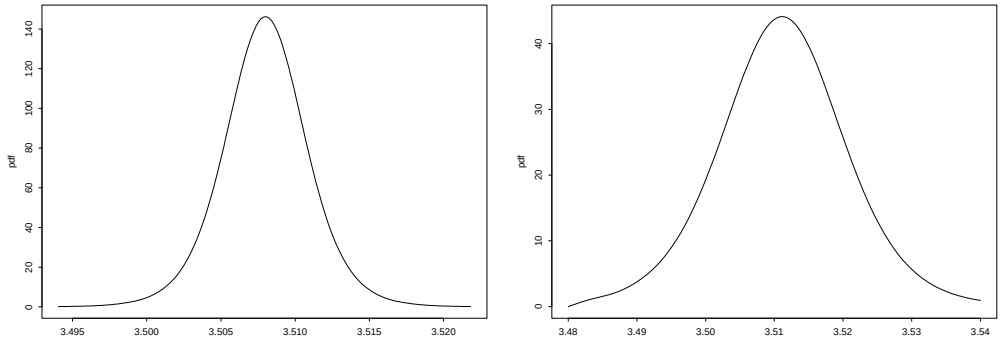


(a) Posterior Mean. (b) Predictive density.

Figure 2.4: *Posterior mean and predictive density for μ generated by Bayes4, Normal prior. Axes are μ (x) and pdf (y).*

Repeating the exercise using a Student's t-distribution with 4 degrees of freedom ($t_{\nu=4}$) as prior, renders the calculation analytically intractable. However, using Bayes4 gives the posterior densities shown in figure 2.5, where the posterior estimate for μ is again 3.508.

All of the examples above are extremely easy to describe. However, as soon as a $t_{\nu=4}$ prior is chosen their solution becomes, analytically at least, intractable. At the same time the result of the analysis is remarkably similar to that obtained with a Normal prior. Performing the analysis using Bayes4 is no more difficult than when a Normal prior is used.



(a) Posterior Mean.

(b) predictive density.

Figure 2.5: *Posterior mean and predictive density for μ generated by Bayes4, $t_{\nu=4}$ prior. Axes are μ (x) and pdf (y).*

The graphs in Figures 2.4 and 2.5 were produced using standard S-Plus routines, which apply spline curves, to interpolate between points.

The problem analysed in the preceding sections is univariate and simple; the use of these methods is complete overkill but is done for comparison. In each case the posterior means and the predictive densities are the same, as is the inference to be drawn from the data. If the problem exceeds 3 parameters, the methods of generating the posterior samples are still valid, as long as they can be applied to the problem. However, the display techniques used are not available for exploring the results derived from them. A uniform way of presenting results from higher dimensional objects would be useful.

2.3 Discussion

In the chosen example several different approaches were taken to a single, simple problem. The key issue being considered here is that the form in

	Mean	Variance
Analytic	3.508000	0.0031600^2
Gibbs Sampler	3.507999	0.0030400^2
Predictive mean and variance	3.508014	0.0031168^2
Asymptotic	3.508000	0.0031600^2
Bayes4 Normal Prior	3.508000	0.0043700^2
Bayes4 t Prior	3.508000	0.0043600^2

Table 2.1: *Results for posterior densities for manufacturing problem.*

which the answer is available depends on the approach taken to solving the problem. There is a mathematical function, some pseudo sample data, an asymptotic expansion, and a numerical approximation. The rest of this thesis is concerned with presentation methods which may be used across all of these implementations.

Note that each of the above approaches gives a similar answer to the problem. Table 2.1 summarises the posterior densities obtained for the posterior mean $E(\mu|\boldsymbol{x})$ and posterior variance $E(\mu^2|\boldsymbol{x}) - E^2(\mu|\boldsymbol{x})$ by each method:

If the posterior mean is used as a point estimate, all the methods show reasonable agreement. Point estimates are useful, but decreasingly so as dimensionality increases. As the dimensionality of the problems increases so does the difficulty of presenting the results, due to the difficulty in assessing interactions of densities of higher than 2 dimensions. This makes the

application of non-analytic methods such as MCMC estimation attractive.

Chapter 3

Monte Carlo Markov Chain Sampling

The example in the previous chapter was trivial but the change in prior from Normal to Student's t rendered it intractable to analytic methods without making either numerical quadrature or sampling more difficult. The main limitation to the use of both of these has been their computational intensity but, as Moore's Law (Moore, 1965) predicts, this is becoming less of a problem.

In Bayesian analysis the problem dimensionality is the same as the number of parameters of the problem. In problems of up to 10 parameters it is possible to use Monte Carlo integration or numerical quadrature (Smith et al., 1987; Naylor, 1982; Naylor and Smith, 1988), beyond that it rapidly becomes difficult. Monte Carlo Markov Chain (MCMC) algorithms – especially Gibbs (Geman and Geman, 1984) and Metropolis-Hastings (Hastings, 1970; Metropolis et al., 1953) – are now commonly used when it is required to produce a sample from a density. These methods can be used when the

density is specified either as a complete set of conditional densities or as a multivariate density and are very powerful, able to handle problems of the very high dimensionality needed.

The basic strategy is to construct a Markov Chain sampling scheme for which it can be shown that the equilibrium distribution is the density of interest. The great advantage of MCMC is that it is possible to devise methods such that the density of interest need only be known to proportionality, thus removing the requirement for knowledge of the constant of proportionality and the troublesome integration required to obtain it. Interest here is in a sample obtained from MCMC as the result of an analysis or from a posterior density, specified as either a full set of conditionals (Gibbs) or a complete density (Metropolis-Hastings).

3.1 Markov Chains

Some Theory

Consider a sequence of random variables $X_0, X_1, \ldots$ representing the state of some system at times $0, 1, \ldots$. These define a stochastic process, said to be in state i at time n if $X_n = i$. Suppose the system is in state i at time n and the transition probability that it will be in state j at time $n+1$ is P_{ij}, then

$$P_{ij} = P\{X_{n+1} = j | X_n = i, X_{n-1} = i_{n-1}, \ldots, X_1 = i_1, X_0 = i_0\}$$

.

If this depends only on the state i and on no previous state, i.e.

$$P_{ij} \quad = \quad P\{X_{n+1} = j | X_n = i\},$$

then this is called a Markov process[1]. The values P_{ij} are called the transition probabilities of the Markov Chain and the process is said to have no memory in the sense that P_{ij} depends only on $X_n = i$.

Within the state space of the process, if $(X_t)_{t\geq 0}$ is a Markov chain it is said that i leads to j (written as $i \rightarrow j$) if

$$P_i(X_t = j \text{ for some } t \geq 0) > 0. \tag{3.1}$$

Where $P_i(\cdot)$ is the probability of some event happening, given that the chain is in state i. It is held that i communicates with j (written $i \leftrightarrow j$ if both $i \leftarrow j$ and $i \rightarrow j$. This is an equivalence relation, having reflexivity, symmetry and transitivity. Sets of states that can communicate with each other can be considered to be equivalence classes. States belonging to different equivalence classes do not communicate, though one way transition is possible. If a Markov Chain has all its states belonging to one equivalence class it is said to be irreducible.

In practical terms, if the process is not irreducible then it can become "stuck" in one section and cannot be said to produce a "fair" representation of the density. An irreducible process is one in which it is possible to reach any state, starting in any state (Feller, 1970, p 385).

[1] A process which has no memory, i.e. only the current state of the process influences where it goes next, is called a *Markov process*. If this process can assume only a finite or countable set of states it is usual to refer to it as a *Markov chain* (see Norris, 1997)

It can be shown that the sample obtained from the MCMC tends to a sample from the distribution of interest after some (large) number of steps n (Gilks et al., 1996).

Example

Perhaps the simplest possible example of a Markov Chain is the random walk. The state space is the set of integers (positive, negative and zero) and the transition probabilities are

$$\begin{aligned} P_{ij} &= q \qquad for\ j = i+1 \\ &= 1-q \quad for\ j = i-1 \\ &= 0 \qquad\quad \text{otherwise} \end{aligned}$$

This process can be viewed as that of a particle constrained to move in single steps up or down an infinitely long line, with constant probability of moving up (q) or down ($1-q$) one step. The particle cannot take two or more steps at a single time point. Such a process is a 1D 'random walk' in discrete time.

3.2 Practical Methods of MCMC

Some methods that are commonly used in constructing MCMC samplers are now considered. Interest is in methods that lead towards a system for viewing the results of analysis in some uniform way, regardless of the method of analysis. The possible starting points for this can be separated into four cases. Knowledge about the density is contained in:

1. A sample from some simulation analysis.

This could be a sample from an MCMC simulation.

2. A set of conditional densities.

   For example those used to generate Figure 3.3.

3. A fully specified density.

   For example $f(x,y) = (2\pi\sigma\sigma_1)^{-1} \exp\left[-\frac{1}{2}\left\{\frac{(x-\mu_1)^2}{\sigma_1^2} + \frac{(y-\mu)^2}{\sigma^2}\right\}\right]$.

4. Some combination of 2 and 3.

   For example if $p(x|y_1,y_2)$ and $p(y_1,y_2) = f(y_1,y_2)$ (see Gilks and Best, 1995, for a practical approach to combining samplers) .

In cases 2, 3 and 4 some form of simulation is needed to obtain a representative sample, for 2 and 4 Gibbs sampling is used (Smith and Roberts, 1993; Gilks et al., 1996) and for case 3 Metropolis-Hastings sampling, (Metropolis et al., 1953; Hastings, 1970; Chib and Greenberg, 1994). For 4, Gibbs and Metropolis-Hastings samplers can be combined, (Gilks and Best, 1995).

There are other techniques that are used for analysis and sampling, however all are special cases of the general framework discussed in Metropolis et al. (1953) and Hastings (1970). Following are details of those that are most interesting here.

3.2.1 Gibbs samplers

Gibbs samplers are the most popular and versatile of the MCMC methods discussed here, introduced to mainstream statistical analysis by Geman and Geman (1984) and developed in, amongst others, Gelfand and Smith (1990b). Gibbs samplers were originally used to analyse Gibbs distributions on lattices

and in statistical physics where they were known as the *heat bath algorithm*. For an introduction and history of MCMC samplers see Gilks et al. (1996, Chapter 1).

Theory

Given a vector of random variables $T = (X, Y_1, \ldots, Y_2)$, some distribution $f(T) = f(x, y_1, \ldots, y_n)$, and the requirement to find characteristics (for example, but not limited to, position and shape parameters) of the marginal density

$$f(x) = \int \ldots \int f(x, y_1, \ldots, y_n) dy_1 \ldots dy_n,$$

then the obvious route is to calculate $f(x)$ and use it to find the required information. However, there are few cases where the integrations can be performed.

The Gibbs sampler provides an alternative method of obtaining a sample $X_1, \ldots, X_m \sim f(x)$ without having access to a direct mathematical description of $f(x)$. By simulating a large enough sample, the mean, variance and any other characteristic of $f(x)$ can be found. The MCMC converges to a sample from the required density and, assuming good quality random numbers, as the sample gets larger the effect on the precision of the estimates of a single extra sample on the values decreases. This means that by taking sufficient samples, values can be estimated to any desired degree of precision. This is contrary to the experience of sampling from a 'real' population where a sample can only increase to the size of the population; in MCMC there is no limit on the size of the sample. For example if some value that has a value in the region of 1 is being estimated and 1000000 samples which vary between 1.001 and 0.999 are taken, then the effect of adding another sample

will only appear in the 8th decimal place of the answer.

The end results of any calculation, although based on a simulation, are the population quantities. For example $\overline{f(x)} = 1/m \sum_{i=1}^{m} X_i$ as

$$\lim_{m\to\infty} \frac{1}{m} \sum_{i=1}^{m} X_i = \int_{-\infty}^{\infty} x f(x) dx = E[X]$$

By taking m large enough, any population statistic can be obtained to any required degree of precision.

As an example, starting with the pair of random variables (RVs) (X, Y), the Gibbs sampler generates the following sequence from $f(x)$ by sampling from the conditional distributions $f(x|y)$ and $f(y|x)$

$$Y'_0, X'_0, Y'_1, X'_1, Y'_2, X'_2, \ldots, Y'_k, X'_k$$

the initial value $Y'_0 = y'_0$ is specified, the remainder are obtained by alternatively generating

$$X'_j \sim f(x|Y'_j = y'_j)$$

$$Y'_{j+1} \sim f(y|X'_j = x'_j)$$

This is, clearly, a Markov Chain as it fulfils the condition, laid down in section 3.1, that the current state depends only on the previous state and no other. Under quite general conditions, (see Brooks and Roberts, 1995), the distribution of X'_k converges to $f(x)$ (the true marginal of X) as $k \to \infty$. Thus for k large enough, the final point of the sequence is effectively a sample point from $f(x)$. Also, the distribution of (X'_k, Y'_k) converges to $f(x, y)$.

Practice

The Gibbs sampler is a system that allows the generation of a sample from a density defined in terms of all its full conditional densities. Smith (1991) suggested that MCMC benefited from running many chains in parallel and the S-plus system's ability with vectorised arithmetic makes running multiple chains easy.

The density is approximated in the following way:

1. Obtain all the full conditional densities in the form $f(\theta_i|\boldsymbol{\theta}_{-i})$ where $\boldsymbol{\theta}_{-i}$ is the $n-1$ vector consisting of the n parameter vector of the distribution excluding θ_i.

2. Given values for the parameters $\theta_1^{(0)}, \theta_2^{(0)} \ldots \theta_n^{(0)}$ use the conditional distribution $g(\theta_1|\theta_{-1})$ to obtain a new estimate for θ_1, $\theta_1^{(1)}$.

3. Use the estimated parameters $\theta_3^{(0)}, \theta_4^{(0)} \ldots \theta_n^{(0)}$ and the new value $\theta_1^{(1)}$ to obtain a new value for θ_2, $\theta_2^{(1)}$.

4. Repeat, to obtain a complete realisation of $\boldsymbol{\theta}^{(1)}$.

Once the new realisation of $\boldsymbol{\theta}$ has been obtained the process is repeated. Theory states that the algorithm converges to a sample from the density, however the convergence can be extremely slow and it is common to throw away a large number of samples prior to establishing convergence using one of the well established tests (see, for example, Brooks and Roberts, 1995).

Figure 3.1 shows the first 5 samples, including the starting position $(20, -20)$, of a simple Gibbs sampler. The four frames show the progression from the start, each frame showing the two samples obtained in each step from the two

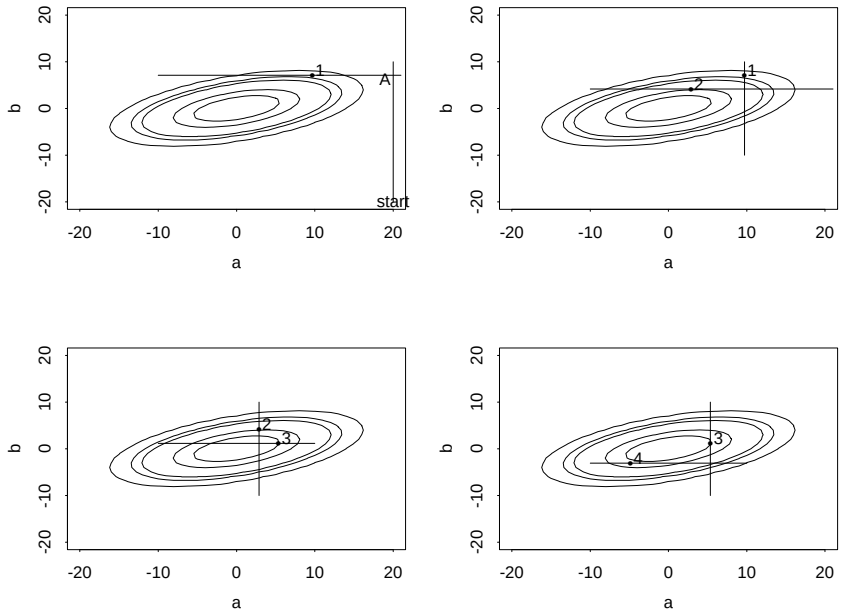


Figure 3.1: *The first 5 samples (including the starting position* $(20, -20)$*) of a Gibbs sampler in state space* $\Re^2$*. The distribution sampled from is* $a \sim N(0, 5),\ b \sim N(0, 2.5),\ \rho_{ab} = 0.5$*. The contour lines are percentage contour lines for the target density.*

conditional densities (following the line from start in the first diagram leads from the point $(x^{(0)}, y^{(0)})$ to the point $(x^{(1)}, y^{(0)})$ and the next line leads to the point $(x^{(1)}, y^{(1)})$). Note that the start point can be anywhere in the state space of the distribution. The chain quickly enters the 90% contour line[2] shown for the target distribution; in the case of a more complex distribution it can take many more steps for the chain to reach this point. The chain converges to the required distribution but it is sometimes difficult to predict the rate of convergence. The S-Plus code used to generate this sampler is

[2]The 90% contour line is the line within which 90% of a sample from the density is expected to lie.

shown in figure 3.2.

```
function(x = c(0, 0, 5), y = c(0, 0, 2.5), rho = 0.5, n = 20, burn
= 0) {
    # set up the constants
    FAC <- sqrt(1 - rho^2)      #
    n <- n + burn
    # plot the contours
    con <- normgrid(a, b, mu = c(x[2], y[2]),
                sig = c(x[3], y[3]), rho = rho)
    contour(a, b, con, labex = 0, levels = c(0.1,
        0.05, 0.01, 0.005, 0.001))      #
    # generate the samples
    for(i in 1:n) {
        # generate a new Y from the current X
        ynew <- rnorm(1, (rho * y[3])/x[3] * x[1], FAC * y[3])
        # generate a new X from the new Y
        xnew <- rnorm(1, (rho * x[3])/y[3] * ynew, FAC * x[3])
        # output if past the burn in,
        # in this case output the number i
        if(i > burn)
            text(x[1], y[1], as.character(i - burn))
        # update the output
        x[1] <- xnew
        y[1] <- ynew
        out <- c(out, c(x[1], y[1]))
    }
    # output the plotting info for reuse
    out <- list(out, con)
    out
}
```

Figure 3.2: *S-Plus code for a simple Gibbs sampler.*

The density shown in figure 3.1 is a bivariate Normal with means zero, variances 2.5 and 5 and covariance $\rho = 0.5$. The well-known equations for the two conditional densities for such a density may be found, for example, in

Feller (1970) and are:

the conditional density

$$p(X_2|X_1) = \frac{1}{\sqrt{2\pi(1-\rho^2)\sigma_2^2}} exp\left[-\frac{(x_2 - \rho(\sigma_2/\sigma_1)x_1)^2}{2(1-\rho^2)\sigma_2^2}\right],$$

the conditional expectations

$$E(X_2|X_1) = \rho(\sigma_2/\sigma_1)x_1,$$

$$E(X_1|X_2) = \rho(\sigma_1/\sigma_2)x_2,$$

and the variances

$$\text{var } (X_2|X_1) = (1-\rho^2)\sigma_2^2,$$

$$\text{var } (X_1|X_2) = (1-\rho^2)\sigma_1^2.$$

The 10,000 samples in Figure 3.3 were generated in a similar manner, however this time the S-Plus routines make use of the ability of S-Plus to handle vectorised arithmetic to reduce the use of the for loop. This gives a much more efficient program. The code used is shown in Figure 3.4.

3.2.2 The Metropolis-Hastings Algorithm

The Metropolis-Hastings algorithm is a method first proposed in Metropolis et al. (1953) and generalised in Hastings (1970).

Theory

If the target density $\pi(x)$ is known to proportionality, and a Markov process with candidate-generating probability $q(x, y)$ is constructed, so the probability of transition from state x to state y is $q(x, y)$, then, with the chain at

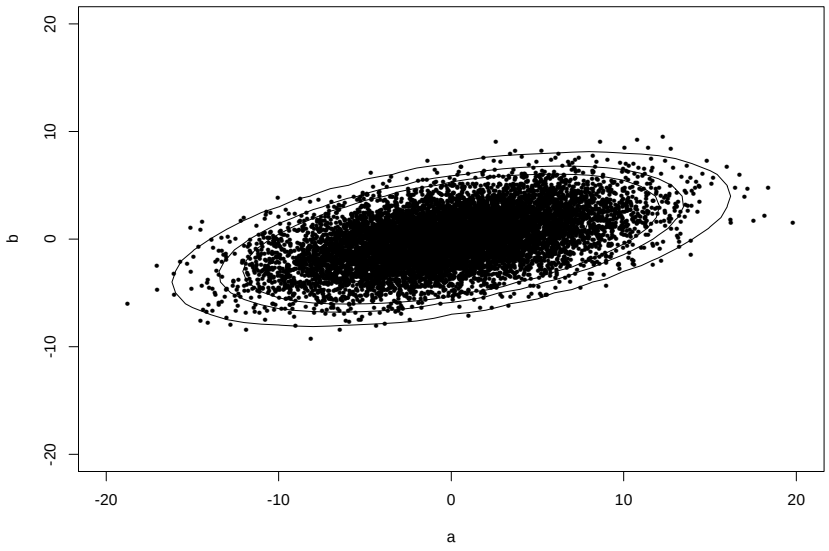


Figure 3.3: *10,000 samples from a Gibbs sampler in state space* $\Re^2$*. The distribution sampled from is* $a \sim N(0,5),\ b \sim N(0,2.5),\ \rho_{ab} = 0.5$*. The contour lines are percentage contour lines for the target density.*

point $X_n = x_n$, a candidate value Y for X_{n+1} is generated from the density $q(x,y)$. This candidate is accepted with probability $\alpha(x,y)$ where

$$\alpha(x,y) = \begin{cases} \min\left\{\frac{\pi(y)q(y,x)}{\pi(x)q(x,y)}, 1\right\} \text{ if } \pi(x)q(x,y) > 0, \\ 1 \text{ if } \pi(x)q(x,y) = 0. \end{cases}$$

$\alpha(x,y)$ is called the acceptance probability. Details of the derivation of the acceptance probability have been extensively discussed in the literature, see e.g. Chib and Greenberg (1994).

The acceptance probability only depends on π through the ratio $\pi(y)/\pi(x)$, so π is only needed to proportionality. If the step is accepted the chain moves to $X_{n+1} = y$, otherwise it remains at $X_{n+1} = x$.

```
function(x = c(0, 0, 5), y = c(0, 0, 2.5), rho = 0.5,
        n = 10000, burn = 10,cont=T)
{
    FAC <- sqrt(1 - rho^2)      #
# draw the contour lines
    if(cont)
        contour(a, b, normgrid(a, b, mu = c(x[2], y[2]),
                        sig = c(x[3], y[3]), rho = rho), labex =0,
                        levels = c(0.1, 0.05, 0.01, 0.005, 0.001))      #
# replicate the x and y data to give 1 stream for each output
    x2 <- rep(x, n)
    dim(x2) <- c(3, n)
    x2 <- t(x2)
    y2 <- rep(y, n)
    dim(y2) <- c(3, n)
    y2 <- t(y2) #
# burn in
    for(i in 1:burn) {
        ynew <- x2[,2] + rnorm(n, (rho * y2[, 3])/x2[, 3]
                        * (x2[, 1] - x2[,2]), FAC * y2[,3])
        xnew <- y2[,2] + rnorm(n, (rho * x2[, 3])/y2[, 3]
                        * (ynew - y2[,2]), FAC * x2[, 3])
        x2[, 1] <- xnew
        y2[, 1] <- ynew
    }
    points(x2, y2)
    list(x2, y2)
}
```

Figure 3.4: *Vectorised S-Plus code for a Gibbs sampler.*

If the candidate generating density is symmetric, an important special case, $q(x, y) = q(y, x)$ the probability of a move reduces to $\pi(y)/\pi(x)$, hence, if $\pi(y) \geq \pi(x)$, the chain moves to y, otherwise it moves with probability given by $\pi(y)/\pi(x)$. In other words, if the jump is to a point with higher probability

it is always accepted, if to a point with lower probability it is accepted with a non-zero probability defined by the target density, see, again, Chib and Greenberg (1994) and Metropolis et al. (1953).

If the Markov process is irreducible and the candidate generating density is symmetric, then $q(x, y) > 0$ and $q(x, y) = q(y, x)$ for all x and y and

$$\alpha(x, y) = \min\left\{\frac{\pi(y)}{\pi(x)}, 1\right\}.$$

Random Walk Chains

If the target density is of dimension k and $\boldsymbol{E} = R^k$ where $\boldsymbol{E}$ is the state space of the target density $f(\cdot)$, then the candidate $\boldsymbol{Y}$ is generated by drawing $\boldsymbol{Z}$ independently from $f(\cdot)$ and setting $\boldsymbol{Y} = \boldsymbol{x} + \boldsymbol{z}$, $q(\boldsymbol{x}, \boldsymbol{y}) = f(\boldsymbol{y} - \boldsymbol{x}) = f(\boldsymbol{z})$, and the kernel driving the chain is a random walk.

In this case q is symmetric, the process is irreducible and the acceptance probability is

$$\alpha(\boldsymbol{x}, \boldsymbol{y}) = \min\left\{\frac{\pi(\boldsymbol{y})}{\pi(\boldsymbol{x})}, 1\right\}.$$

Independence Chains

Candidate steps Y can also be chosen from a fixed density f. In this case $q(x, y) = f(y)$ and

$$\alpha(x, y) = \min\left\{\frac{w(y)}{w(x)}, 1\right\}$$

where $w(x) = \pi(x)/f(x)$. The function w is the importance weight function that would be used in importance sampling if observations were generated

from f.

Practice

The easiest algorithm to implement is the random walk. It has the major advantage that the candidate can be generated from a symmetrical distribution, in the sense that $q(x, y) = q(y, x)$, and so the transition probability is

$$\alpha(x, y) = \min\left\{\frac{\pi(y)}{\pi(x)}, 1\right\}$$

In addition, the process will not leave the state space once it has entered it, so $\pi(x) = 0$ is not possible.

3.2.3 Re-sampling

Re-sampling is a technique that allows a sample from one density to be used to generate a sample from a similar or related density. For example a sample from a prior density may be used to obtain a sample from a posterior density.

As a specific example of this approach, let us consider the Rejection method.

Suppose that given a sample from a continuous density $g(\theta)$ a sample is required from a density $h(\theta)$ that is absolutely continuous with respect to $g(\theta)$ [3].

More generally, given a function $f(\theta)$ which is normalisable to a density

$$h(\theta) = \frac{f(\theta)}{\int f(\theta) d\theta}$$

[3] All areas of $h(\theta)$ can be reached from $g(\theta)$, i.e. $g(\theta)$ is an envelope of $h(\theta)$.

and there exists an identifiable constant $M > 0$ such that $\frac{f(\theta)}{g(\theta)} \leq M$, it is possible to obtain a sample from $h(\theta)$ in the following way, (see Smith and Gelfand, 1992).

1. Generate θ from $g(\theta)$.
2. Generate u from uniform $(0, 1)$.
3. If $u \leq \frac{f(\theta)}{Mg(\theta)}$ accept, θ.
4. repeat steps 1-3.

Any accepted θ is then a random variate from $h(\theta)$.

3.3 Posterior densities

After a Bayesian analysis a posterior density exists in some form. If the analysis was carried out analytically, or using quadrature that results in estimates of the parameters of the distribution, there is a mathematical description of a density in a form that can be used to construct a sampler for that density. If the analysis used MCMC methods the posterior exists as a sample from the converged chain.

Whatever form the posterior is obtained in, it is possible to obtain a representative sample from it which can then be used to produce marginal and conditional densities. In order to view or present such summaries some means of handling high dimensional sample data is needed.

The sample can be as large as required so there is no problem with the inherent sparsity of data in high dimensional space, the so called 'curse of

dimensionality', (referred to in, for example, Wand and Jones, 1995, p. 90, though the expression is much older). The sample size required to obtain a particular accuracy in Kernel Density Estimation for the first 10 dimensions is quantified in Silverman (1986, p. 93ff).

Visual examination of any object of greater than two dimensions requires some method of extracting a 2D subset or series of subsets from that object. There are a large number of techniques for doing this which preserve the original character of the object to a greater or lesser extent. The next chapter reviews several of the more common systems along with the Projection Pursuit family. This family seems most appropriate to the work at hand, not least because it introduces little perceived distortion and provides an easily understood view.

Chapter 4

Viewing Multivariate Data

Given some collection of points, drawn in some fashion from a density of high dimensionality, a method of viewing the sample in order to appraise it (and hence the underlying density) is required. Such methods generally produce some one, two or three dimensional condensation of the data or series of views projected from the data.

Many of the available methods are ad-hoc in the extreme and not really useful outside of the context in which they were designed. As an extreme example of this consider Chernoff faces (Chernoff, 1973) or any other method (essentially identical) that uses variation in shape or colour as a means of examining data of more than three dimensions.

Other methods – for example Principle Component Analysis (PCA) – while effective in identifying the sources of variation within a sample, do not provide views of the data that help with obtaining a coherent picture of its internal structure. In contrast, projection methods (sometimes referred to as dimension reduction methods), because of their preservation of data and

their presentation of a sequence of distinct views of that data, have proved to be of more general value for this work (see, for example, Asimov, 1985; Jones and Sibson, 1987). The process of adding a rotation to PCA, either Varimax or Oblimin, should, in theory, separate the variables so that highly correlated variables load a lot on one factor and very little on the others. In reality, this doesn't always happen and the process of moving in a dense way through the Projection Pursuit or Grand Tour achieve the same objective and it is agreed more.

4.1 Projection Methods

Projection methods fall into two categories, those that project the data onto a lower dimensional subspace and those that use some method to compress the data into few dimensions, possibly introducing some distortion to the data in the process. For this work the Projection Pursuit family of algorithms, discussed in section 4.1.1, has proved the most useful and is an example of the first type. An example of the second type is the Cone Plot, discussed in 4.1.2.

4.1.1 The Projection Pursuit Family

This family of methods depends on taking points from the sample space and projecting them onto one or more subspaces. Usually the subspace is one, two or three dimensional.

Assume a data set of size N and dimensionality K, giving a $K \times N$ matrix X, then each sample is a row vector and the coordinate vectors are row vectors.

If a is a K row vector then $a^T X$, where a^T denotes the transpose of a, is an N row vector and is the orthogonal projection of X in the direction of a scaled by the magnitude of a. If H is a function measuring how interesting[1] such a sample is then $H(a^T X)$ is, for fixed X, a function $I(a)$ of the projection direction a. Such a function is a *projection index* . Projection pursuit attempts, by numerical optimisation, to find local optima of $I(a)$. Usually a is constrained to be of unit magnitude (Jones and Sibson, 1987). This lends itself to automated selection of a set of 'best'[2] projections for future, human or automated, examination.

Projection pursuit is ideal for initial automatic examination of a data set, resulting in several "interesting" views for further examination. However, if the examination is intended to be carried out by a human operator, a series of views that changes in a way acceptable to that operator is required. In addition the series of projections should be, in some way, complete. Projection Pursuit may be seen as a series of projections from the data space to a sequence of subspaces, if the subspaces are, in some sense, adjacent to each other and dense in the set of possible subspaces the algorithm is known as the Grand Tour, (Asimov, 1985). The basic structure of the Grand Tour algorithm is as follows:

1. Choose a set of axes that determine a subspace.

2. Project the data onto the subspace.

3. View the projection.

[1]The function measuring interest might be as simple as a measure of deviation from normality, but can essentially be any function that can be defined on the data by the user.

[2]Where 'best' is a subjective judgement of interest as defined in 1

4. Rotate the surface through a predetermined angle following a path through the sample space.

5. Project onto the new subspace and repeat.

This may be seen as analogous, in K dimensions to the well known spin plot in three dimensions (see section 4.2.6).

Steps 1,2,3 and 5 above are fairly easy, the rotation is a well documented matrix operation (see 7.1 for the starting plane and derivation of an N dimensional rotation matrix). All that is required for a successful Grand Tour is some means of ensuring that the projection planes are dense in the set of all possible planes. This is dependant on finding a set of real numbers that are linearly independent over the integers.

Kronecker's Theorem

Kronecker's Theorem (Hardy and Wright, 1954, Theorem 442, p. 373) gives a way of defining a set of numbers that includes a number that is arbitrarily close to any number in a given set. In one dimension

> If ϑ is irrational, α is arbitrary, and N and ϵ are positive, then there are integers n and p such that $n < N$ and
>
> $$|n\vartheta - p - \alpha| < \epsilon \tag{4.1}$$
>
> Hardy and Wright (1954, p. 373, Theorem 438)

Asimov (1985) uses the equivalence that the N dimensional torus may be thought of as a Euclidean space R^N in which all arithmetic is performed modulo 1. Symbolically

$$T^N \simeq \frac{R^N}{2\pi Z^N} \tag{4.2}$$

where Z^N is the integer lattice in R^N. It is well known that dense curves may be found on T^N via the following

Proposition

Let $\{\lambda_1, \ldots, \lambda_N\}$ be a set of real numbers that are linearly independent over the integers. Then the curve $\alpha : R \to T^N$ via $\alpha(t) = (\lambda_1 t, \ldots, \lambda_N t)$ has dense image in T^N. (Note that the coordinates $\lambda_i t$ are interpreted modulo 2π)

Hardy and Wright (1954, p. 381 ff)

Real numbers $u_1, \ldots, u_N$ are said to be *linearly independent over the integers* if the only sequence of integers $\{K_1, \ldots, K_N\}$ for which the equation $\sum_{i=1}^{N} K_i u_i = 0$ holds is with $K_i = 0$ for all i (an obvious example being the logarithms of the prime numbers).

This leads to the Torus method of defining a Grand Tour in Asimov (1985).

Two examples of linearly independent sets of numbers $(z\lambda_K)$ are given in Asimov (1985):

1. *Let $\lambda_K = \sqrt{p_K}$ = the square root of the Kth prime ($p_1 = 2, p_2 = 3, \ldots$). Let z be almost any irrational, positive real.*

2. *Let $\lambda_K = e^K \ mod\ 1$. Let z be almost any irrational, positive real.*

The implementation of the Grand Tour in appendix A is based on the first of these (though changing it to the second is trivial).

The Grand Tour is a particular case of Projection Pursuit (Asimov, 1985; Posse, 1990) such that the set of projection planes is dense in the set of all possible planes. If the path followed by the projection method through the data is chosen carefully, these methods produce a series of changing views of the data that appear coherent to human viewers.

If the intention is to use an automated method to detect features of interest, it is only necessary that the path be one in which the method of selecting projection subspaces chooses a set of spaces that rapidly become dense in the set of all possible choices. In addition it is possible to use these methods for purely machine operation and no visual output, in this case the step to the next projection can be randomised and the most interesting data projection in the neighbourhood searched for. This should produce a set of interesting projections for further analysis (human or automated) if the concept of "interesting" can be quantified. For example, in many cases, interesting might be equivalent to "non-normal" or "multi-modal". As a trivial example the data might be the output from some MCMC simulation, in which case, the definition of interesting might be some projection that deviates from the expected density.

Projection methods are easily implemented and, given an MCMC implementation, lend themselves to the viewing of functions. They are easy to combine with density estimation to give a contour view of the underlying density.

Figures 4.1 and 4.2 are examples of a small number of views from two Grand Tours. Figure 4.1 is a sample from a 24 dimensional posterior density, being cell probabilities of a $2\times2\times6$ contingency table from a clinical trial, where the parameters are some function of the cell probabilities. The only interesting feature here is the normality of the views. Figure 4.2 is a sample of the

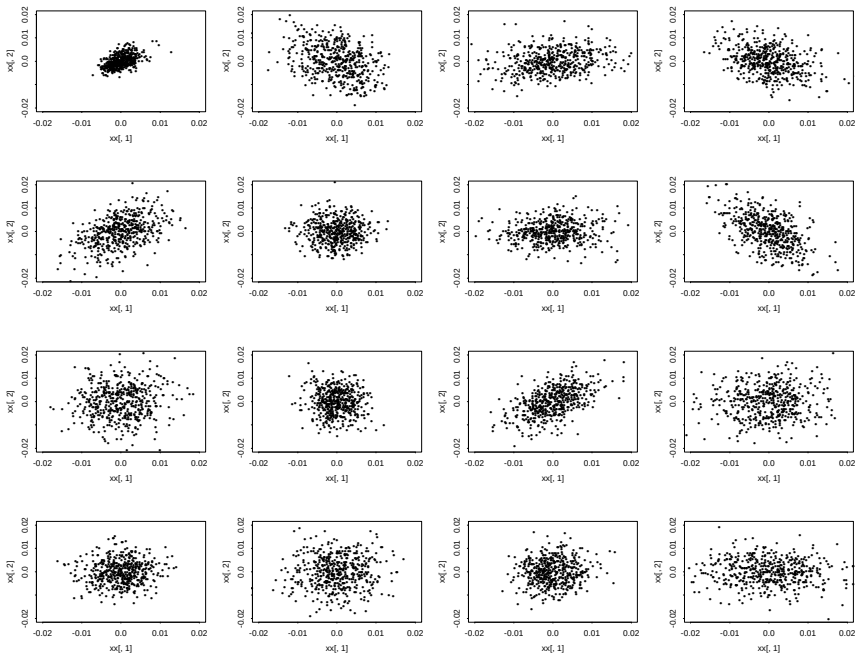


Figure 4.1: *Sample of a Grand Tour - projecting on a two dimensional target - applied to the clinical trial data. The Axes are effectively the axes of the target and as such represent values of the projected data, i.e. they are relative to the step of the tour. The views here are produced from the first 16 steps of a tour choosing a large step size to give a range of differing views rather than a set of close, and therefore similar images (using a seed of* $\sqrt{5}$ *in the algorithm given in Section 4.1.1)*

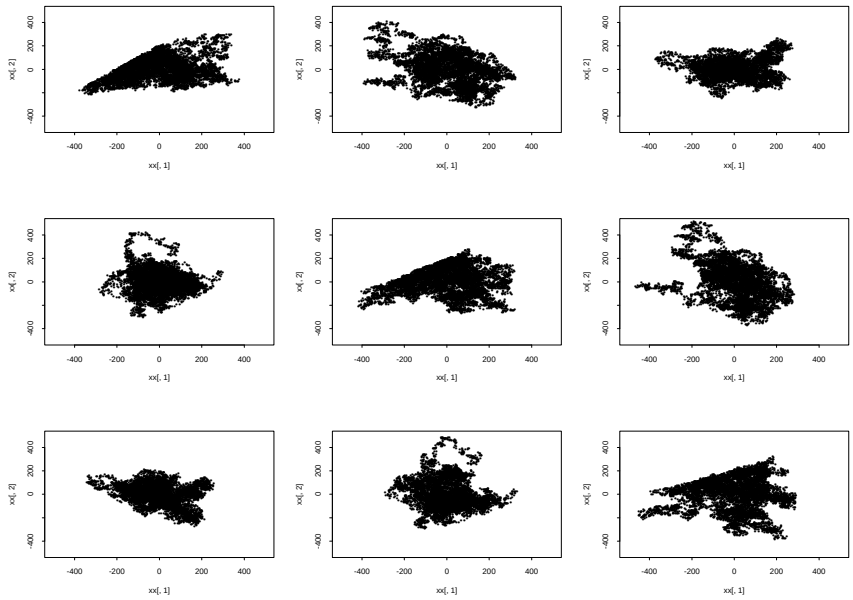


Figure 4.2: *Sample of a Grand Tour - projecting on a two dimensional target. The data is the result of applying the Metropolis-Hastings algorithm to the Archaeological problem. The Axes are effectively the axes of the target and as such represent values of the projected data, i.e. they are relative to the step of the tour. The views here are produced from the first 9 steps of a tour choosing a large step size to give a range of differing views rather than a set of close, and therefore similar, images (using a seed of* $\sqrt{5}$ *in the algorithm given in Section 4.1.1).*

posterior density given by a MCMC approach to the archaeological problem specified in Naylor and Smith (1988) and discussed in 6.2. Note the "holes" in the projection, indicative of sampling problems, and the straight edge indicating a boundary to the sample space. In effect the Metropolis Hastings approach designed for the problem had several limitations preventing access to parts of the sample space, this is clearly shown in the figure.

4.1.2 Cone Plots

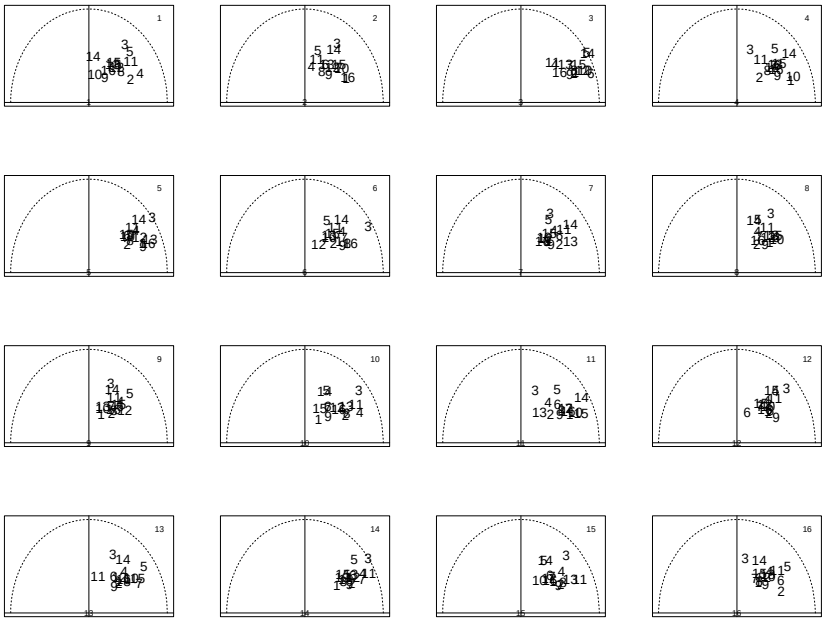


Figure 4.3: *Sample of a Cone plot - applied to the clinical trial data, first 16 values. The x axis is defined by two points from the data and the other points are plotted relative to that.*

Cone plots, a term introduced by Dawkins (1993, 1995) are produced in the following manner:

1. Call the data set P and $P_i \in P$ is a single point of the set.
2. Choose two points in P, denoted by V and A.
3. Compute the Euclidean distance R_i from V to all the points in the data set.
4. Compute all the angles $\theta_i = AVP_i$.

5. Plot the points $(R_i \cos\theta_i, R_i \sin\theta_i)$.

The Cone plot has application in the examination of data sets with structure that is internal (i.e. structure that cannot be seen without removing part of the data). For example, a data set that consists of an n dimensional hollow sphere is easily examined with a Cone plot. This type of data set was proposed by Wegman (1990) as a test of multi-dimensional EDA techniques. However, the limitation of the method to small numbers of points limits the possibilities of density estimation. In addition the Cone plot introduces substantial distortion to the data before viewing due to the 'folding' effect. As interest here is in the advantages derived from the availability of large samples, the subsequent limitation of the "curse of dimensionality", and the estimation of the underlying density, the Cone plot is of little use in this work.

4.2 Other Techniques

The following are graphical techniques that have been examined for their potential as visualisations of multi-dimensional data sets, however for reasons discussed in subsequent sections they have not proved suitable for the current work.

4.2.1 Andrews Plots

The Andrews plot is a system that represents each point of a data set as a separate object, see (see Andrews, 1972). In this case each point in the data set $X = (x_1, x_2, \ldots, x_n)$ is mapped onto a function of the form:

Tooth type	1	2	3	4	5	6
A. British	-5.35	-7.07	-9.37	-4.28	-2.15	-2.93
B. Australian aboriginal	3.93	-6.04	-8.87	-2.16	-0.5	-1.09
C. Gorilla male	3.12	6.66	6.28	4.96	4.13	4.60
D. Gorilla female	1.45	1.73	4.82	3.96	3.35	3.63
E. Orang-outang male	2.83	5.10	5.11	2.72	1.21	1.49
F. Orang-outang female	1.49	1.63	3.61	1.29	-0.171	0.0503
G. Chimpanzee male	0.38	3.82	3.46	-1.65	-2.32	-1.92
H. Chimpanzee female	0.01	0.231	3.05	-2.25	-2.65	-2.15
I. Paranthropus crassidens	-4.52	-6.49	-7.79	3.45	4.91	3.72
J. Pithecanthropus pekinensis	-1.81	-2.94	-6.63	-0.369	-1.32	1.09

Table 4.1: *Largest canonical variable for 6 teeth Andrews (1972, Table 2).*

$$f_X(t) = x_1/\sqrt{2} + x_2 \sin(t) + x_3 \cos(t) + x_4 \sin(2t) + x_5 \cos(2t) + \ldots \quad (4.3)$$

and the function plotted on the range $-\pi \leq w \leq \pi$.

There is a possible extension to this type of plot to make it useful for the dynamic examination of the output of a simulation. The output is passed to a file and the file monitored by a program that dynamically uses an Andrews plot to display the last n values from the simulation. In this way the user might gain some indication of convergence of the simulation. As a static viewing device the plot is too dependent on the ordering of the data to make it useful for a large number of data points. It also places a heavy demand on

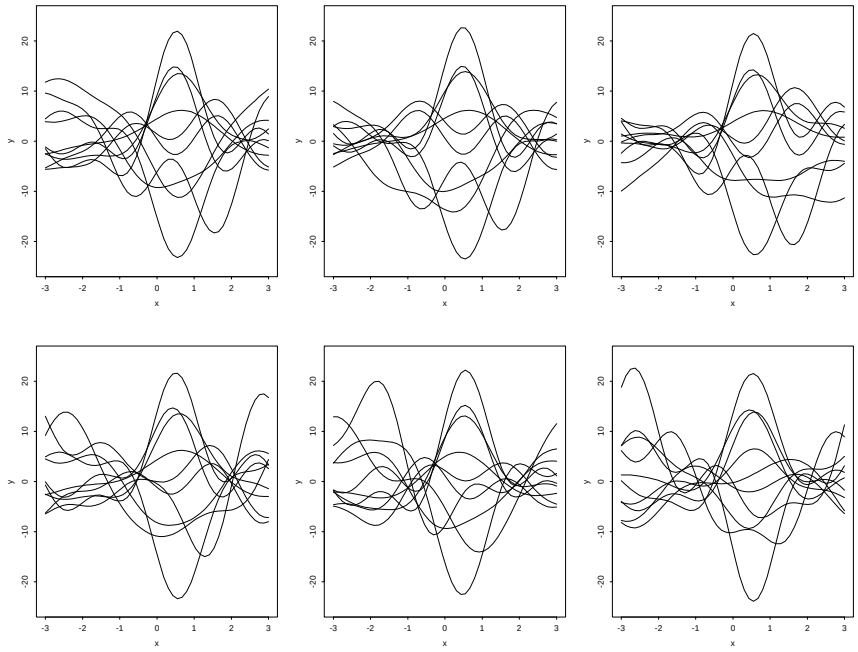


Figure 4.4: *Six Andrews plots of the teeth data, the plots each have two columns of the data swapped showing the influence of ordering. The x axis is the value of t and so has range* $[-\pi, \pi]$ *the y axis is the value* $f_X(t)$ *where* $f_X(t)$ *is defined in Equation 4.3*

the user in terms of concentration and use of time. See the example of tooth data in Figure 4.4 and Table 4.1 as an illustration.

4.2.2 Star Diagrams

Star diagrams are a representation of an n dimensional data set by the addition of a series of spikes to a plotted point. The plotted point represents two of the dimensions, and the length of each spike one of the remaining dimensions (Fienberg, 1979).

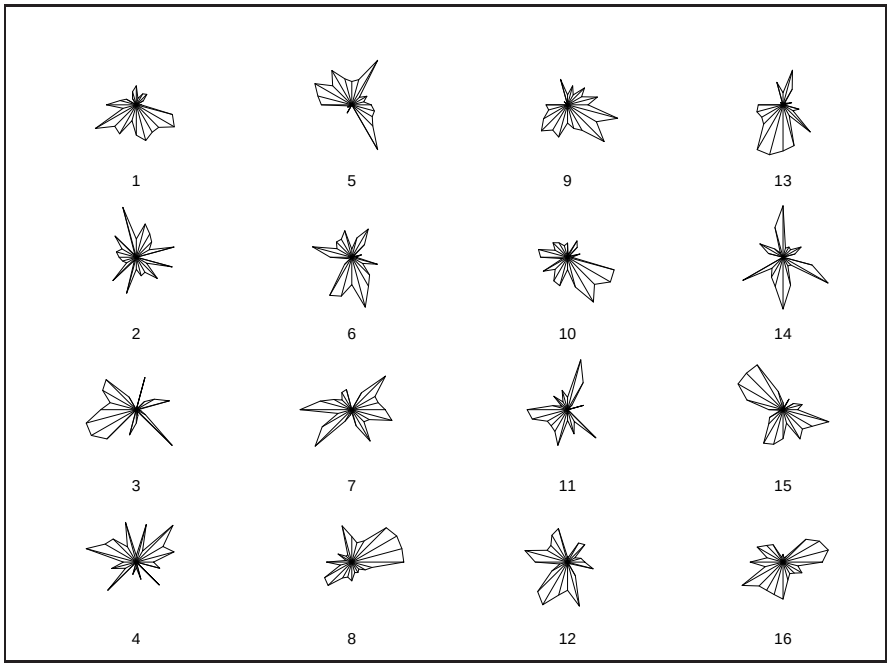


Figure 4.5: *Sample of a Star plot - applied to the clinical trial data, first 16 points. The axis here are meaningless, magnitude being represented by the line length within the star.*

The Star diagram represents a single point in n-dimensions, a data set with several points has a Star plot for each point. The practical limit of usability is in the region of 20 points, so for the type of plot required here, with sometimes many hundreds of points, the system is obviously unusable. However, for small collections of data of high dimensionality, the Star plot allows an easy, visual, comparison of the points in the data set. Figure 4.5 shows 16 data points of dimension 24.

```
> require(TeachingDemos)
> set.seed(17)
> faces(matrix(sample(1:1000,128,),16,8),main="random faces")
```

Figure 4.6: *R code for the Chernoff faces example.*

4.2.3 Chernoff Faces

Chernoff faces are a system for Exploratory Data Analysis (EDA) of multivariate data that use the idea of varying features of an image according to the values of a high dimensional object. They are typical of such systems, being difficult to interpret, impractical, unusable for large (> 20) numbers of points and limited in the number of dimensions of the data by the number of distinct features in the image. The 'facial features' each correspond to one dimension of the data and are varied according to the data value. See, for example, the 'faces' command in the TeachingDemos package of R. In R, having loaded the package using the `install.packages()` command, the code in Figure 4.6 will produce a page of "faces".

For details see (Chernoff, 1973).

A Chernoff face represents a single point in n-dimensions, a data set with several points has a face for each point. For the type of plot required here, with sometimes many hundreds of points and in which there is no interest in identifying individual points, the system is inappropriate. For comparisons of small sets of high dimensional data the faces can be useful in a similar way to a Star plot.

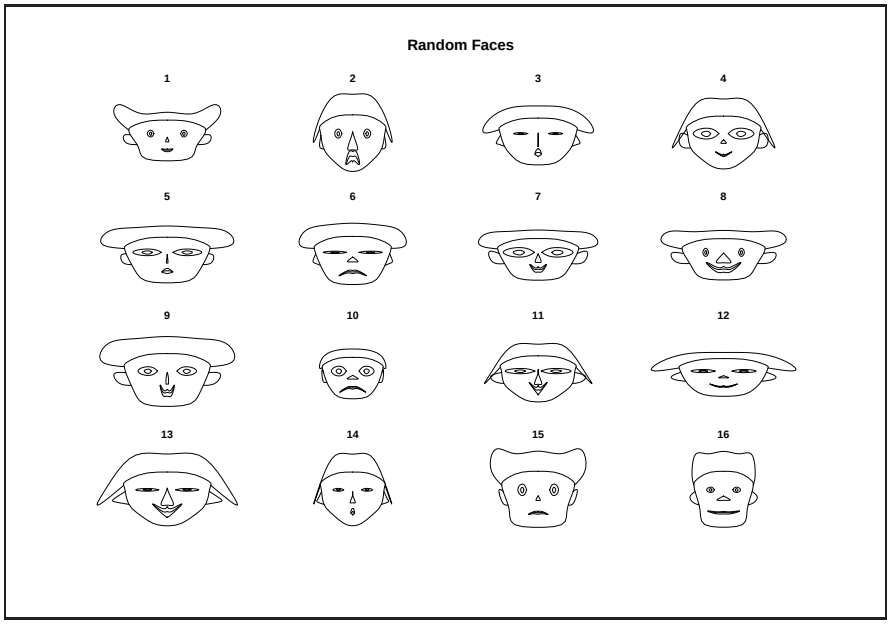


Figure 4.7: *Sample of Chernoff faces - applied to 16 random data points, generated as in Figure 4.6. Each face represents one data point.*

4.2.4 Scatterplots

In its crude form the Scatterplot consists of a simple (x, y) plot of the values of two variables. Scatterplots are often used to look for features, such as linearity, in low dimensional data.

Scatterplot matrices are an extension of this. For n dimensional data a Scatterplot matrix is a 'matrix' of plots in which the plot of the ith variable against the jth variable, $i \neq j$, is the (i, j) entry of the 'matrix'. The basic idea is a generalisation of a draughtsman's plot in which the top, front and side views of an object are shown (Cleveland and McGill, 1984). A valuable extension to this is the technique of *brushing* in which the same point (or group of points) is highlighted on all the 2D plots, (see Weihs, 1993; Becker and Cleveland, 1987).

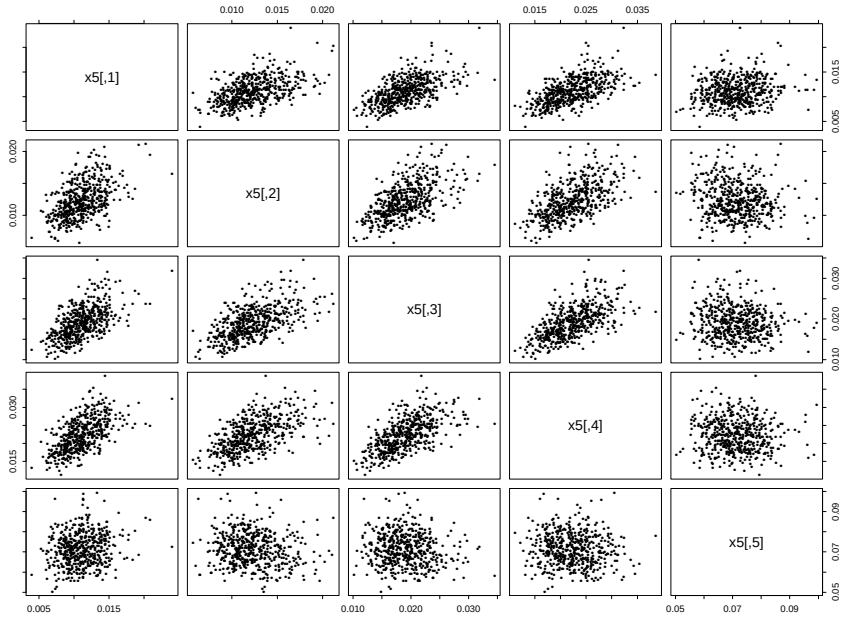


Figure 4.8: *Sample of a Scatterplot matrix applied to the clinical trial data, first 5 dimensions. Axes represent the variable values.*

The plot shown in Figure 4.8 is a five dimensional data set. The main problems here are a) the system will become very unwieldy for any system of dimensionality much larger than five and, b) the views shown are not views of the whole data set (but are of sets of two dimensions extracted from the whole).

4.2.5 Parallel Coordinate Representation

Parallel Coordinate Representation was first proposed by Wegman (1990) and is a variation on methods which take each point as an independent entity such as the Andrews plot.

A parallel coordinate diagram is produced as follows:

1. draw the n axes in parallel.

2. for each point in the data set $X = (x_1, x_2, \ldots, x_n)$ plot each component x_j on axes j.

3. join the points comprising X with a line.

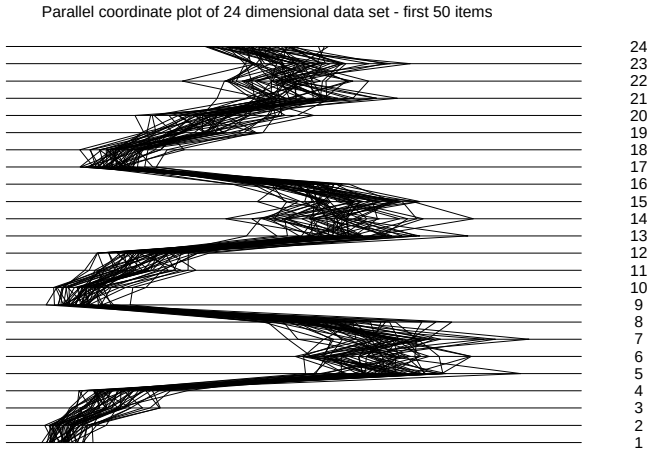


Figure 4.9: *Sample of a Parallel coordinate plot - applied to the clinical trial data, first 50 points. Axes are magnitude horizontally and point number vertically.*

Like the Andrews plot, the Parallel Coordinate Diagram is highly dependent on the ordering of the data. In addition, with a large number of points, the plot soon looks like a single black blob. The difficulty in interpreting such a plot is illustrated in Figure 4.9. Again it is useful for small data sets, and especially so for comparing identically ordered data (see Fiorini and Inselberg, 1989).

4.2.6 Spin Plots

Spin plots are a scatter plot that consists of three dimensions of the data set projected onto two dimensions with the ability to interactively change the projection angle. This gives a simple system for examining and exploring low dimensional data. When combined with principal component analysis (see section 4.2.7), this may be particularly powerful, as three dimensions may hold, for example, 90% of the variation of some data sets. However, in reducing the viewed dimensions to three, while much of the variation may be displayed, some is not, and this can result in missing important subtleties of the data.

The Spin plot can be used to look at data with more than three dimensions in much the same way that the Scatter plot matrix can. However, the addition of the spin controls to the matrix of plots makes them hard to understand and adds little useful information to the analysis. As a tool for viewing three dimensions of data it is excellent. It can be further improved by the addition of brushing and combining it with Principle Component Analysis extends its use to objects of higher dimensionality.

4.2.7 Principal Component Analysis

Principal component analysis (PCA) transforms a number of possibly correlated variables into a smaller number of uncorrelated variables called principal components. The variables are sorted into an order based on the contribution they make to explaining the variance in the data. A sufficient number of variables are preserved to explain the required amount of variability and the rest discarded. For example if the data is 10 dimensional and the first

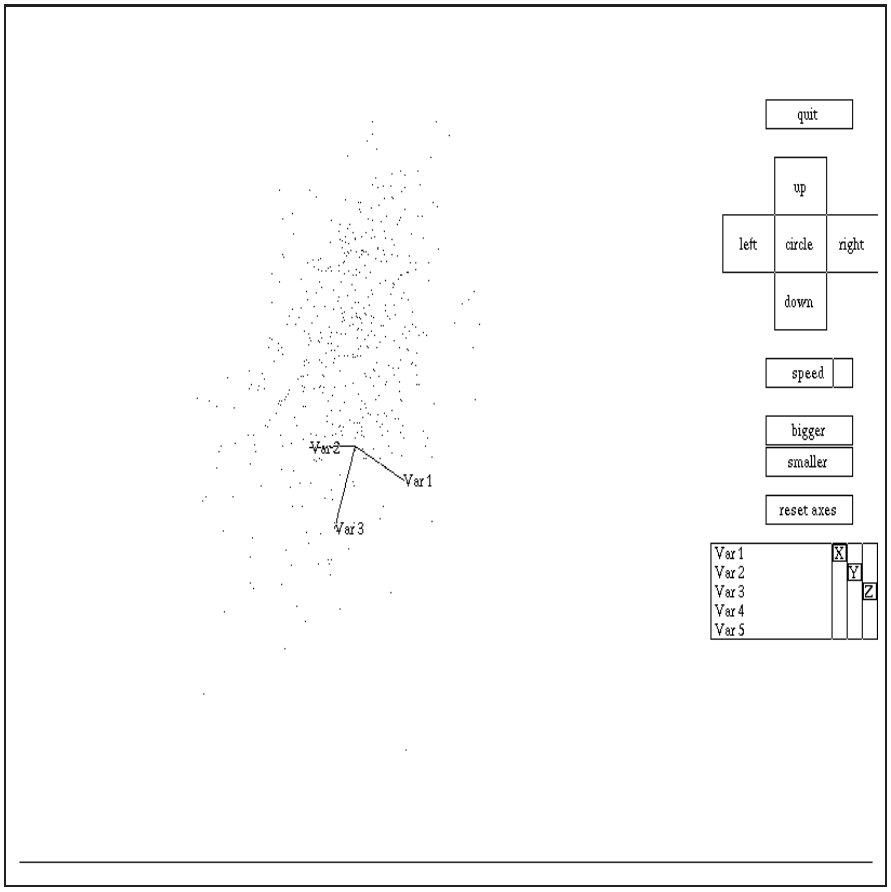
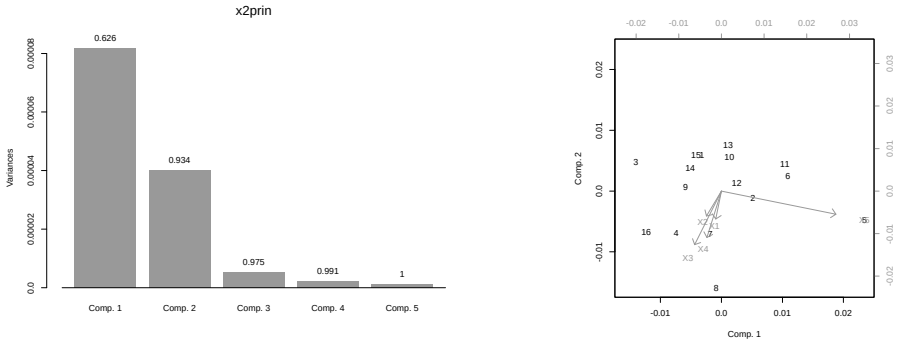


Figure 4.10: *Sample of a Spin Plot applied to the clinical trial data, first 5 dimensions, first three dimensions selected. Axes are projected magnitude.*

three variables explain 90% of the variability, they might be sufficient for the required analysis.

Figure 4.11 shows two uses of graphical techniques with PCA. The data are limited to 5 dimensions to reduce congestion. 93% of the variation in this data was explained by the first two components.

There are several methods of deriving the ordering for the components and their contribution to variability, one of the most common being the covariance method:



(a) Scree plot

(b) Biplot

Figure 4.11: *Two views of a PCA applied to the first 5 dimensions of the clinical trial data. Axes relate to the variance attributed to the components. The figure was generated by the standard R functions* `screeplot` *and* `biplot`*. (A more complete discussion of these techniques can be obtained in Mardia et al., 1979; Venables and Ripley, 2002).*

1. Organise the data set

   Suppose you have data comprising a set of observations of M variables, arrange the data as a set of N data vectors $\mathbf{x}_1 \ldots \mathbf{x}_N$ with each $\mathbf{x}_n$ representing a single grouped observation of the M variables.

   - Write $\mathbf{x}_1 \ldots \mathbf{x}_N$ as column vectors, each of which has M rows.
   - Place the column vectors into a single matrix $\mathbf{X}$ of dimensions $M \times N$.

2. Calculate the empirical mean

   - Find the empirical mean vector u by calculating the mean along each dimension $m = 1 \ldots M$.

$$[m] = \frac{1}{N} \sum_{n=1}^{N} X[m, n] \tag{4.4}$$

3. Calculate the deviations from the mean.

   Mean subtraction is an integral part of the solution towards finding a principal component basis that minimizes the mean square error of approximating the data.

   - Subtract the empirical mean vector u from each column of the data matrix $\mathbf{X}$ forming the matrix $\mathbf{B}$.

4. Find the covariance matrix

   Find the $M \times M$ empirical covariance matrix $\mathbf{C}$ from the outer product of matrix $\mathbf{B}$ with itself:

$$\mathbf{C} = \mathbb{E}\left[\mathbf{B} \otimes \mathbf{B}\right] = \mathbb{E}\left[\mathbf{B} \cdot \mathbf{B}^*\right] = \frac{1}{N}\mathbf{B} \cdot \mathbf{B}^* \tag{4.5}$$

   where

   $\mathbb{E}$ is the expected value operator, $\otimes$ is the outer product operator, and $*$ is the conjugate transpose operator. Note that if B consists entirely of real numbers, which is the case in many applications, the "conjugate transpose" is the same as the regular transpose.

5. Find the eigenvectors and eigenvalues of the covariance matrix

   - Compute the matrix $\mathbf{V}$ of eigenvectors which diagonalises the covariance matrix $\mathbf{C}$:

$$\mathbf{V}^{-1}\mathbf{C}\mathbf{V} = \mathbf{D} \tag{4.6}$$

     where $\mathbf{D}$ is the diagonal matrix of eigenvalues of $\mathbf{C}$. This step will typically involve the use of a computer-based algorithm for computing eigenvectors and eigenvalues.

- Matrix $\mathbf{D}$ will take the form of an $M \times M$ diagonal matrix, where

$$\mathbf{D}[p,q] = \lambda_m \qquad \text{for} \qquad p = q = m \tag{4.7}$$

  is the mth eigenvalue of the covariance matrix $\mathbf{C}$, and

$$\mathbf{D}[p,q] = 0 \qquad \text{for} \qquad p \neq q \tag{4.8}$$

- Matrix $\mathbf{V}$, also of dimension $M \times M$, contains M column vectors, each of length M, which represent the M eigenvectors of the covariance matrix $\mathbf{C}$.
- The eigenvalues and eigenvectors are ordered and paired. The mth eigenvalue corresponds to the mth eigenvector.

6. Rearrange the eigenvectors and eigenvalues

   Sort the columns of the eigenvector matrix V and eigenvalue matrix D in order of decreasing eigenvalue, making sure to maintain the correct pairings between the columns in each matrix.

7. Compute the cumulative energy content for each eigenvector

   The eigenvalues represent the distribution of the source data's energy among each of the eigenvectors, where the eigenvectors form a basis for the data. The cumulative energy content g for the mth eigenvector is the sum of the energy content across all of the eigenvectors $1 \ldots m$:

$$g[m] = \sum_{q=1}^{m} \mathbf{D}[p,q] \qquad \text{for} \qquad p = q \qquad \text{and} \qquad m = 1 \ldots M \tag{4.9}$$

8. Select a subset of the eigenvectors as basis vectors

   Save the first L columns of $\mathbf{V}$ as the $M \times L$ matrix $\mathbf{W}$:

   Use the vector $\mathbf{g}$ as a guide in choosing an appropriate value for L. The goal is to choose as small a value of L as possible while achieving the required value of $\mathbf{g}[\cdot]$ on a percentage basis. For example, you may want to choose L so that the cumulative energy $g[\cdot]$ is above a certain threshold, for example 90 percent. In this case, choose the smallest value of L such that

$$g[m = L] \geq 90\% \tag{4.10}$$

9. Convert the source data to z-scores

   Create an $M \times 1$ empirical standard deviation vector $\mathbf{s}$ from the square root of each element along the main diagonal of the covariance matrix $\mathbf{C}$:

$$\mathbf{s} = \{s[m]\} = \sqrt{C[p,q]} \qquad \text{for } p = q = m = 1 \ldots M \tag{4.11}$$

   Calculate the $M \times N$ z-score matrix:

$$\mathbf{Z} = \frac{\mathbf{B}}{\mathbf{s} \cdot \mathbf{h}} (\text{divide element} - \text{by} - \text{element}) \tag{4.12}$$

   Note: While this step is useful for various applications as it normalises the data set with respect to its variance, it is not an integral part of PCA!

10. Project the z-scores of the data onto the new basis

    The projected vectors are the columns of the matrix

$$\mathbf{Y} = \mathbf{W}^* \cdot \mathbf{Z} = \mathbb{KLT}\{\mathbf{X}\} \tag{4.13}$$

The columns of matrix $\mathbf{Y}$ represent the Karhunen-Loeve transforms (KLT) of the data vectors in the columns of matrix $\mathbf{X}$.

Although PCA is "perfectly general" and makes no basic assumptions, this method for deriving the PCA from a set of points using the variance-covariance matrix of the points depends on an underlying assumption of normality. Also the measure of how much variation can be captured by the leading PCA term assumes normality.

4.2.8 Comments

All of the methods in section 4.2, have one or more of the following problems making their use for the current project not very practical.

1. They cannot deal with many more than 3 dimensions. For example Chernoff faces have about 10 usable features and with more than 5 dimensions they become very confusing to interpret.

2. The "Curse of Dimensionality". In examining a data set obtained by some experiment, there is a limited quantity of data. If the data has 2 dimensions, 300 points may be a sufficient data set, if it is of 300 dimensions, 300 points is very sparse (Silverman, 1986).

3. Computational difficulties. Some of the methods require a large number of computations in a short time to be viable.

4. Interpretation difficulties. Distinguishing detail in two similar plots where detail is obscured by the necessity of fitting a series of plots into a limited area, or where there are many plots to compare, is difficult.

5. Usability, varies between the methods with some requiring operator training and some being practically useless except in special cases.

Table 4.2 attempts to summarise some of the differences between various display methods.

In summary, many of the methods examined have a lot of merit in data analysis. Some, for example Chernoff Faces, have little use except in very limited situations. The choice of which to use here comes down to several factors:

1. Retention of data. Ideally there would be no loss of data, or, even, reduction in available information from the data.

2. Ease of implementation. As a system of exploratory analysis ease of use and the availability of a simple version of the algorithm is desirable.

3. Maturity. The method chosen should be well established in a form that requires as little specialist interpretation as possible.

The Grand Tour fits all of these, there are variations and improvemants to the basic system but these are not considered neccessary or desirable here. It is possible that similar use could be made of other systems, for example PCA with rotation, however, the Grand Tour is the one chosen here. In terms of the software produced, discussed in Chapter 7 and Appendices A and B, the requirement was for simple to use software for EDA. Therefore

	Max Dim	Computation	Interpretation	Usability
Projection	$\propto N$	low	simple	good
Cone plot	$\approx N$	low	difficult[a]	good
Andrews plot	≈ 10	medium[b]	difficult[c]	poor[d]
Star Diagram	≈ 30	high	difficult[e]	limited
Chernoff faces	15[f]	high	difficult[g]	limited
Scatterplot	≈ 10[h]	low	simple	good
Parallel coordinate representation	≈ 20	low[i]	difficult[j]	poor
Spin plot	≈ 5[k]	high	simple	good
Principal component analysis	high[l]	variable	simple[m]	good

[a] except for internal features
[b] high if the dynamic extension is used
[c] due to the sensitivity to ordering
[d] skill needed to interpret
[e] but easy for comparison of points
[f] in the R version
[g] but easy for comparison of points
[h] limit is legibility, for n parameters t he scatterplot is an $n \times n$ array.
[i] but can have a dynamic extension like Andrews plots
[j] due to the sensitivity to ordering
[k] allowing for some manual choice of displayed 3
[l] limited by data and computation time
[m] usually used with a display technique, e.g. a spinplot

Table 4.2: *A comparison of methods for viewing high-dimensional data sets.*

the simplest version of the Grand Tour was used. There is a wide range of available literature detailing refinements to both Projection Pursuit and the Grand Tour and their use, but these refinements, see for example Lee and Verleysen (2007), are not helpful here as the intended use of the software is for exploratory analysis of new data (i.e. where little is known about data type or structure).

These problems can be minimised if it is possible to tailor the data to the requirements of the display technique, however, for the present work on visualizing large data sets it is difficult to achieve that tailoring. In addition the system should be usable by observers who are interested in other specialities and who do not necessarily want to assimilate an unfamiliar way of viewing data.

It is also the case that many of these methods are good at highlighting "odd" values in the sample, or tracking individual data items or groups. In the present context there is no real meaning to an individual data point — the sample is truly anonymous and has no outliers such as might be expected in a measured set of data. Consequently, the remainder of this thesis assumes some projection pursuit method (e.g. the Grand Tour) is to be used for dimension reduction as there is no need to limit the data in any way.

Chapter 5

Univariate density estimation

As stated, Bayesian analysis typically results in either a mathematical expression for the posterior density or, given MCMC treatment, a sample from the posterior. In the first case it is usually a trivial matter to obtain a sample from that density. The sample, in either case, will have dimensionality equal to that of the parameter space of the posterior, typically this will be significantly higher than 3. As discussed in Chapter 4 some method can then be used to obtain reduced dimensional "views" of the sample, for the purposes of this thesis this is the Projection Pursuit family of tools and results in 1D or 2D projections from the sample of interest.

Having obtained a low dimensional projection the problem is how to represent it. Ideally that representation should consist of some form of estimation of the underlying density. Initially, consider a 1 dimensional projection. This may be a marginal density for some univariate functions of the parameter vector $\boldsymbol{\theta}$ or a predictive density for some future observation y. In either case the presentation or density estimation problem is the same.

In this chapter a review of frequentist density estimation techniques is presented. The issues relating to the use of these methods for presenting marginal posterior densities are also discussed and a number of concerns and problems identified. Kernel Density Estimation (KDE) would normally be used to attempt to recover the underlying density from a data sample, here it is intended to apply the same technique to the output obtained as a projection from a sample from a posterior density.

5.1 Density Estimation

Given some experimental data $\boldsymbol{x} = \{x_1, x_2, \ldots, x_n\}$, assumed to be a random sample from some distribution (i.e. values of $g(\boldsymbol{\theta})$ or y), the problem is estimating the probability density function (pdf), $f_X(\cdot)$. This density estimation problem has had much attention in recent papers. Books by Scott (1992), Silverman (1986) or Wand and Jones (1995) provide a good overview of the field.

5.1.1 The Histogram

The simplest density estimator is the histogram. This is formed by first dividing the real line into intervals called *bins*. In the case of bins of equal width h the histogram is a step function which estimates the density at a point x by the function

$$\hat{f}(x) = \frac{1}{nh}(\text{no. of } x_i \text{ in the same bin as } x) \tag{5.1}$$

where n is the sample size. However, in constructing the histogram it is

necessary to choose the origin and the bin width h. Both of these make a significant difference to the performance of the method, Figure 5.1 shows the effects of different start points and number of bins when plotting some of the data set "Observations of eruptions of the Old Faithful geyser in Yellowstone National Park, USA", from Weisberg (1980). The data consists of the eruption lengths (in minutes) of 107 eruptions of the Old Faithful geyser and is analysed using several existing KDE methods in Silverman (1986). For further examples see Scott (1992).

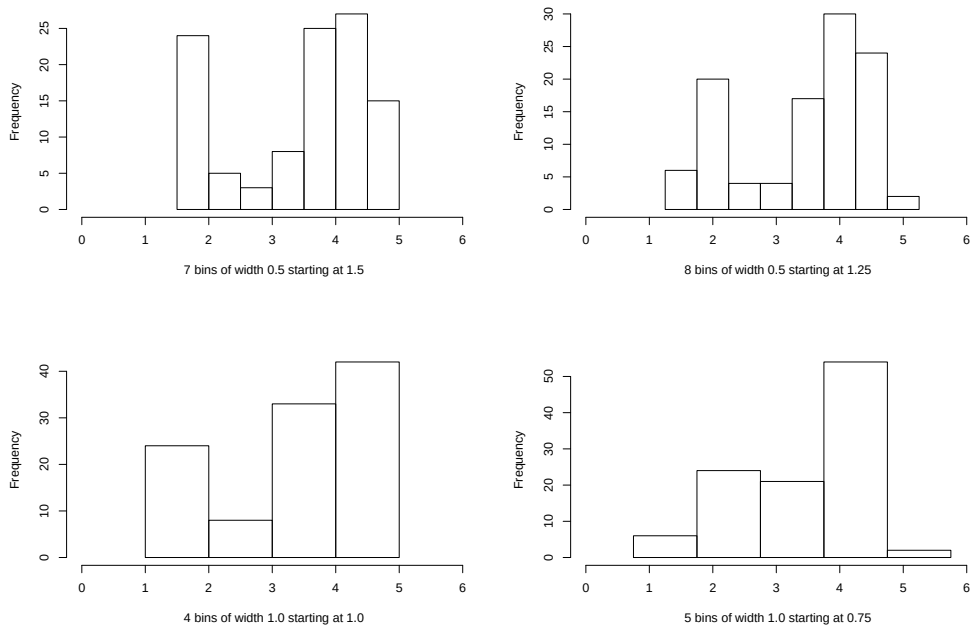


Figure 5.1: *Four different histograms of the Old Faithful data. Axes are value (x) and frequency (y).*

The histogram can be generalised by allowing the bin widths to vary (Silverman, 1986; Scott and Terrell, 1987), such that

$$\hat{f}(x) = \frac{1}{n} \frac{(\text{no. of } x_i \text{ in the same bin as } x)}{(\text{width of bin containing } x)} \tag{5.2}$$

The bin width is often called the smoothing parameter as it specifies the amount of smoothing being applied to the data – a small value giving a more jagged appearance. The histogram is an excellent tool for Exploratory Data Analysis (EDA)(Tukey, 1977), however, it is of limited use for the application considered here. The histogram has the unfortunate feature that it estimates all densities as step functions. As the densities that result from Bayesian analysis are usually continuous, and part of the interest in KDE is to obtain some estimate of their smoothness, a continuous estimate of the density function is desirable. In addition the histogram's sensitivity to bin position and width means that, in order to obtain a satisfactory representation, multiple histograms, or even some form of averaged histogram (Scott, 1992), are needed.

5.1.2 Polygon Methods

By connecting the centre point of the top of each bin with a continuous line a *frequency polygon* is obtained. Scott (1983, 1985) considered the problem of choosing amongst the collection of multivariate frequency polygons each with the same smoothing parameter but differing bin origins. Rather than choosing the smoothest[1] curve or surface, he proposed averaging a series of such polygons. As the average of piecewise linear curves is also piecewise

[1] A smooth curve here is one with no rapid changes or oscillations, so the smoothest is the one with the least change in slope over its length.

linear[2] the result appears to be a frequency polygon.

A similar device that is just as general is the Averaged Shifted Histogram (ASH) in which several shifted histograms are averaged[3]. Again since the average of piecewise constant functions is itself a piecewise constant function, the resulting ASH appears to be a frequency polygon as well. In practice the ASH is made continuous using some form of linear interpolation. ASH is a useful tool for density estimation, however these methods are complex and do not yield the smooth style of density estimate preferred for the application here.

5.1.3 The Naive Estimator

For a random variable X, the probability density function is defined as a limit of a probability for an interval, as the interval width reduces to zero. That is

$$f(x) = \lim_{h \to 0} \frac{1}{2h} \Pr(x - h < X < x + h). \tag{5.3}$$

This allows estimation of $P(x-h < X < x+h)$, for any h, by the proportion of the sample falling in the interval $(x - h, x + h)$. Hence, for a small h, $\frac{1}{2h} \Pr(x - h < X < x + h)$ would be an approximation to $f(x)$. Replacing the probability with a relative frequency gives the naive estimator

[2]Write a linear segment as $y_i = mx_i + c$, so summing several linear segments, at the point (x_i, y_i) gives $y_i = \Sigma_{j=1}^{n}(\boldsymbol{m}_j x_i + \boldsymbol{c}_j) = \Sigma_{j=1}^{n}\boldsymbol{m}_j x_i + \Sigma_{j=1}^{n}\boldsymbol{c}_j = x_i \Sigma_{j=1}^{n}\boldsymbol{m}_j + \Sigma_{j=1}^{n}\boldsymbol{c}_j$ which is obviously a linear segment.

[3]A shifted histogram being one with the same data and bin width but with a different starting point on the x axis.

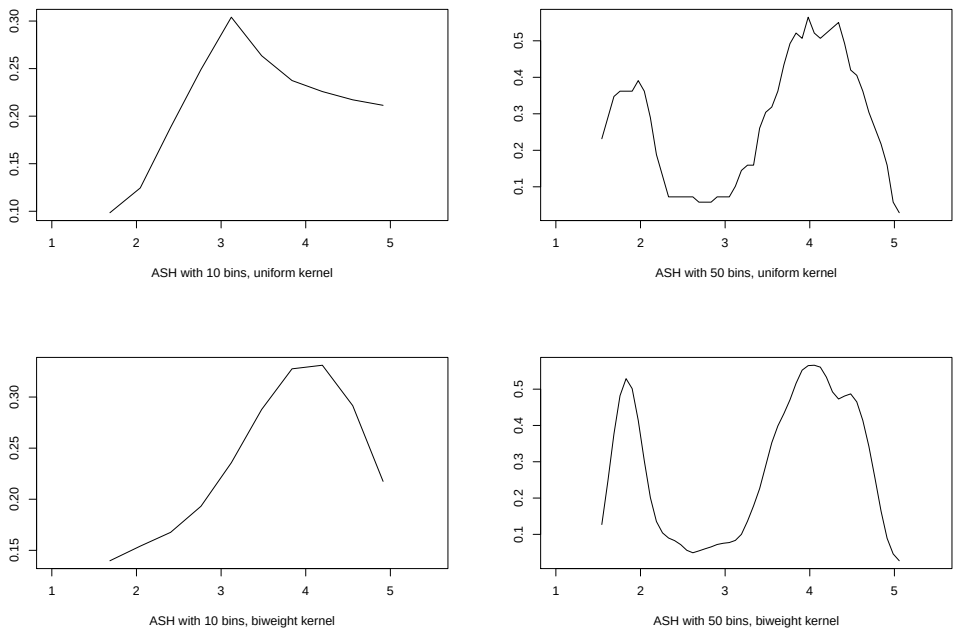


Figure 5.2: *ASH of the Old Faithful data with different bin counts and kernels. The vertical axis here is a normalised score.*

$$\hat{f}(x) = \frac{1}{2nh}(\text{ no. of } x_1, \ldots x_n \text{ falling in } x \pm h) \tag{5.4}$$

see Silverman (1986).

Defining the weight function

$$w(x) = \begin{cases} \frac{1}{2} & \text{if } |x| < 1 \\ 0 & \text{otherwise} \end{cases} \tag{5.5}$$

allows the naive estimator to be written

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} w\left(\frac{x - x_i}{h}\right). \tag{5.6}$$

Like the histogram the naive estimator is a step function not a continuous function. For this reason it is not wholly satisfactory either as a density estimate or for presentation. For the recovery of continuous densities a smooth estimator that operates in a way that is consistent with its use in a Grand Tour is required (see section 4.1.1) (i.e. it should be fast enough that the flow of projections is not disrupted). KDE is a generalization of the naive estimator which replaces the weight function with a kernel function. If this kernel function is a smooth, continuous, probability density function then the density estimate will be a smooth, continuous, probability density function too, obtaining the necessary smoothness characteristics.

5.2 The Kernel Density Estimator

5.2.1 Introduction

If the weight function in (5.6) is replaced by a *kernel function* K which satisfies

$$\int_{-\infty}^{\infty} K(x)dx = 1 \tag{5.7}$$

The *kernel density estimator* is defined as (see for example Silverman, 1986, p. 15)

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right). \tag{5.8}$$

where h, is the window width or *bandwidth* of the estimator. Note that the estimate smoothness depends on the bandwidth. If h is small then the

estimate will consist of spikes centred on the x_i if h is large then the estimated density tends to the uniform and all detail is obscured.

The estimate obtained is continuous if K is continuous and so may avoid the problems associated with the naive estimator or the histogram. A real disadvantage, in this context, of both the naive estimator and the histogram is that they both exhibit a lack of continuity.

The estimated density is a sum of n functions, where n is the number of items of data. This means that it has the same properties as the kernel function – if K is a probability density function[4] then so too is $\hat{f}$.

It should also be noted that the naive estimator, as defined above, is a KDE with the non-continuous Kernel

$$K_n(x) = \begin{cases} \frac{1}{2} & \text{if } -1 < x < 1 \\ 0 & \text{otherwise} \end{cases} \tag{5.9}$$

However, this kernel gives an estimate with discontinuities similar to, if not so visually obvious as, the histogram.

It is usual that $K(x) \geq 0, \quad \forall x$, however there are arguments for sometimes using kernels which take negative values (see Silverman (1986) section 3.6). This can lead to problems and, as the potential advantages are not large, the kernel functions used in the present work are everywhere non-negative.

[4]A probability density function is a function f defined on an interval (a, b) and having the following properties.

1. $f(x) \geq 0$ for every x
2. $\int_a^b f(x)dx = 1$

5.2.2 The Adaptive Kernel Estimator

The kernel estimator does suffer from one significant drawback, the need to estimate, or choose the bandwidth. If density is such that different values of h are best for different areas of it, then the best a single value of bandwidth can do is a compromise. For example, when applied to long-tailed distributions the fixed kernel estimator will tend to under-smooth the tail. In order to overcome this, the bandwidth of the estimator may be allowed to vary, for example it is possible to have a bandwidth inversely proportional to the local density obtained from some previous estimate, Silverman (1986, p. 100).

The steps given in Silverman (1986) for arriving at this are

1. Find a *pilot estimate* $\widetilde{f}(\boldsymbol{t})$ that satisfies $\widetilde{f}(x_i) > 0$ for all i

2. Define *local bandwidth factor* λ_i by

$$\lambda_i = \left\{ \frac{\widetilde{f}(x_i)}{g} \right\}^{-\alpha} \tag{5.10}$$

   where g is the geometric mean of the $\widetilde{f}(x_i)$:

$$\log g = \frac{1}{n} \sum log \widetilde{f}(x_i) \tag{5.11}$$

   and α is the *sensitivity parameter,* a number satisfying $0 \leq \alpha \leq 1$.

3. Define the *adaptive kernel estimate* $\hat{f}$ by

$$\hat{f}(t) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{(h\lambda_i)^d} K \left\{ \frac{t - x_i}{h\lambda_i} \right\} \tag{5.12}$$

   where K is the kernel function and h is the bandwidth. As in the ordinary kernel method, K is a symmetric function integrating to unity.

The first step of this requires the use of some density estimator to obtain a pilot estimate, this does not need to be of particular accuracy and can be a simple as a nearest neighbour estimator[5], the fixed BKDE estimate is used here for the very simple reason that it is already part of the Bayes 4 program written to support this work, see appendix C. The sensitivity parameter α controls the sensitivity of the method to variations in the pilot density, setting $\alpha = 0$ gives the fixed bandwidth KDE. Abramson (1982) gives arguments for choosing $\alpha = \frac{1}{2}$ for reasons of minimising the bias of the estimator and finds that:

> Proportionally varying the bandwidths like $f^{-\frac{1}{2}}$ at the contributing readings lowers the bias to a vanishing fraction of the usual value, and makes for performance seen in well-known estimators that force moment conditions on the kernel (and so sacrifice positivity of the curve estimate).
>
> Abramson (1982)

.

The factor g^α in (5.10) means that the bandwidth factors, λ_i are free of the scale of the data. Moreover

$$h\lambda_i = h\left\{\frac{\widetilde{f}(x_i)}{g}\right\}^{-\alpha} \tag{5.13}$$

$$= \frac{h}{g^{-\alpha}}\left\{\widetilde{f}(x_i)\right\}^{-\alpha}, \tag{5.14}$$

[5]For nearest neighbour the density is taken to be inversely proportional to the distance between the data item and the next nearest, by some measure, data item.

so writing

$$\lambda_i \;=\; \left\{\widetilde{f}(x_i)\right\}^{-\alpha}, \tag{5.15}$$

is equivalent to rescaling h by a factor of $\frac{1}{g^{-\alpha}}$.

5.2.3 The Kernel

It is advantageous to employ a probability density function (*pdf*) as the kernel, since then the estimated density is guaranteed to also be a pdf. A common choice of kernel is the Normal pdf, however there is some advantage in using other functions, both those with a higher probability associated with the tail, and those whose contribution to the estimate decreases to zero for points sufficiently remote from the contributor thus allowing the use of a kernel that allows a better fit to the data. For example, in the case of estimating a density that arrives from survival data, there is a distinct advantage in using a kernel with $f(x) = 0$ for $x < 0$.

The Epanechnikov Density

The Epanechnikov Density is an example of a kernel that has zero contribution outside a range but is smooth inside that range, first suggested by Epanechnikov (1969).

$$K_e(x) = \begin{cases} \frac{1}{2}c_d^{-1}(d+2)(1-\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}) & \text{if } \boldsymbol{x}^{\mathrm{T}}\boldsymbol{x} < 1 \\ 0 & \text{otherwise} \end{cases} \tag{5.16}$$

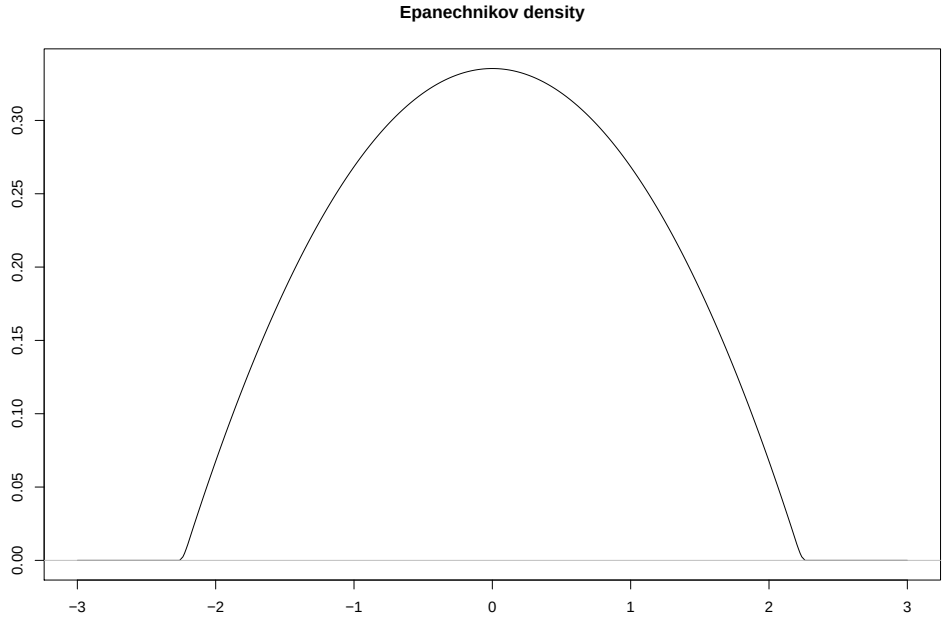


Figure 5.3: *Epanechnikov density, produced trivially by the R command* `plot(density(c(0,0),kernel="epanechnikov"))`. *Axes are value (x) and density (y).*

Where c_d is the volume of the d dimensional unit sphere[6].

So for a 1D KDE, $c_1 = 2$ and

$$K_{e1}(x) = \begin{cases} 3(1-x^2) & \text{if } x^2 < 1 \\ 0 & \text{otherwise} \end{cases} \tag{5.18}$$

[6]The volume of the d dimensional sphere is given by

$$c_d = \frac{\pi^{d/2}}{(d/2)!} \tag{5.17}$$

See McDonald (2003) for derivation and evaluations of $(d/2)!$. This gives values for c_d as follows

$c_1 = 2, c_2 = \pi, c_3 = \frac{4}{3}\pi, c_4 = \frac{\pi^2}{2}$

and for a 2D KDE, $c_1 = \pi$ giving

$$K_{e2}(x) = \begin{cases} 2\pi(1 - \boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}) & \text{if } \boldsymbol{x}^{\mathrm{T}}\boldsymbol{x} < 1 \\ 0 & \text{otherwise} \end{cases} \tag{5.19}$$

If the density estimate is required for some application where it is only required to proportionality (such as contour plotting) then the following kernel may suffice:

$$K_{e}(x) = \begin{cases} (1 - \boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}) & \text{if } \boldsymbol{x}^{\mathrm{T}}\boldsymbol{x} < 1 \\ 0 & \text{otherwise} \end{cases} \tag{5.20}$$

Such kernels save the necessity of determining which points have sufficient influence to be included in the summation, but have some unfortunate mathematical properties that make them unsuitable for the Bayesian KDE in the next section. The main disadvantage being the introduction of an "edge" wherever the density arrives at the zero crossing — if the resulting density is to be smooth, introducing arbitrary cut-offs should be avoided.

5.2.4 Bandwidth

At this point in any discussion of KDE it should be obvious that the one major problem is that of choosing the bandwidth h. In the literature there are many good sources for estimation of bandwidth, see for example articles by Jones et al. (1994, 1992); Park et al. (1994) and books by Silverman (1986); Scott (1983, 1992) and Wand and Jones (1995).

However, for the current project, a series of projections from some density is required (along with a KDE from each of them) quickly enough for a human

operator to perceive continuity. This implies that an automatic (no human intervention) system, with no delay caused by deliberation concerning the smoothness of the density, is needed. Speed of estimation here will improve the flow of images, thus helping the operator assess the density.

If the kernel density estimator is written in terms of some function $K(\cdot|\mu, \theta)$ where μ is a location parameter and θ, which may, more generally, be a vector, is a scale parameter, an estimate of $f_X(\cdot)$ based on the sample $\boldsymbol{x}$, at the point, $X = t$ can then be written as

$$\hat{f}_X(t|\boldsymbol{x}, \boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} K(t|x_i, \boldsymbol{\theta}) \tag{5.21}$$

where n is the number of data items. Note that $\boldsymbol{\theta}$ may be considered to be a generalisation of bandwidth. It now becomes obvious that the bandwidth may be considered as merely another parameter in the specification of the problem. Belief in the posterior for $\boldsymbol{\theta}$ given some data $\boldsymbol{x}$ can be written in the form

$$p(\boldsymbol{\theta}|\boldsymbol{x}) \propto \hat{f}_X(t|\boldsymbol{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta}) \tag{5.22}$$

This naturally leads to the Bayesian approach proposed in the next chapter.

Chapter 6

A Bayesian Kernel Density Estimator

As was seen in Chapter 5 the Kernel Density Estimator with kernel $K(\cdot)$ is defined by

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h}K\left(\frac{x - x_i}{h}\right) \tag{6.1}$$

where h is the *window width*, *smoothing parameter* or *bandwidth*. An approach is proposed in which h is a parameter of the problem, so avoiding both the specification of the bandwidth and the assumption that all projections from a density have the same smoothness.

6.1 A Bayesian Model

A KDE is specified in terms of some function $K(\cdot|\mu, \theta)$ where μ is a location parameter and θ, which may, more generally, be a vector, is a scale parameter. Examples of $K(\cdot|\mu, \boldsymbol{\theta})$ include $K(\cdot)$ a Normal pdf with locator $\mu = x_i$ and scale $\theta = \sigma$. The estimate of $f_X(\cdot)$ based on the sample $\boldsymbol{x}$, at the point, $X = t$ is

$$\hat{f}_X(t|\boldsymbol{x}, \boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} K(t|x_i, \boldsymbol{\theta}) \tag{6.2}$$

where n is the number of data items. Note that $\boldsymbol{\theta}$ may be considered to be a generalisation of bandwidth. Taking $\boldsymbol{\theta} = h$ leads to the standard KDE of (6.1).

The form $\hat{f}_X(t|\boldsymbol{x}, \boldsymbol{\theta})$ exposes clearly the dependence of the estimator on both the sample to hand $\boldsymbol{x}$ and the parameter vector $\boldsymbol{\theta}$ and gives a density estimate that is a probability density if the kernel $K(\cdot)$ is a probability density. The estimate has the continuity of the kernel $K(\cdot)$.

The need to choose h, or more generally $\boldsymbol{\theta}$, is formulated as a problem of inference about $\boldsymbol{\theta}$. Sample data $\boldsymbol{x}$ and model (6.1) allow the building of a likelihood for $\boldsymbol{\theta}$ and belief about the smoothness is expressed in a prior $p(\boldsymbol{\theta})$. Choice of $\boldsymbol{\theta}$ is provided by the posterior distribution $p(\boldsymbol{\theta}|\boldsymbol{x})$.

By taking $\hat{f}_X(x)$ to be a model for the data, a likelihood function can be constructed in the following way:

Define $\boldsymbol{x}_{(i)} = \{x_1, x_2, \ldots, x_i\}$, as the sub-sample consisting of the first i elements of $\boldsymbol{x}$. Assuming no particular ordering to the sample data (so that it can, at least, be considered exchangeable with a random sample observed

in the order implied by the subscripts), the likelihood may be written as

$$\ell(\boldsymbol{x};\boldsymbol{\theta}) = \prod_{i=1}^{n} p(x_i|\boldsymbol{x}_{(i-1)},\boldsymbol{\theta}) \tag{6.3}$$

where $p(x_i|\boldsymbol{x}_{(i-1)},\boldsymbol{\theta})$ is the probability density of x_i, given $\boldsymbol{x}_{(i-1)}$ and $\boldsymbol{\theta}$, given by the chosen model.

This is of the style commonly used for a time series $\{x_i\}$, but is more generally true. If, for example, the density were $f_X(x|\boldsymbol{\theta})$ we would have the usual formulation for a random sample

$$\ell(\boldsymbol{x};\boldsymbol{\theta}) = \prod_{i=1}^{n} f_X(x_i|\boldsymbol{\theta}) \tag{6.4}$$

If the KDE (6.2) is taken as a model for observation x_i having seen $\boldsymbol{x}_{(i-1)}$ then (6.4) can be rewritten as

$$\ell(\boldsymbol{x};\boldsymbol{\theta}) = \prod_{i=1}^{n} \hat{f}_X(x_i|\boldsymbol{x}_{(i-1)},\boldsymbol{\theta}) \tag{6.5}$$

It would be difficult to argue that $\hat{f}_X()$ is, in any sense, a 'true' model, but, in the absence of a parametric family for $p()$, it is the 'best' available model.

Since $\boldsymbol{x}_{(i-1)}$ is a parameter of $\hat{f}_X()$ there is generally some minimum number n_0 (say) of values needed for $\hat{f}_X()$ to be defined and it seems reasonable to use a conditional likelihood of the form

$$\ell(\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{x}_{(n_0)}) = \prod_{i=n_0+1}^{n} \hat{f}_{(i)}(x_i|\boldsymbol{x}_{(i-1)},\boldsymbol{\theta}) \tag{6.6}$$

The choice of prior $p(\boldsymbol{\theta})$ will, typically, reflect belief about the smoothness of the true density that $\hat{f}_X()$ is intended to estimate. For example, for the basic KDE (6.1), $\boldsymbol{\theta}$ is taken to be $\{h\}$. In this case a Normal prior for $\boldsymbol{\theta}$ with a high value of μ implies a belief in a rather smooth, near uniform density while a low value implies a non-smooth density. The variance parameter is chosen to reflect the strength of belief, a small value indicating a strongly held belief and a large value the opposite.

With small data sets the choice of prior has a large influence on the smoothness of the final estimate. It may be expected that with large data sets, for example those available from MCMC, the likelihood will dominate in the posterior and roughly non-informative priors will be a convenient initial choice.

Once the likelihood function and the prior are settled, Bayes theorem is applied to obtain the posterior density

$$p(\boldsymbol{\theta}|\boldsymbol{x}) = \frac{\ell(\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{x}_{(n_0)})p(\boldsymbol{\theta})}{p(\boldsymbol{x})} \tag{6.7}$$

in which

$$p(\boldsymbol{x}) = \int_{\Theta} \ell(\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{x}_{(n_0)})p(\boldsymbol{\theta})d\theta \tag{6.8}$$

where the integration is over the whole parameter space Θ.

The result in (6.7) is a posterior density for the parameter vector $\boldsymbol{\theta}$. The model used for the density is a KDE in which many densities are summed to model a single density. This is in no way anything like a realistic model since it depends on the sample size n. It is not believed to be the true model, however the KDE family is very rich and adaptable, and it can be reasonably

expected to provide an adequate model. It is, in any case, presumably the best available model, since otherwise some other density or approach would be used.

Situations in which the model is known, a priori, not to include the 'true' density have been considered by Berk (1966). In this situation the application of Bayes' theorem leads, asymptotically, to the member nearest to the 'true' density, in terms of Kullback-Leibler directed distance (Kullback, 1997). Berk terms this the 'asymptotic carrier'. There is then, some assurance that Bayesian methods will lead to an adequate, even optimal or near optimal, solution within the chosen modeling family — in this case KDE.

The primary objective here is that of estimating the density $f_X(\cdot)$. Within the family of KDEs, defined in terms of $\boldsymbol{\theta}$, it is easy to obtain estimates of the form $\hat{f}(\cdot|\hat{\boldsymbol{\theta}})$, where $\hat{\boldsymbol{\theta}}$ is some convenient point estimate of $\boldsymbol{\theta}$, perhaps the posterior mode or the posterior mean $E(\boldsymbol{\theta}|\boldsymbol{x})$. Such a density could be evaluated at some set of values $\boldsymbol{y}$, as $f(\boldsymbol{y}|\hat{\boldsymbol{\theta}})$, and this estimative density plotted. Here $\hat{\boldsymbol{\theta}}$ is an estimate, not the true value, and the estimate makes no allowance for the shape (or uncertainty in the estimate) of the posterior distribution for $\boldsymbol{\theta}$.

A better approach is to integrate over the parameter space of $\boldsymbol{\theta}$ giving the predictive density of the unobserved data y. See, for example Aitchison and Dunsmore (1975):

$$\hat{f}(y|\boldsymbol{x}) = \int_{\Theta} \hat{f}(y|\boldsymbol{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\boldsymbol{x}) d\boldsymbol{\theta} \tag{6.9}$$

Each point requires the evaluation of an integral like (6.9) for $y = y_i$ – unfortunately, this becomes computationally expensive. Given data with

range $x_{\text{range}} = x_{\max} - x_{\min}$, a set of points is chosen $\boldsymbol{y}$ that give sufficient information to cover x_{range}. The corresponding values of $\hat{f}(y_i|\boldsymbol{x})$ are plotted.

In (6.8) and (6.9) the integration is over the whole parameter space Θ, this is typically of low dimension but in many cases it is not possible to obtain the result analytically. In such cases numerical quadrature[1] can be applied (see Naylor and Smith, 1982).

6.2 An Archaeological Problem

As a first example, consider an MCMC analysis of the model and data described in Naylor and Smith (1988). This concerns inference about five dates forming boundaries between differing, distinct, and abutting periods of pottery production. The data consist of radiocarbon date determinations associated with pottery fragments identified as belonging to one of four periods of pottery production.

Their analysis obtains a posterior density

$$p(\boldsymbol{\alpha}|\boldsymbol{x}) = \frac{e^{L(\boldsymbol{x};\boldsymbol{\alpha})}p(\boldsymbol{\alpha})}{p(\boldsymbol{x})} \tag{6.10}$$

where

[1]Numerical quadrature or integration is the process of approximating an integral of the form $\int_a^b f(x)dx$ by a summation of the form $\sum_{k=0}^{m} w_k f(x_k)$. It can be shown that this can be achieved with any desired level of accuracy by careful choice of the nodes x_k and the weights w_k (e.g., see Davis and Rabinowitz, 1984; Shaw, 1986). As the number of nodes needed increases as the dimensionality of the function, numerical quadrature becomes more difficult to apply as the dimensionality of the problem increases.

$\boldsymbol{\alpha}$ is a vector of dates before present[2] (BP) $\alpha_1 > \alpha_2 > \alpha_3 > \alpha_4 > \alpha_5$, having prior

$$\begin{aligned} p(\boldsymbol{\alpha}) &= 1 \quad \text{if } \alpha_1 > \alpha_2 > \alpha_3 > \alpha_4 > \alpha_5 > 0 && (6.11) \\ &= 0 \quad \text{otherwise} && (6.12) \end{aligned}$$

$p(\boldsymbol{x})$ is the normalizing constant.

$L(\boldsymbol{\alpha}; \boldsymbol{x})$ is the Log-Likelihood

$$\sum_{i=1}^{n} \log p(x_i|s_i, j_i, \boldsymbol{\alpha}) \tag{6.13}$$

where

$$\begin{aligned} p(x_i|s_i, j_i, \boldsymbol{\alpha}) &= \int_{\alpha_{j+1}}^{\alpha_j} p(x|\mu(\theta), s) p(\theta|\alpha_j, \alpha_{j+1}) d\theta && (6.14) \\ &= (\alpha_j - \alpha_{j+1})^{-1} \sum_{i=l_i(j)}^{l_2(j)+1} I_i && (6.15) \end{aligned}$$

with (6.16)

$$\begin{aligned} I_i &= \int_{d_{i-1}(j)}^{d_i(j)} p(x|a_i + b_i\theta, s) d\theta && (6.17) \\ &= b_i^{-1} \left\{ \Phi\left[\frac{a_i + b_i d_i(j) - x}{s}\right] - \Phi\left[\frac{a_i + b_i d_{i-1}(j) - x}{s}\right] \right\} && (6.18) \end{aligned}$$

$\Phi(\cdot)$ being the standard Normal distribution function and $d_i, a_i, b_i, l_i(j)$ and $\mu(\theta)$

[2]Present for the article was taken to be 1983.

are values expressing a piecewise linear radiocarbon calibration curve (Clark, 1979).

$$p(\theta|j,\boldsymbol{\alpha}) = p(\theta|\alpha_j,\alpha_{j+1}) \quad \alpha_j > \theta \geq \alpha_{j+1} \tag{6.19}$$

$$= 0 \quad \text{otherwise} \tag{6.20}$$

where here θ is the actual date of manufacture.

The analysis presented here, using MCMC, incorporates into the likelihood a calibration curve, defined by Clark (1979) in piecewise linear segments, relating radiocarbon date to calendar date. This curve includes inversion (i.e. it is not monotonically increasing), which renders its use difficult. Figure 6.1 shows univariate posterior densities estimated using BKDE for four of the five boundary dates.

Taking $\boldsymbol{\theta} = \log(h)$ (ensuring that the estimate of h at any point is positive) to have prior $N(0,1)$, represents a loosely held belief ($\sigma = 1$) that the density is smooth ($\mu = 0$), corresponding to $h = 1$. That is h in the range 0.135 to 7.389 being considered probable a priori.

These densities are seen to be considerably less smooth than the corresponding results of Naylor and Smith (1988) who used an older, but smoother, calibration curve. There is some evidence that the likelihood is not smooth and other authors have found that there are considerable problems for maximum likelihood analyses (Clark, 1979; Cunliffe, 1984).

The analysis returns as many points as required. In this example $10,000$ points were generated and then reduced to 500 by extracting every 20th

result [3]. Taking $n_0 = 1$, a uni-dimensional likelihood is constructed with one previous sample.

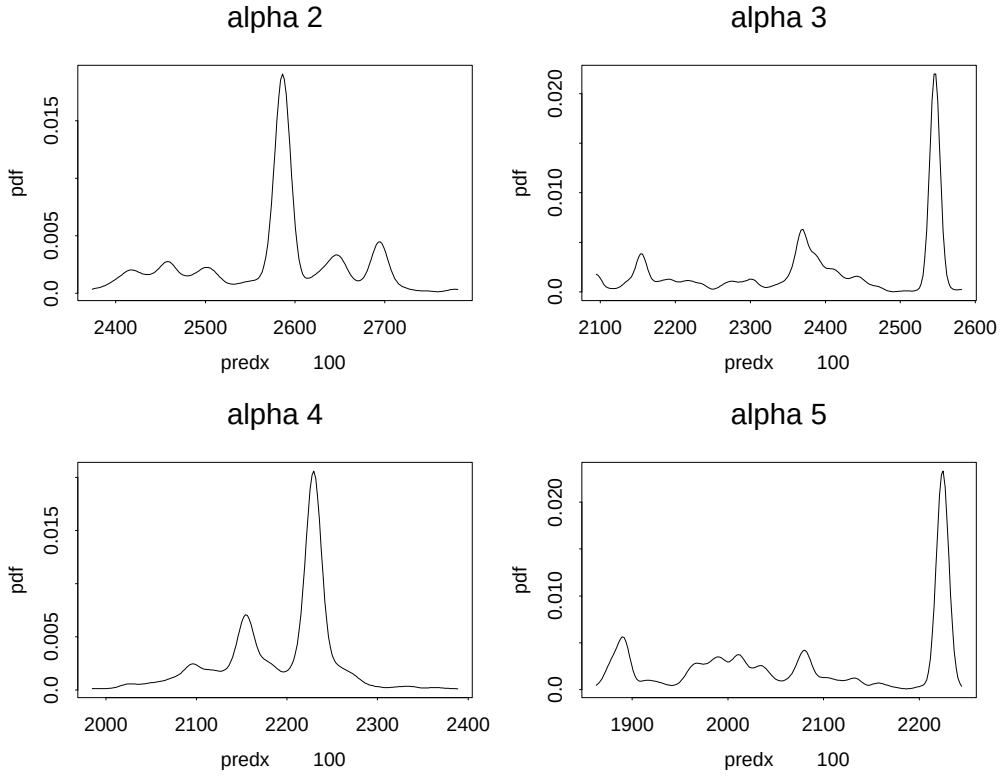


Figure 6.1: *Bayesian estimation. Predictive densities for the four date boundaries in the Archaeological problem. Axes are years Before Present (BP) (x) and density (y).*

In Figure 6.2 the effect of changing the prior is observable:

1. Prior $N(0, 1)$ loosely held belief in a smooth density, $(0.135 \leq h \leq 7.389)$.

[3]Extracting a sub-sample of the data in this way removes correlation from the result. A sample size of 500 is amenable to computation and allows demonstration of the influence of a strong prior. If 10,000 points are used then the information in almost any prior is significantly less than that in the data.

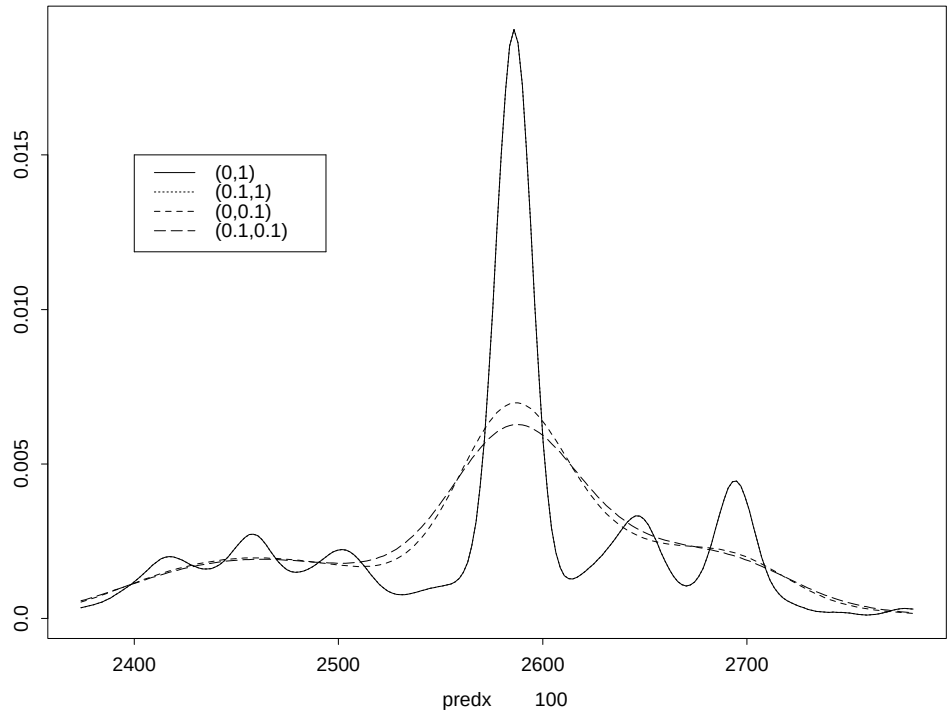


Figure 6.2: *Archaeological problem, showing the effect of using different priors for* $\boldsymbol{\theta} = \log(h)$. *Axes are years Before Present (BP) (x) and density (y).*

2. Prior $N(0.1, 1)$ loosely held belief in a slightly smoother density ($0.149 \leq h \leq 8.166$).

3. Prior $N(0, 0.1)$ stronger belief in a smooth density ($0.729 \leq h \leq 1.372$).

4. Prior $N(0.1, 0.1)$ stronger belief in a slightly smoother density ($0.806 \leq h \leq 1.516$).

On initially examining Figure 6.2, it is apparent that 1 and 2 show the same curve, the weak prior being overridden by the data. In 3 and 4 the curves are similar and smooth, the information in the data being insufficient to override the strong prior.

Having a degree of belief in the confidence placed in an "expert opinion" is

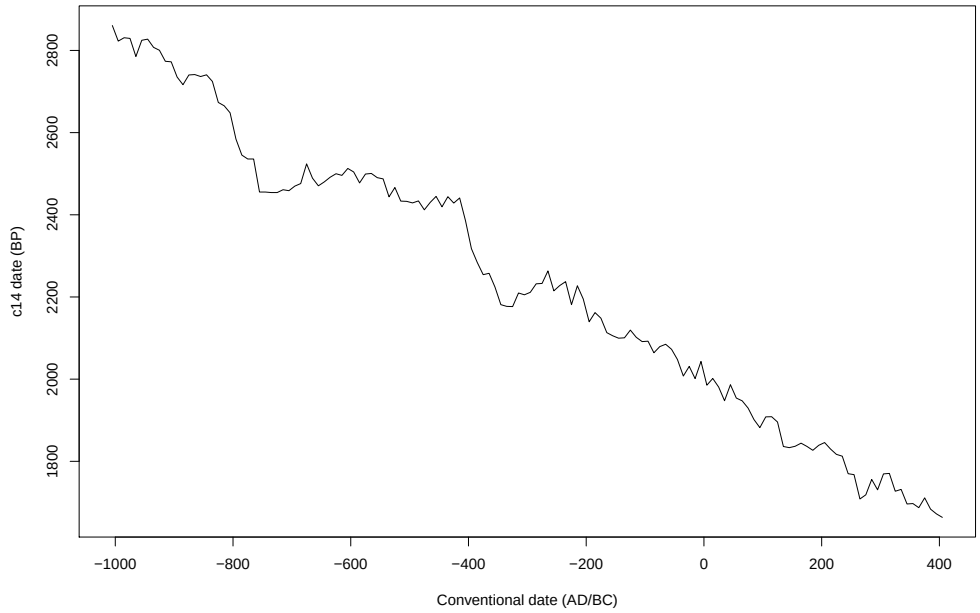


Figure 6.3: *1998 International* ^{14}C *atmospheric data set (24,000 to 0 BP). Axes are the conventional date (BC/AD) (x) and the equivalent date given by examining atmospheric carbon (BP).*

central to a subjective Bayesian approach to a problem. Priors arising from either previous knowledge of a process or from some reliable analysis of a situation may be assumed to have some degree of validity. This is expressed here in 3 and 4 by the adoption of a small value for σ leading to the retention of the smooth curve in the density estimate. The larger prior value for σ in 1 and 2 expresses the opposite, i.e. an assumption that there is no prior belief in a smooth curve and that is reflected in the density estimation obtained. Equally a strong belief in a jagged curve could be imposed, which would seem to be more appropriate here. In all cases, more data reduces the influence of the prior.

Finally, note that the jagged curve in 1 and 2 might be taken to indicate that the curve is under-smoothed, however, the ^{14}C calibration curve used here is in nowise linear as can be seen in Figure 6.3, and this might be reflected in the posterior.

6.3 Bayesian Adaptive Kernel Estimates

The method described above is readily applied to observed data. As a second example consider the Old Faithful data set again.

The curve shown in Figure 6.4 is the density, estimated using the above methods and a non-informative prior $N(0, 1)$, showing the two main modes.

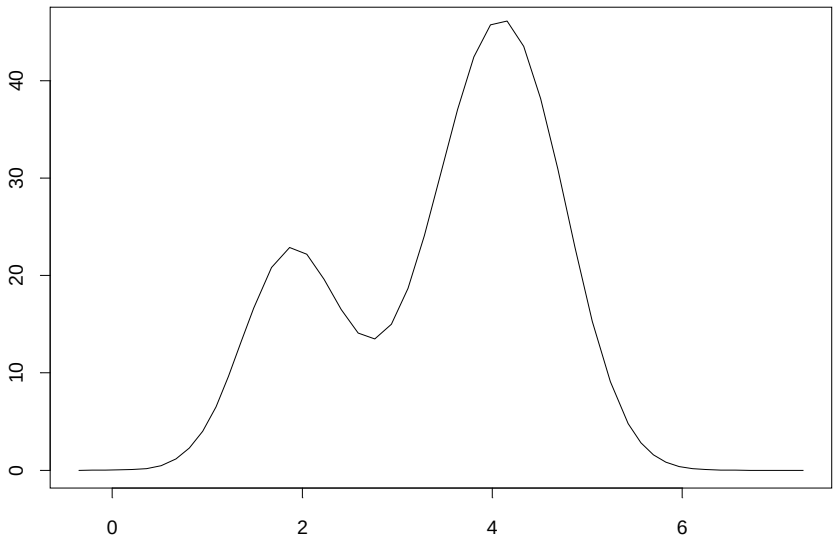


Figure 6.4: *Old faithful data, fixed h. Axes are eruption duration (x) and non-normalised, estimated density (y).*

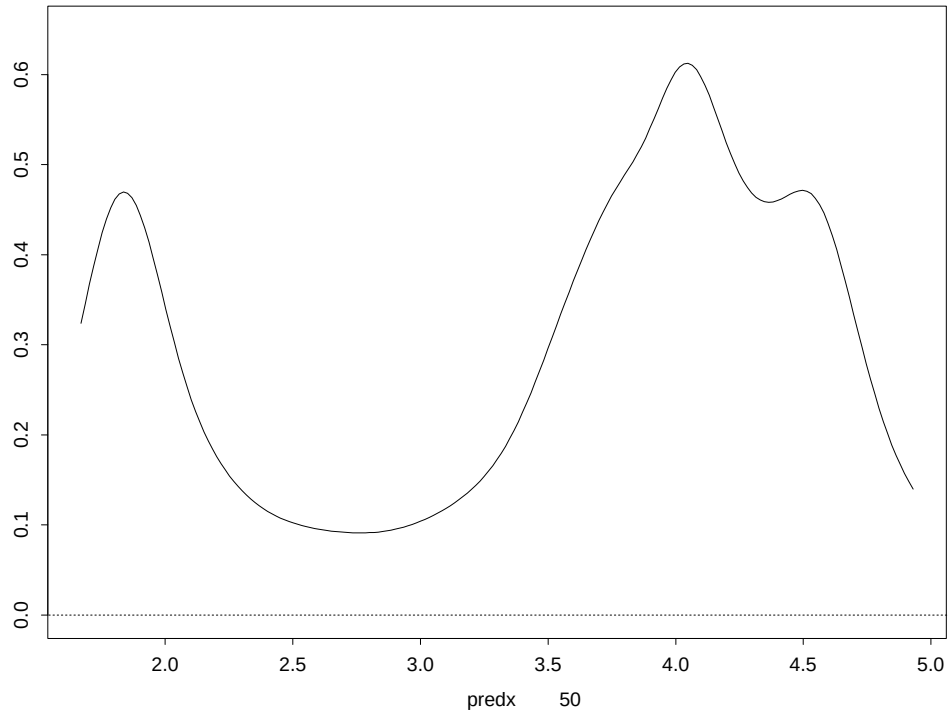


Figure 6.5: *Old faithful data, variable h, Predictive density* $p(\boldsymbol{y}|\boldsymbol{x})$ *where* x *is limited to values within the limits of the data set. Axes are eruption duration (x) and normalised, estimated density (y).*

As can be seen from the estimate, the density has two areas of high density and three of relatively low density. Intuitively a large bandwidth is required in an area of low density and a small bandwidth in an area of high density, smoothing out the tails and revealing more detail in the high density areas. This leads to the notion of an adaptive, two pass, estimate where the bandwidth is inversely proportional to the local density.

An estimate is required that works in a similar way. An initial KDE is used to modify the bandwidth used in the final estimate.

First a fixed bandwidth KDE is adopted as pilot estimate, so that, for such a pilot, on a sample $\boldsymbol{x}_{(n)}$

$$\hat{f}_n(t) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h_p}K\left(\frac{t-x_i}{h_p}\right) \tag{6.21}$$

where h_p is the pilot bandwidth. Following Abramson (1982) and taking $\alpha = \frac{1}{2}$ in 5.10 gives

$$\lambda_i = \sqrt{\frac{g_n}{\hat{f}(x_i)}} \tag{6.22}$$

as the local bandwidth. For univariate data with ($d = 1$) equation (5.12) gives

$$\hat{f}_n(t) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h\lambda_i}K\left(\frac{t-x_i}{h\lambda_i}\right) \tag{6.23}$$

where

$$h\lambda_i = h\sqrt{\frac{g_n}{\hat{f}(x_i)}} = \frac{h_1}{\sqrt{\hat{f}_n(x_i)}} \tag{6.24}$$

The factor g_n is absorbed in h as discussed in section 5.2.2. So it is seen that (6.23) is of the form

$$\hat{f}_n(t|\boldsymbol{x},\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}K(t|x_i,\boldsymbol{\theta}). \tag{6.25}$$

This is similar to (6.2) and is still easily accommodated in the formulation based on (6.3).

Taking

$$p(x_i|\boldsymbol{x}_{(i-1)}, \boldsymbol{\theta}) = \hat{f}_{(i-1)}(x_i) \tag{6.26}$$

a likelihood for the analysis can be constructed as before.

Figure 6.5 shows the density obtained from the eruption duration subset of the Old Faithful data using the adaptive KDE. The two main peaks are still there, however there is a third peak appearing at around 4.5 minutes that is not obvious in the simple estimate. This is also seen in Figure 6.6 and, to a slightly reduced level, in the larger data set in Figure 6.7[4] (this data has 298 items as opposed to 107).

6.4 Examples – Hard to estimate densities

The paper by Berlinet and Devroye (1994) is an attempt to provide a torture trial of a wide range of KDEs. Appendix D contains estimates of each of the 28 densities in the paper, the estimates given are:

- A histogram.

  Providing an example of simple density estimation, a sample of 1000 from the density in question giving a reasonable picture.

- An estimated density for a sample of 10000 points using the inbuilt R function `density()`.

  R provides a function `density` that is a highly accurate KDE, this,

[4] In these figures, 17 bins refers to the command used to generate the image, in this case `hist(oldf2$lengthm, main=" ",xlab=" ", breaks = 17)`, this is a much higher bin count than would normally be used with this data but is an informed choice.

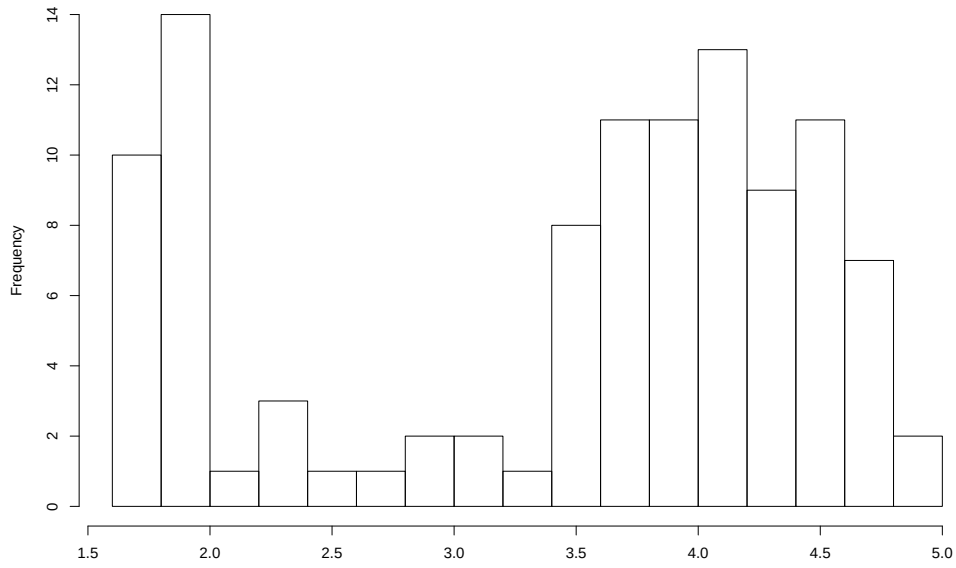


Figure 6.6: *Old Faithful data - histogram with 17 bins.*Axes are eruption duration (x) and frequency (y).

with a sample of 1000 data items is provided as a reference (referred to as the R estimator in the following).

- An estimated density for a sample of 1000 points using BKDE.
- An estimated density for a sample of 1000 points using adaptive BKDE.
- An estimated density for a sample of 100 points using BKDE.
- An estimated density for a sample of 100 points using adaptive BKDE.

Each sub-figure is overlaid with the output from

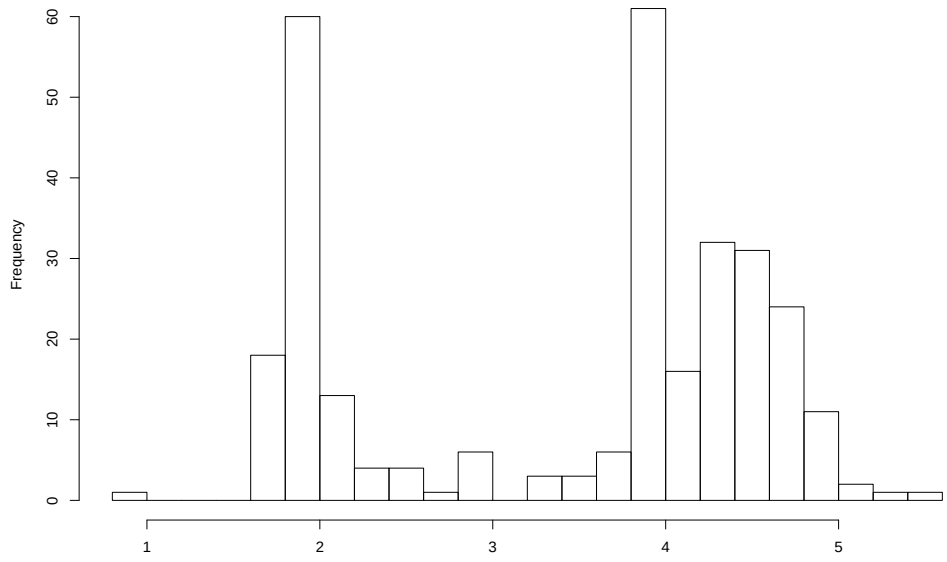


Figure 6.7: *Old Faithful data, larger data set - histogram with 17 bins.Axes are eruption duration (x) and frequency (y).*

`dberdev(seq(-n,m,0.01),dnum=1)`[5].

which gives the source density. The histogram and the R density estimator are from samples that are generated for the figure, the BKDE and VBKDE figures are from the same samples in each case. Each sample is taken as found and each test is a single sample unless otherwise noted. Presented here are a few of the estimations where BKDE can be seen to be significantly different to the expected result, from the histogram or from the R estimator.

It should be noted that the interest here is not in absolute accuracy, but in good behaviour over a wide range of density shapes. The application of

[5]Where the value of `dnum` varies from 1 to 28 giving the appropriate line for each density. The values `n` and `m` provide a limit to the range of `dberdev` so the graph is not swamped.

KDE to a problem, combined with the use of a Bayesian predictive estimation is known to select the best possible model, from those available (Berk, 1966). KDE (of any form) presents a very rich set of available models. For particular data it is possible to select a kernel to get a better fit, but that requires intervention. The techniques presented here are intended to give a good initial view of as wide a range of data as possible, making their use as automatic as possible. For this reason the comparison in what follows is with the R estimator (in its default mode) and with the BKDE estimators using a normal kernel. Formal goodness of fit is not applied because the density estimation is returned as density values at the predictive points. In addition the aim is not to optimise density estimation

For each estimate the y axis is normalised to a $0-1$ scale, x axis is value.

6.5 Exponential Density

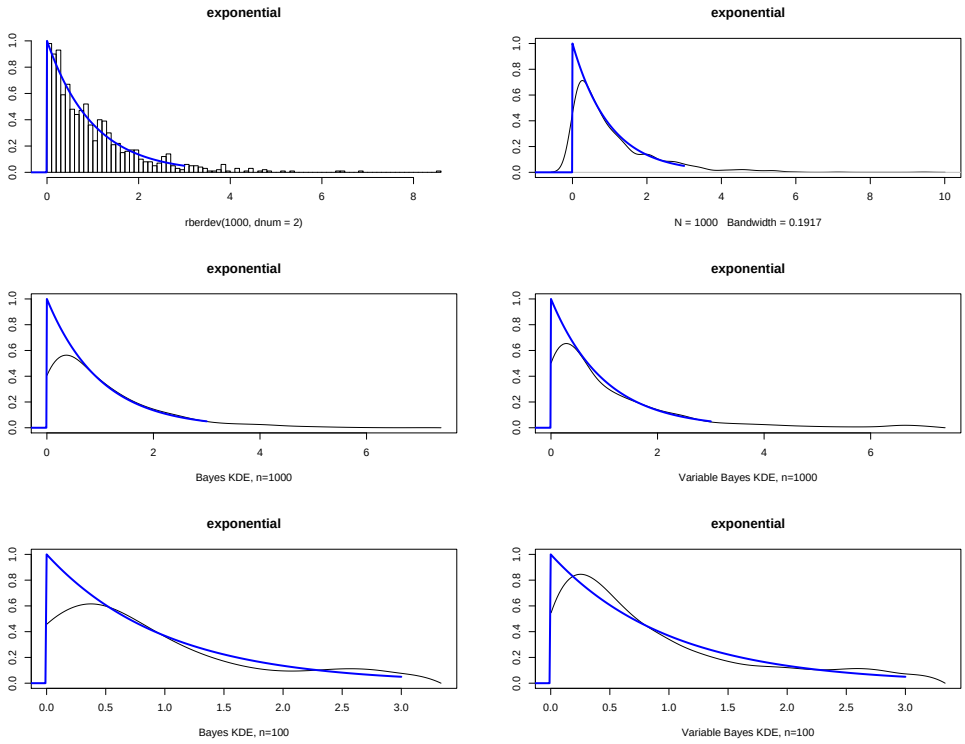


Figure 6.8: *Exponential density.*

This shows a difference in software set-up. The R estimator returns values below 0 which cannot exist, the BKDE estimators are limited to the range of the data by a choice made when writing the problem specific code for Bayes4. Other than this the plots are all similar.

6.6 Maxwell Density

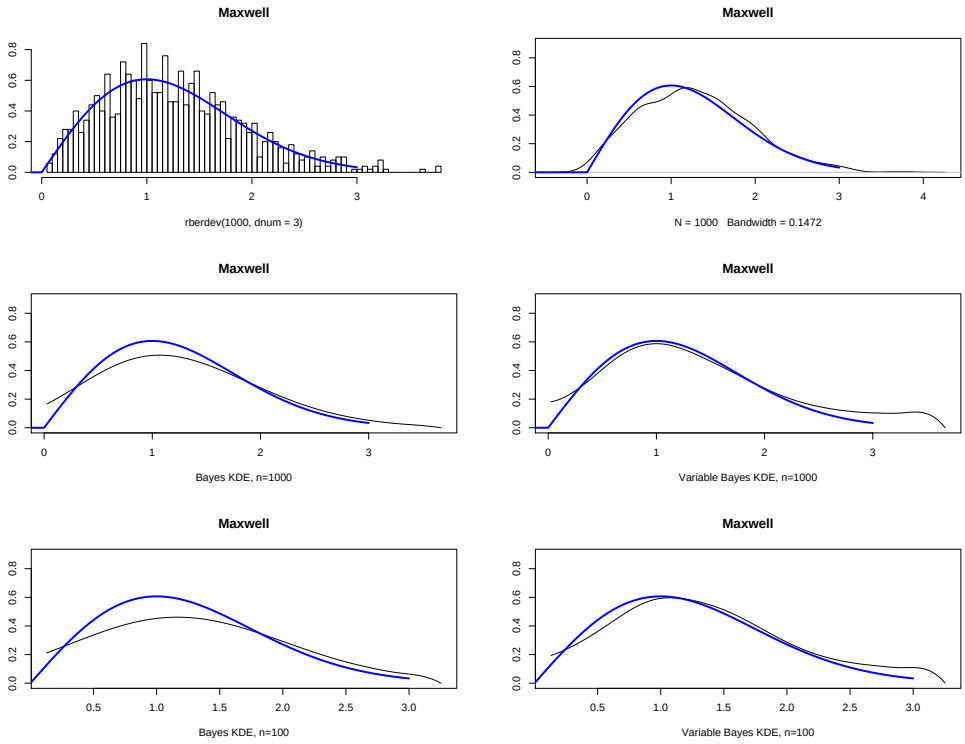


Figure 6.9: *Maxwell density.*

Here the plots are again similar, however the R estimator is showing a lack of smoothness all four BKDE estimates are similar.

6.7 Cauchy Density

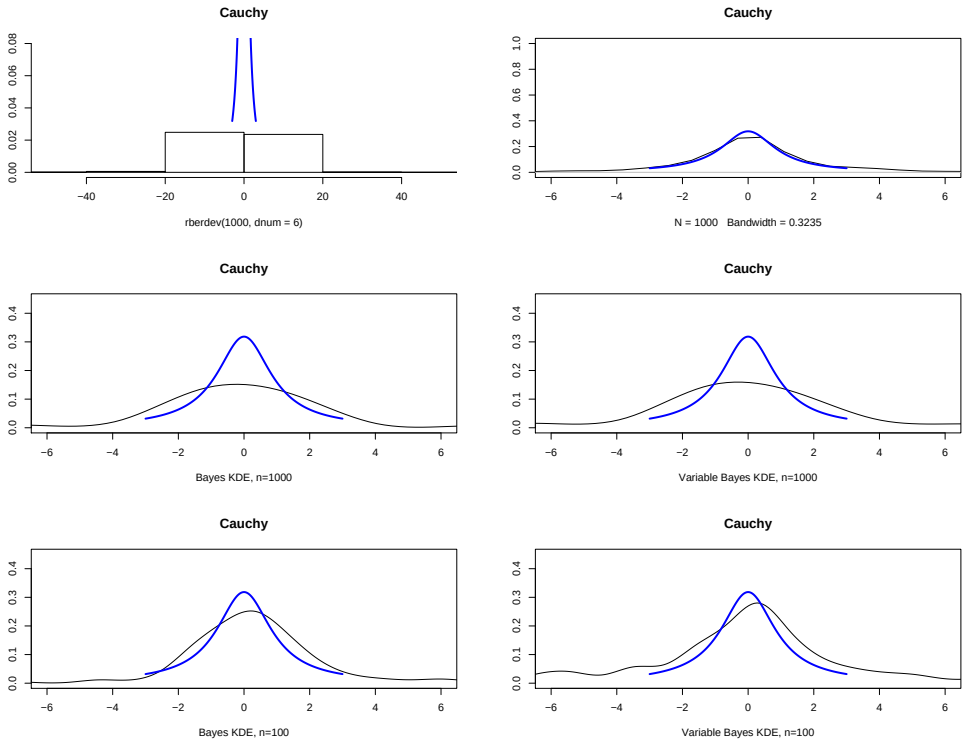


Figure 6.10: *Cauchy density.*

The Cauchy Density is a difficult density that the BKDE does not handle well, however this is interesting because BKDE has performed better with the sample of 100. This is due to sample variation and not intrinsic to the method.

6.8 Infinite Peak Density

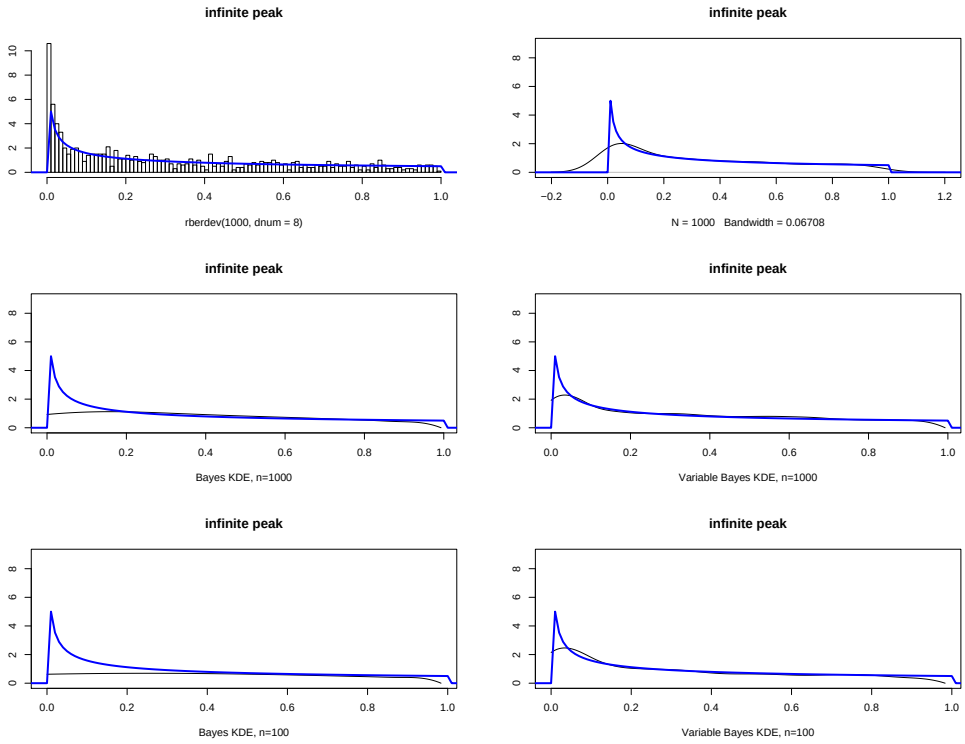


Figure 6.11: *Infinite Peak density.*

Here the advantage of limiting to the range of the data is seen with the R estimator producing a very different estimate to the BKDE, and the variable BKDE performing somewhat better than BKDE.

6.9 Pareto Density

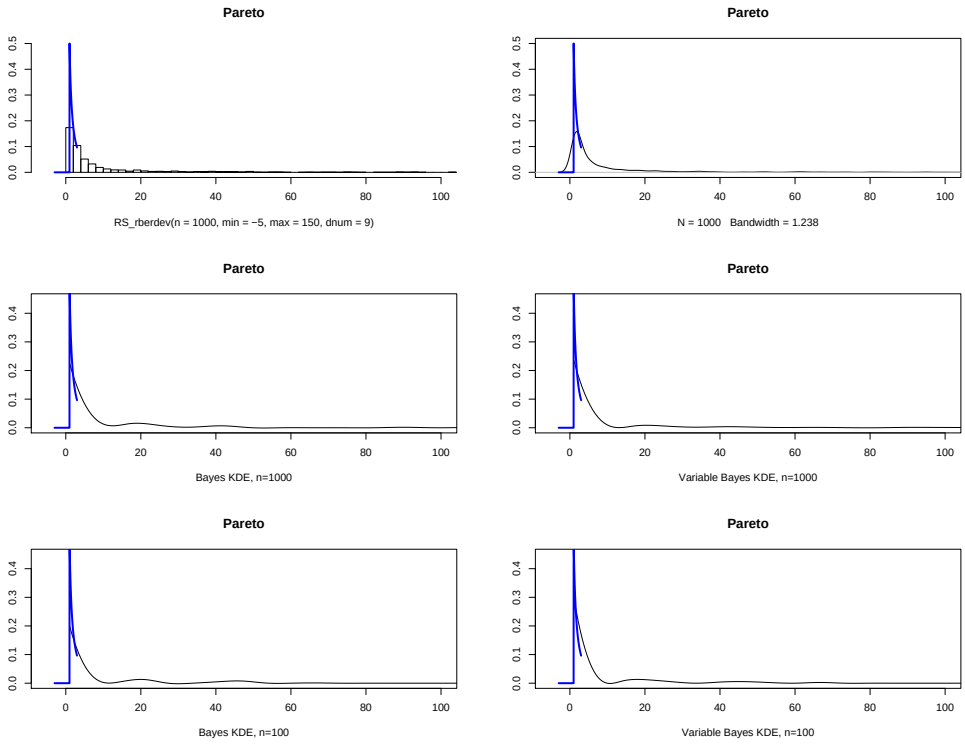


Figure 6.12: *Pareto density.*

This is an example of a density which can have an extremely wide range of values. In 10 samples of 1000 from this density the maximum values were:

277150.1	5461690	3065065	408696.6	130378.8
510953.3	142911.3	22289502871	12736939	4845980

Unfortunately this wide range makes it difficult to obtain an estimate. For this reason the sample here is limited to a maximum value of 150 with the sample drawn from a parent sample large enough to give the desired sample size with values less than 150. Once this has been done the estimators all give acceptable results.

6.10 Beta (2,2) Density

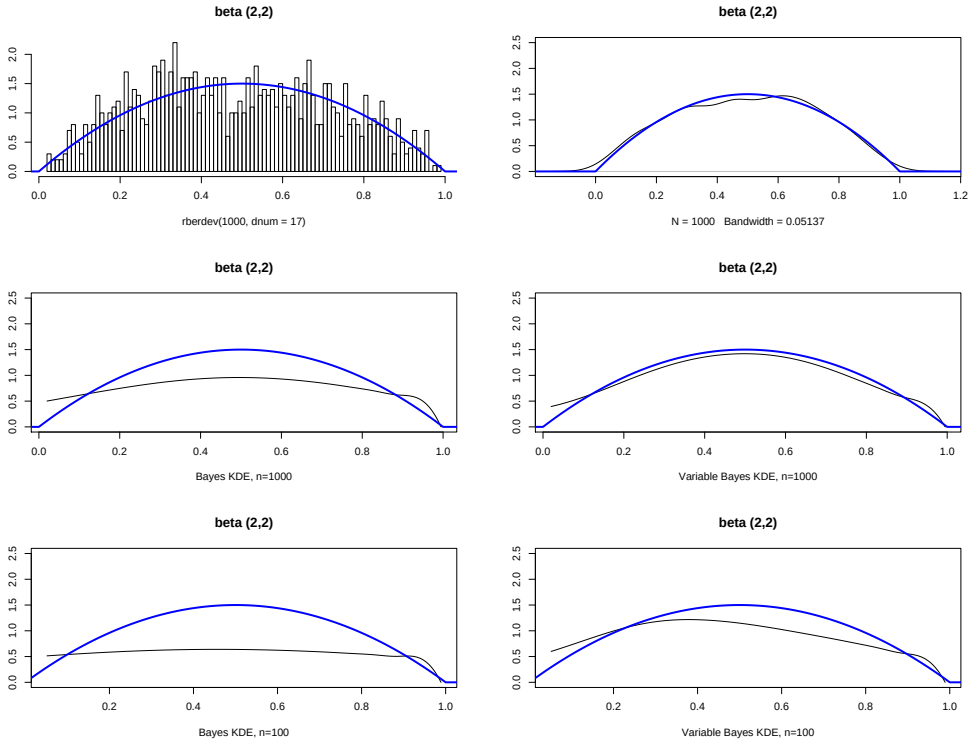


Figure 6.13: *Beta (2,2) density.*

The Beta (2,2) Density is limited in both the negative and positive directions. It can be difficult to estimate with a kernel that has tails to infinity. This is apparent here, as is the failure of non-adaptive BKDE with this density.

6.11 Smooth Comb Density

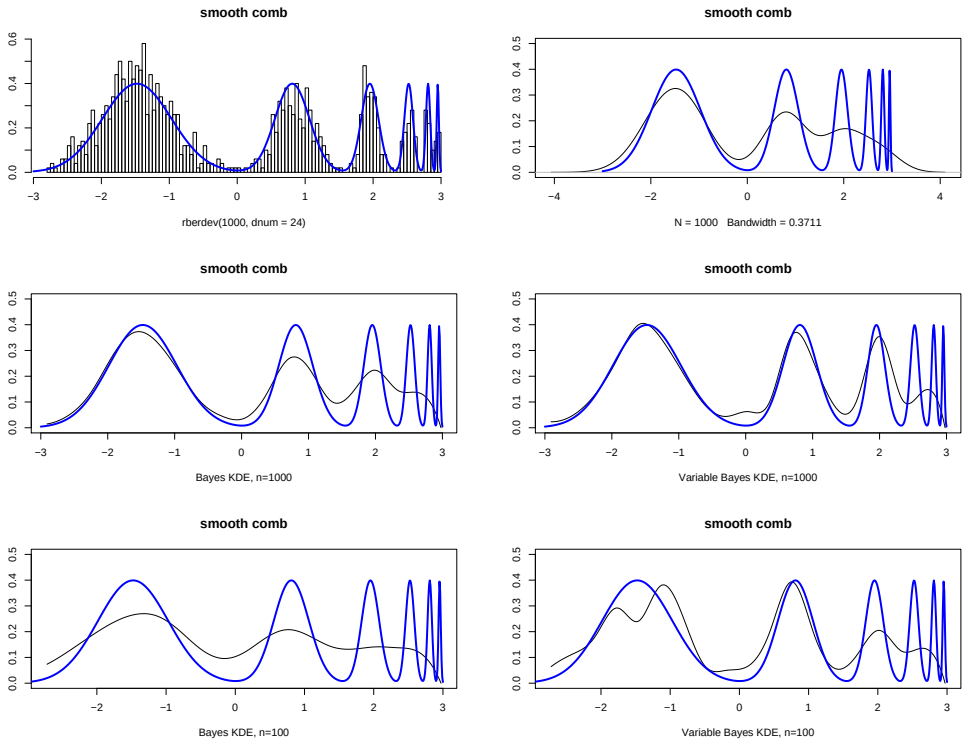


Figure 6.14: *Smooth Comb density.*

The Smooth Comb Density a difficult density that appears to have some affinity with BKDE with even the samples of size 100 producing acceptable results.

6.12 Sawtooth Density

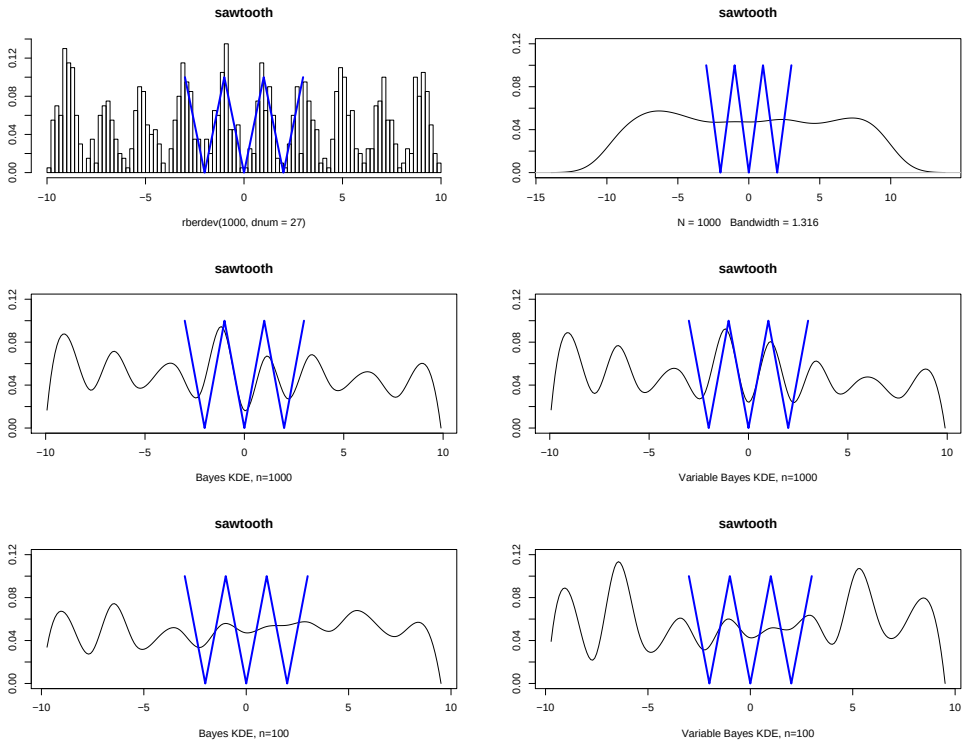


Figure 6.15: *Sawtooth density.*

The Sawtooth Density is another density where BKDE produces reasonable results even with a sample of 100.

6.13 Extension to bivariate density estimation

Outputs from the Grand Tour can be projections onto subspaces of any dimensionality, usually one two or three for convenience, as discussed in section 4.1.1. Extending the BKDE discussed above to more than one dimension is

both natural and easy. A KDE is still specified in terms of some function $K(\cdot|\mu, \theta)$ and an estimate of $f_X(\cdot)$ based on the sample $\boldsymbol{x}$, at the point, $X = \boldsymbol{t}$ is written as

$$\hat{f}(\boldsymbol{t}; \boldsymbol{H}) = \frac{1}{n} \sum_{n=1}^{i} K_{\boldsymbol{H}}(\boldsymbol{t} - \boldsymbol{x}_i) \tag{6.27}$$

where

$$K_{\boldsymbol{H}}(\boldsymbol{x}) = \frac{1}{\sqrt{|\boldsymbol{H}|}} K\left(\boldsymbol{x}\boldsymbol{H}^{-1}\boldsymbol{x}^{-1}\right) \tag{6.28}$$

$\boldsymbol{H}$ is a bandwidth matrix and K is some bivariate kernel.

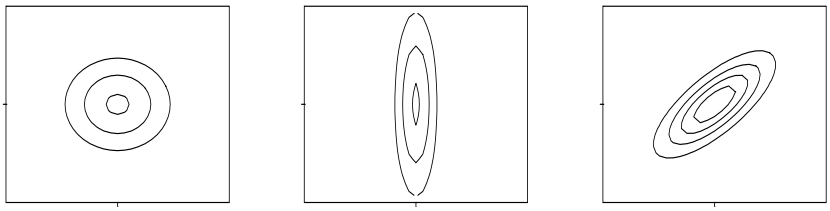
The estimate of $f_X(\cdot)$ based on the sample $\boldsymbol{x}$, at the point $\boldsymbol{t}$, $\boldsymbol{X} = \boldsymbol{t}$ is seen to again be of the form

$$\hat{f}_X(\boldsymbol{t}|\boldsymbol{x}, \boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} K(\boldsymbol{t}|\boldsymbol{x_i}, \boldsymbol{\theta}) \tag{6.29}$$

where n is the number of data items. Note that $\boldsymbol{\theta}$ may be considered to be a generalisation of bandwidth. Taking $\boldsymbol{\theta} = h$ with univariate data still leads to the standard KDE of (6.1), however, taking $\boldsymbol{\theta} = \boldsymbol{H}$ leads to some higher dimensional estimate. In this case interest is in bivariate data and a 2×2 matrix form of $\boldsymbol{H}$.

There are three possible orders of complexity for $\boldsymbol{H}$; if $\boldsymbol{H} \in \mathcal{F}$, the class of all symmetric, positive, definite 2×2 matrices, then there are 3 bandwidth parameters to choose; if $H \in \mathcal{D}$, the subclass of all diagonal, positive, definite 2×2 matrices, then there are 2 bandwidth parameters to choose; and finally, if $H \in \mathcal{S}$, where $\mathcal{S} = \{h^2\boldsymbol{I} : h > 0\}$, there is only 1 bandwidth parameter to choose.

However, a compromise between the work needed to estimate the bandwidth



(a) (b) (c)

Figure 6.16: *Contour plots of Normal kernels parameterised by (a)* $\boldsymbol{H} \in \mathcal{S}$*, (b)* $\boldsymbol{H} \in \mathcal{D}$ *(c)* $\boldsymbol{H} \in \mathcal{F}$*. Axes are both normalised, the same and arbitrary.*

and the time taken to perform the estimation is required. Fukunaga (1972, p. 175) suggests a simple way of obtaining a bandwidth matrix of arbitrary orientation (see Silverman, 1986, p. 78). Take $\boldsymbol{H}$ to be of the form

$$\boldsymbol{H} = h^2 \boldsymbol{S} \tag{6.30}$$

where $\boldsymbol{S}$ is the covariance matrix. This approach is equivalent to sphering the data (i.e. transforming it to have unit covariance matrix).

This gives an estimate of the form

$$\hat{f}(\boldsymbol{x}) = \frac{1}{n\sqrt{\det S}} \sum_{i=1}^{n} \frac{1}{h} k\left(\frac{(\boldsymbol{x} - \boldsymbol{x}_i)^T (\boldsymbol{x} - \boldsymbol{x}_i)}{h^2 S} \right) \tag{6.31}$$

It can be shown (Wand and Jones, 1995, p. 106) that, for the multivariate $N(\mu, \sigma)$ distribution, the Asymptotic Mean Integrated Squared Error (AMISE) optimal $\boldsymbol{H}$ satisfies

$$\boldsymbol{H}_{AMISE} = c\Sigma \tag{6.32}$$

for a scalar constant c. This implies that, for the multivariate Normal, sphering is appropriate. There is, unfortunately, no equivalent result for estimation of arbitrary density shapes. The approach taken for the version of the bivariate BKDE incorporated into the Grand Tour is to sphere the data. By taking $\hat{f}_X(x)$ to be a model for the data a likelihood function is constructed as before.

$$\ell(\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{x}_{(n_0)}) = \prod_{i=n_0+1}^{n} \hat{f}_{(i)}(\boldsymbol{x}_i|\boldsymbol{x}_{(i-1)},\boldsymbol{\theta}). \tag{6.33}$$

Choice of prior for $\boldsymbol{\theta}$ again indicates belief in the smoothness of the underlying density and in the strength of that belief. This gives the posterior density

$$p(\boldsymbol{\theta}|\boldsymbol{x}) = \frac{\ell(\boldsymbol{x};\boldsymbol{\theta},\boldsymbol{x}_{(n_0)})p(\boldsymbol{\theta})}{p(\boldsymbol{x})} \tag{6.34}$$

and the predictive density

$$\hat{f}(\boldsymbol{y}|\boldsymbol{x}) = \int_{\Theta} \hat{f}(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta})p(\boldsymbol{\theta}|\boldsymbol{x})d\boldsymbol{\theta}. \tag{6.35}$$

6.14 Discussion

Interest here is in initial examination of data. This means that we are interested in automated methods that produce easy to evaluate images. For this reason we have chosen automation over accuracy of estimation or single dimension projection. The main area of compromise is in kernel choice. If we are aware that there is a limit to the density, for example in estimating the underlying distribution for survival data (where negative values are not

possible), then it is reasonable to choose estimation methods that do not violate this. However, when the expected density is not only unknown but unpredictable it makes sense to choose a general kernel such as the Normal, or possibly one with definite limits such as the Epanechnikov.

The application of Bayesian model selection to such a rich family of models with moderately large amounts of data is computationally intensive. The Software used here, written within Bayes4, runs in On^3 time[6]. The plotted output is a spline curve plotted using the *plot* function in R, this provides an easy way to plot data but does not provide a sample suitable for distance based goodness of fit measures such as the Kolmogorov-Smirnov tests (available in R via the `ks.test()` function) or information based tests such as Kullback-Leibler divergence (available in R via the `KLdiv()` function).

The choice of kernel is as general as possible, if a single density estimate is required it makes sense to tailor the kernel for the expected target density and to use adaptive estimation.

If a density is such that different values of bandwidth are appropriate for different areas of it, then the best a single value of bandwidth can do is a compromise. Intuitively a large bandwidth is required in an area of low density and a small bandwidth in an area of high density, smoothing out the tails and revealing more detail in the high density areas. For example,

[6]In mathematics, computer science, and related fields, big O notation describes the limiting behavior of a function when the argument tends towards a particular value or infinity, usually in terms of simpler functions. Although developed as a part of pure mathematics, this notation is now frequently also used in the analysis of algorithms to describe an algorithm's usage of computational resources: the running time or memory usage of an algorithm is often expressed as a function of the length of its input using big O notation.

when applied to long-tailed distributions the fixed kernel estimator will tend to under-smooth the tail. In order to overcome this, the bandwidth of the estimator may be allowed to vary in some way, for example it is possible to have a bandwidth inversely proportional to the local density obtained from some previous estimate,

BKDE seems to provide some advantages:

1. The Bayes Density Estimator provides a kernel density estimate, without the need for bandwidth choice by the user. This has the distinct advantage that, when applied to a series of projections, the bandwidth need not be assumed to be the same for all projections. The corollary to this is that the data need not be distorted in trying to ensure uniform smoothness for all projections.

2. The model found by the BKDE should be the best available from the family of models provided (Berk, 1966), which, in the case of KDE models, gives a wide range to choose from. This leads to a high degree of confidence in the density estimate obtained.

3. The bandwidth found is not a point estimate but a density with location and shape parameters. This allows for examination of the chosen bandwidth for suitability and might lead, in further work, to a system of kernel suitability evaluation.

4. The choice of kernel affects the final model, some kernels being more suitable for particular types of data, for example, survival data is not well served by any kernel that allows negative values. The BKDE allows for rapid selection of kernel and might lead to suitability analysis of the kernel.

5. The prior allows for the modelling of belief in the smoothness of the underlying density. The strength of that belief can also be represented and allows for a wide variation in the balance between that belief and the information from the data.

6. With the large samples obtained from MCMC simulation, the prior is dominated by the information from the data in the likelihood, however it is possible to force a "wrong" prior on the system. A very strongly held belief in a prior, for example $N(0.1, 0.01)$ in section 6.2 needs substantially more data to modify than that of $N(0, 1)$. used in graph 1. of Figure 6.2.

The examples from Berlinet and Devroye (1994) are designed to be difficult to estimate and the results in that paper are all averages of 20 different samples of size 100. The overwhelming conclusion to be drawn from it is that no one KDE will do well at all densities and some experimentation with method is needed. However, the BKDE in one of its forms produced acceptable estimates of a large number of the densities without the need for human intervention. As a method KDE compares well to several others, in terms of producing a reasonable output, and can be considered at least the equal of most.

In these examples both BKDE and adaptive BKDE have been used. The adaptive variant of any KDE is useful in density estimation where the density changes, as observed in Section 6.3. Intuitively a large bandwidth is required in an area of low density and a small bandwidth in an area of high density: smoothing out the tails and revealing more detail in the high density areas. However, the addition of the initial estimate used to govern the varying bandwidth of the final estimate adds an operation On^3 to each iteration

of the final estimate. while this is useful in a single estimate (for example the Old Faithful data estimate shown in Figure 6.5), it is an unnacceptible increase in overhead for the automated Grand Tour.

Chapter 7

Integration of the Grand Tour and Bayesian Kernel Density Estimator

Previous chapters have reviewed existing tools that lend themselves to the process of examining some high dimensional, posterior density. This would, ideally, lead to a system that could be encapsulated in a single software system that would take either a mathematical description of a posterior density, or a sample from such a density, and display it in a human-friendly fashion.

The obvious problem is in going from a mathematical description of a density to a sample from that density. Once obtained, using MCMC methods, the sample can be of any required size so the "curse of dimensionality" can be ignored, but Markov chain simulations are very problem specific and so the first step in the sequence has to be written individually for each problem that needs it.

The Grand Tour and Density Estimation phases, however, fit together well, the automation available with Bayesian Kernel Density Estimation (BKDE) providing the required functionality for the combination. This chapter will explore that joining.

The Grand Tour produces a conditional density from the complete sample. This conditional density is dependent on the current tour step position. As the operator moves through the tour they see a sequence of projected densities. When an interesting projection is found, the operator should be able to stop and change the view to either a contour plot, or a greyscale density view.

In order to achieve the density view KDE is used, however, conventional KDE has the difficulty that the bandwidth needs to be chosen accurately for each projection. This, with its customary operator intervention, takes far too much time and disrupts the flow of information. The Bayesian Kernel Density Estimator (BKDE) discussed in Chapter 6 removes the necessity for such operator intervention, data modification (for example, sphering) or the use of *ad-hoc* values.

7.1 Basic Grand Tour S-Plus Implementation

The Grand Tour is implemented as several sets of S-Plus routines (see Appendix A). A basic tour is shown in Figure 4.1. As discussed in Section 4.1.1 a Grand Tour is a series of projections from n dimensional space onto a space of lower dimension, the target space is usually 1D or 2D.

In the current work the target is a plane in one or two dimensions and the tour is achieved by rotating this target, in n dimensions, and then projecting

the data onto it. For example, if the problem space has five dimensions then the starting plane for a 2D output is represented by

$$P = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

and each step of the tour is given by RP where R is the rotation matrix. Some authors use of the first 1, 2 or 3 rows of the data matrix as a projection matrix, however the above ensures axes that are orthogonal.

The rotation matrix

The rotation matrix is derived as follows:

A rotation in 2D looks like $\begin{pmatrix} cos\ \theta & sin\ \theta \\ -sin\ \theta & cos\ \theta \end{pmatrix}$ in p dimensions it is made up of a product of matrices like

$$\begin{pmatrix} cos\ \gamma_{12} & sin\ \gamma_{12} & & & 0 \\ -sin\ \gamma_{12} & cos\ \gamma_{12} & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{pmatrix}$$

with the general form

$$\begin{pmatrix}
1 & & & & & & & & & & \\
& \ddots & & & & & & & & & \\
& & 1 & & & & & & & & \\
& & & cos\ \gamma_{ij} & & \dots & & sin\ \gamma_{ij} & & & \\
& & & & 1 & & & & & & \\
& & & \vdots & & \ddots & & \vdots & & & \\
& & & & & & 1 & & & & \\
& & & -sin\ \gamma_{ij} & & \dots & & cos\ \gamma_{ij} & & & \\
& & & & & & & & 1 & & \\
& & & & & & & & & \ddots & \\
& & & & & & & & & & 1
\end{pmatrix}$$

The matrix for the total rotation given by $R_{12}(\gamma_{12}), \dots, R_{1p}(\gamma_{1p}), R_{23}(\gamma_{23}), \dots, R_{2p}(\gamma_{2p})$ where $R_{ij}(\gamma_{ij})$ is the matrix that rotates the subspace (X_i, X_j) through the angle γ_{ij} and corresponds to the identity matrix except for the elements:

$$(i,i) = (j,j) = cos(\gamma_{ij})$$

$$(i,j) = -(j,i) = sin(\gamma_{ij})$$

Labeling $\gamma_{12}, \dots, \gamma_{1p}, \gamma_{23}, \dots, \gamma_{2p}$ as $\gamma_1, \dots, \gamma_L$ with $L = 2p - 3$, the angles are given by

$$\gamma_i = z\sqrt{P_i}, i = 1, \dots, L$$

where P_i is the i^{th} prime number and z is any irrational number. Marriott and Eslava (1994) use a step of $z = \sqrt{5}$ which "is large enough to produce pseudo-random projections".

R is the product of all the R_{ij} matrices.

7.2 Examples

In the Grand Tour implementation given in Appendix A, mouse control is provided to allow sweeping forward or backward from the current view. Tools are provided that give a contour plot or a coloured image plot of the current projection based on a BKDE of the density embodied in the view. Example screen shots of this implementation are shown in Figure 7.1 and Figure 7.2.

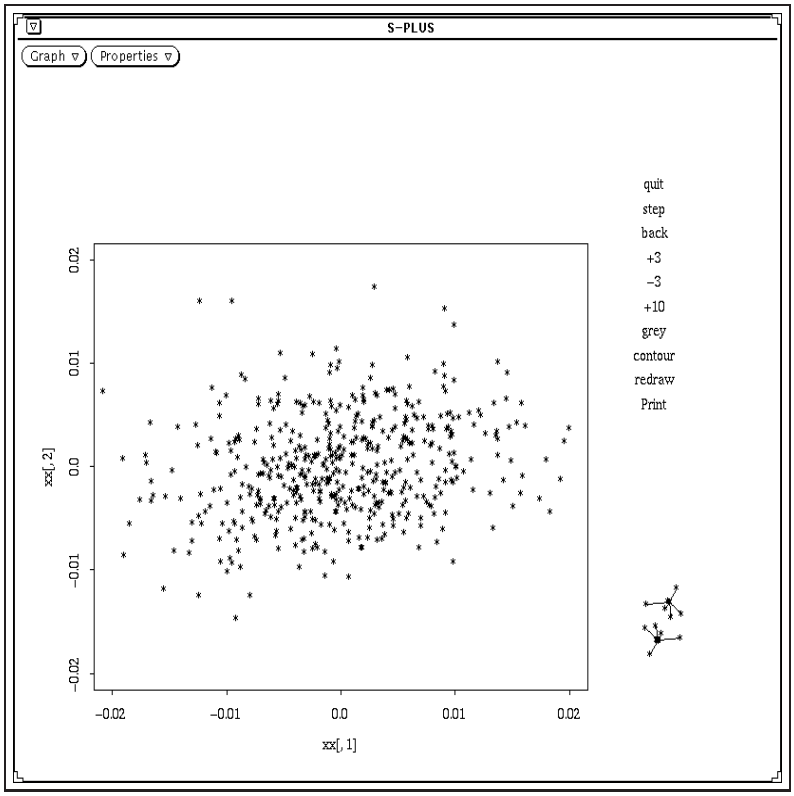
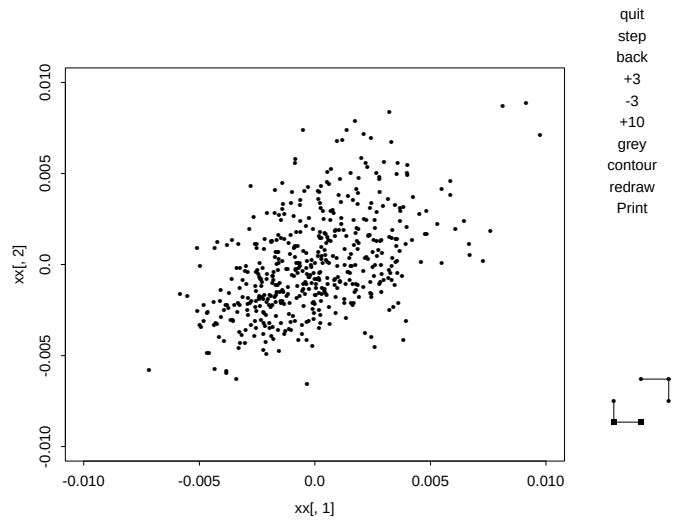
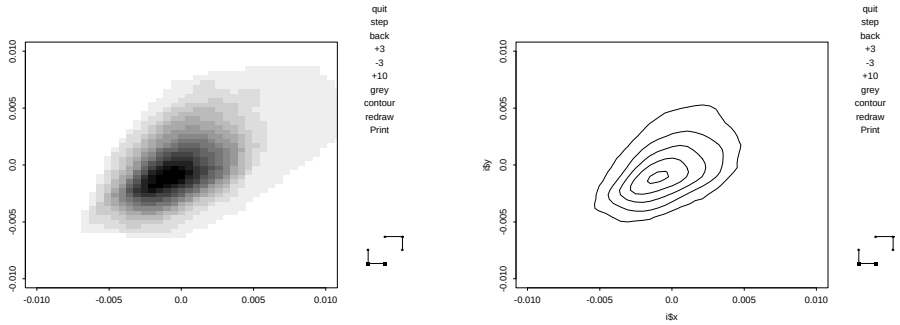


Figure 7.1: *View generated by the Grand Tour routines written in S-Plus - 24 dimensional data set, 2 dimensional target. Axes are projected values sized to fit all possible projections.*

In addition, an indicator to give a visual guide to orientation of the view is provided. This consists of two opposite corners of an n dimensional "cube"



(a) Scatter plot.



(b) Greyscale image.

(c) Contour plot.

Figure 7.2: *Three views of the starting projection of the same Grand Tour of the 24 dimensional data set, 2 dimensional target. Axes are projected values sized to fit all possible projections.*

that have been rotated to the same orientation as the main view.

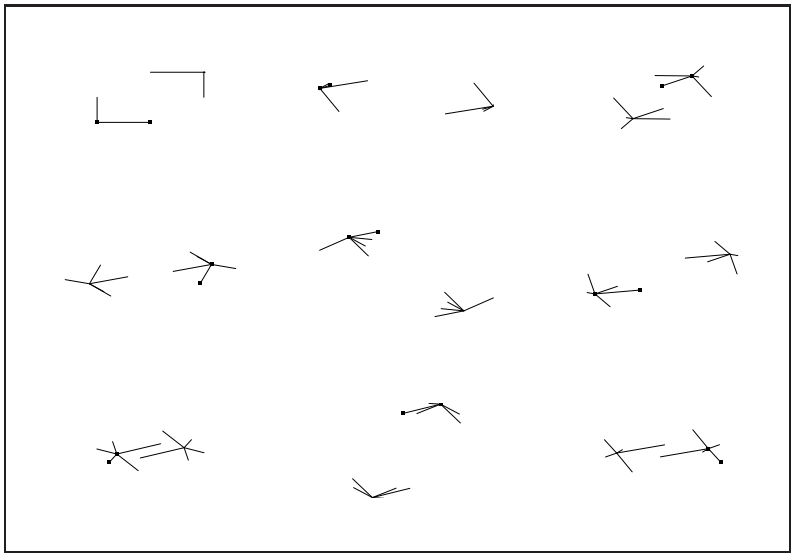


Figure 7.3: *Examples of the indicator – a 5 dimensional "cube" rotated through 8 steps of a fairly coarse Grand Tour.* **Note:** *the two black squares always mark the ends of the same segment of the cube.*

7.3 Discussion

Once the basic elements of the tour (the rotation matrix, stepping function and display function) are available then the tour can be constructed simply. Various different configurations have been used to produce the examples here and are shown in appendices A, B and C in sufficient detail to allow the reader to construct various, useful tours in R or S-Plus, with or without BKDE.

Given a sample from a posterior density, regardless of how the sample is obtained, the combination of the Grand Tour with some kind of good, automatic, density recovery technique. In this case the new Bayesian Kernel Density Estimator effectively fulfils the main aim of this research.

In examining a new density in this way the overwhelming neccessity is for the operator to be able to quickly assess the display. In looking at the Grand

Tours here the initial intention was to use a 2 dimensional projection target. However, once the software was used it quickly became apparent that a 1 dimensional target gave a display that this operator found easier to use. 3 dimensional projections using a spin plot as the viewing medium were so slow that they were useless in a high dimensional, exploratory analysis.

Chapter 8

Examples

Previous chapters evaluate three types of tool useful in the examination of posterior densities.

MCMC simulation allows the generation of a sample from a posterior density expressed mathematically. Numerous tools for this are available, both commercially and from organisations that support the copyleft system. The BUGS software written at the MRC Biostatistics Unit[1] (Lunn et al., 2000) is one.

Given high dimensional data **the Grand Tour** gives a series of views, generated in a coherent way, that can be displayed for analysis. In appendix B, R routines that implement the Grand Tour are discussed.

The Bayesian Kernel Density Estimator (BKDE) provides fast, automated density estimation for each view produced by the Grand Tour

[1]MRC Biostatistics Unit, Institute of Public Health, Robinson Way, Cambridge CB2 2SR, UK the software is available from the MRC web page at http://www.mrc-bsu.cam.ac.uk/bugs

giving a more intuitive way to evaluate each view.

In this and the following sections are some examples of the software in use.

The first figures show several steps from a tour through a five dimensional data set in which four of the dimensions are Normal and one is bimodal, the bimodality increasing from modes at $(0, 0)$ to $(0, 20)$.

Then comes a sequence where four of the dimensions are Normal and one is geometric, the geometric having parameters ranging from 0.1 to 1.

Next the BUGS system was used to produce two posterior samples. BUGS enables a user to carry out MCMC analyses in a structured way and to examine the results of that analysis.

The examples here are from two of the examples included in the BUGS distribution, *Rats* and *Surgical*. The descriptions of the problems are taken from the BUGS example documentation. BUGS own graphical output is shown for comparison. The version of BUGS is the latest available for UNIX operating systems.

8.1 Five dimensional data sets

The five dimensional data sets used in this chapter are manufactured data sets of 100 points in five dimensional space. In the first sequence the points are all $N(0, 1)$. In the next four figures the first dimension is a mixture of $N(0, 1)$ with $N(3, 1)$, $N(5, 1)$, $N(10, 1)$ and $N(20, 1)$ respectively.

In the next five sequences the first dimensions' values are drawn from a geometric distribution with parameters ranging from 0.1 to 1.

All the samples here are randomly generated using the standard generators in R or S-Plus.

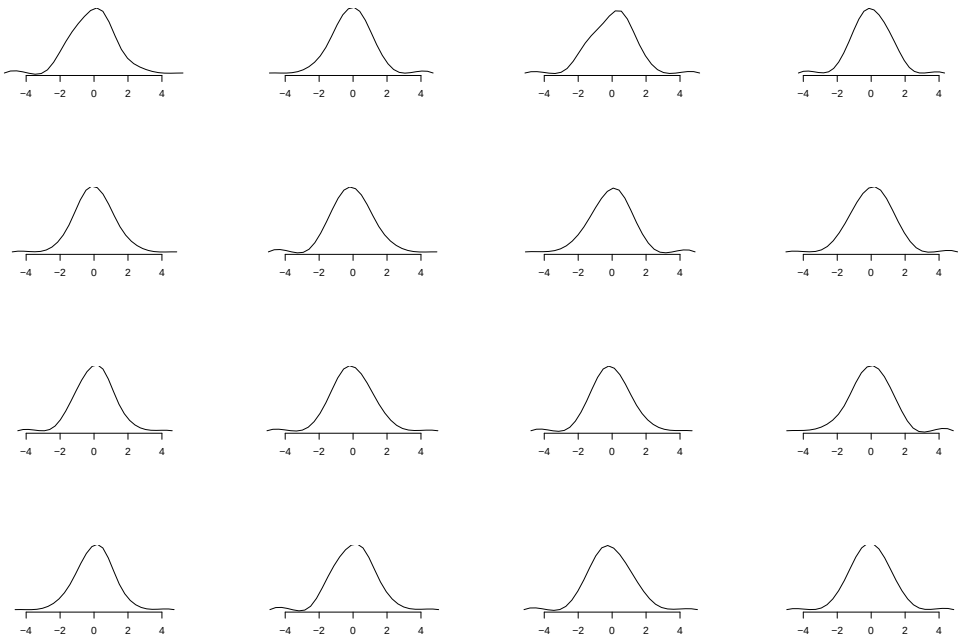


Figure 8.1: *Five dimensional data, all dimensions independently distributed* $N(0,1)$. *Axes are value, (x) and normalised density (y) (not displayed).*

The small size of the sample used to generate Figure 8.1 leaves some deviation from Normality, otherwise the sample is as expected.

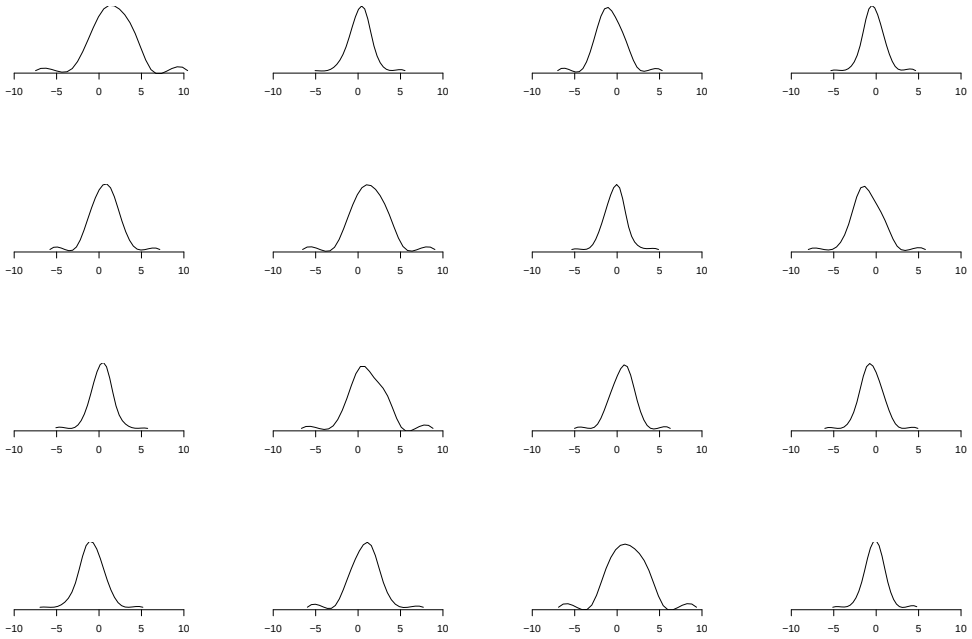


Figure 8.2: *Five dimensional data, dimensions 2 - 5 independently distributed* $N(0,1)$*, dimension 1 is a mixture of* $N(0,1)$ *and* $N(3,1)$*. Axes are value, (x) and normalised density (y) (not displayed).*

In Figure 8.2, the two Normal densities mixed for the 1st density are starting to make some of the samples look non-normal and adding to the overall width of the plot.

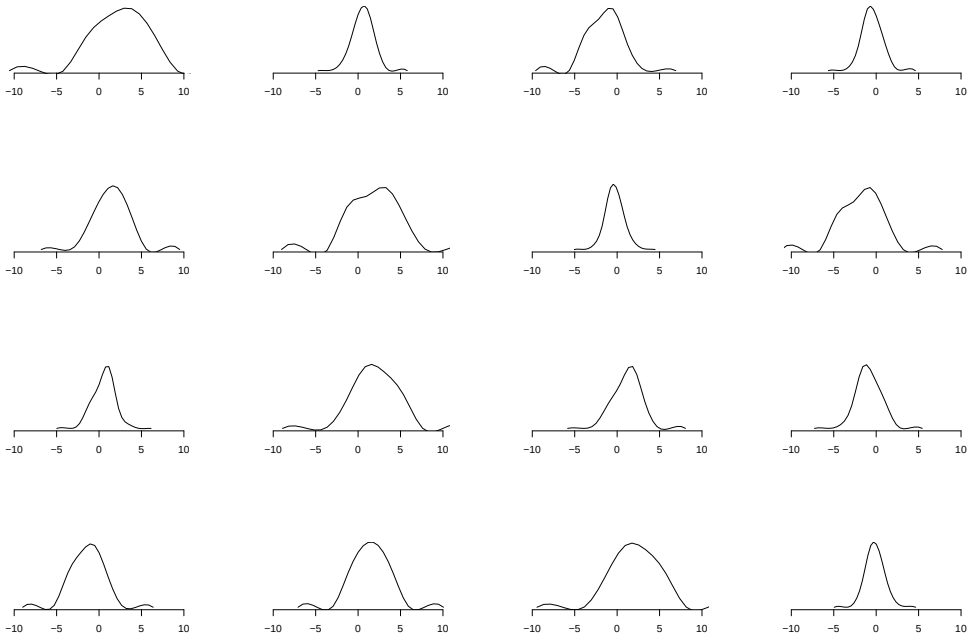


Figure 8.3: *Five dimensional data, dimensions 2 - 5 independently distributed* $N(0,1)$*, dimension 1 is a mixture of* $N(0,1)$ *and* $N(5,1)$*. Axes are value, (x) and normalised density (y) (not displayed).*

In Figure 8.3, distinct evidence of two peaks is starting to emerge in some views.

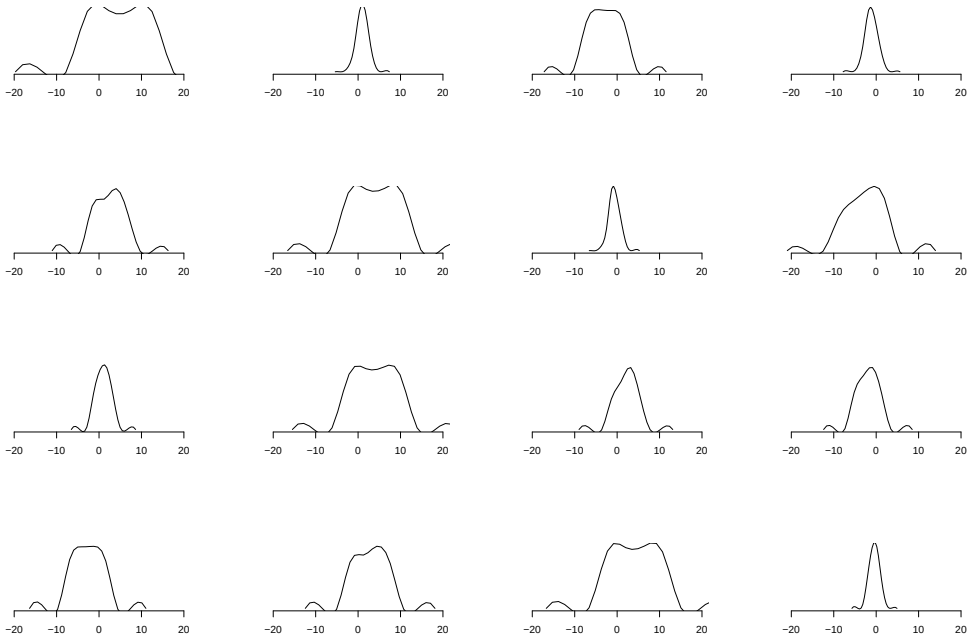


Figure 8.4: *Five dimensional data, dimensions 2 - 5 independently distributed* $N(0,1)$*, dimension 1 is a mixture of* $N(0,1)$ *and* $N(10,1)$*. Axes are value, (x) and normalised density (y) (not displayed).*

Clearly some views in Figure 8.4 show two peaks. Some of the views are still wholly Normal, this is where the rotation has moved to an angle in which the 1st dimension of the data is orthogonal to the projection plane.

Figure 8.5: *Five dimensional data, dimensions 2 - 5 independently distributed* $N(0,1)$*, dimension 1 is a mixture of* $N(0,1)$ *and* $N(20,1)$*. Axes are value, (x) and normalised density (y) (not displayed).*

As shown in Figure 8.5, there are very distinct peaks, especially in the first view that is orthogonal to dimensions 2 to 5. The purely Normal views look very thin in comparison to the width of the other views.

The exploration of a largely Normal object is typical of the outcome of Bayesian analysis, large deviations from Normal often indicating a problem with the process, see Figure 4.2 for an example.

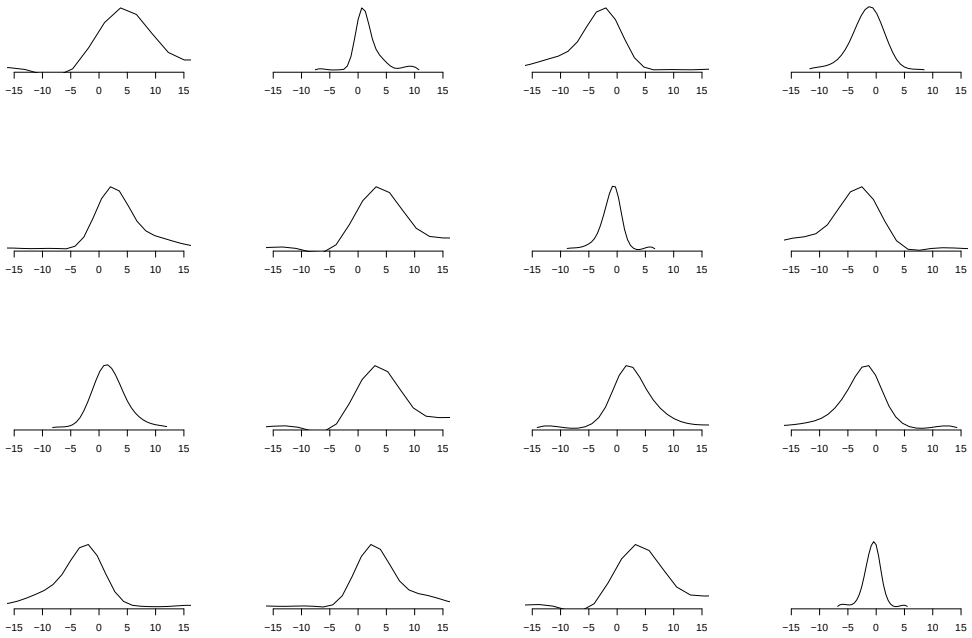


Figure 8.6: *Five dimensional data, dimensions 2 - 5 independently distributed* $N(0,1)$*, dimension 1 is Geometric* (0.1)*. Axes are value, (x) and normalised density (y) (not displayed).*

The first view in Figure 8.6 shows the clearly geometric 1st dimension and the effect that it is having on the rest of the views.

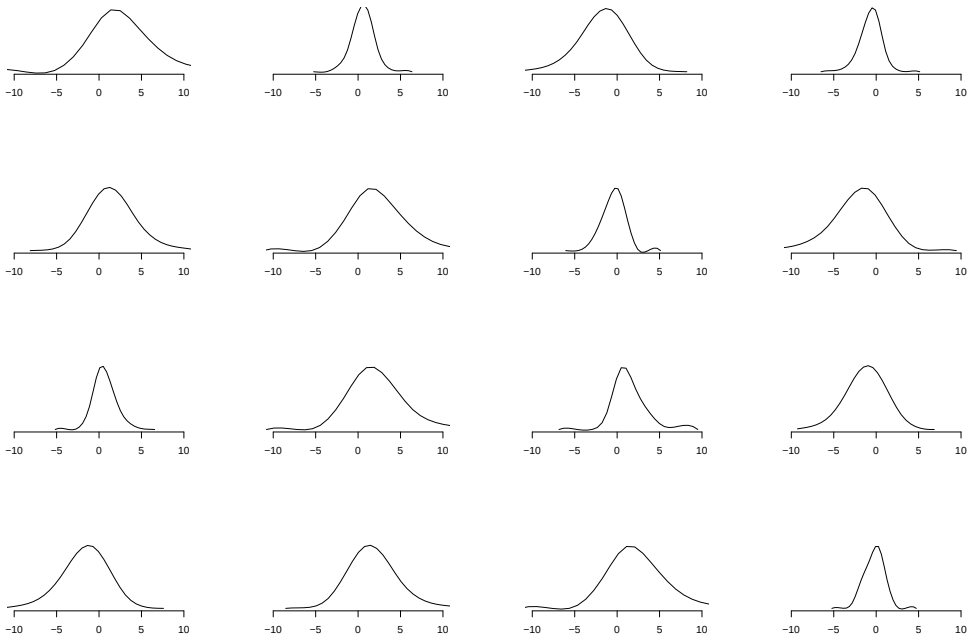


Figure 8.7: *Five dimensional data, dimensions 2 - 5 independently distributed* $N(0,1)$*, dimension 1 is Geometric* (0.25)*. Axes are value, (x) and normalised density (y) (not displayed).*

The 1st view in Figure 8.7 is still plainly geometric and showing effects on the other views.

Figure 8.8: *Five dimensional data, dimensions 2 - 5 independently distributed* $N(0,1)$*, dimension 1 is Geometric* (0.5)*. Axes are value, (x) and normalised density (y) (not displayed).*

As the parameter increases, the 1st dimension becomes thinner and more symmetric. This blends in with the four Normal dimensions and the plots in Figure 8.8 look similar to those in Figure 8.1.

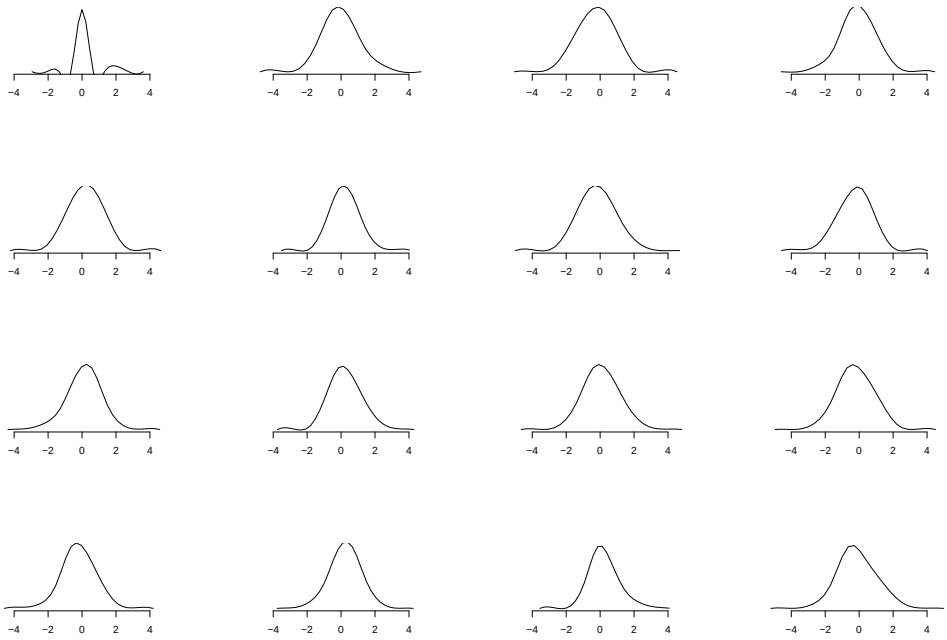


Figure 8.9: *Five dimensional data, dimensions 2 - 5 independently distributed* $N(0,1)$*, dimension 1 is Geometric* (0.75)*. Axes are value, (x) and normalised density (y) (not displayed).*

In Figure 8.9 the 1st dimension is becoming a spike and showing less and less effect on the views that are not parallel to one of the five axes.

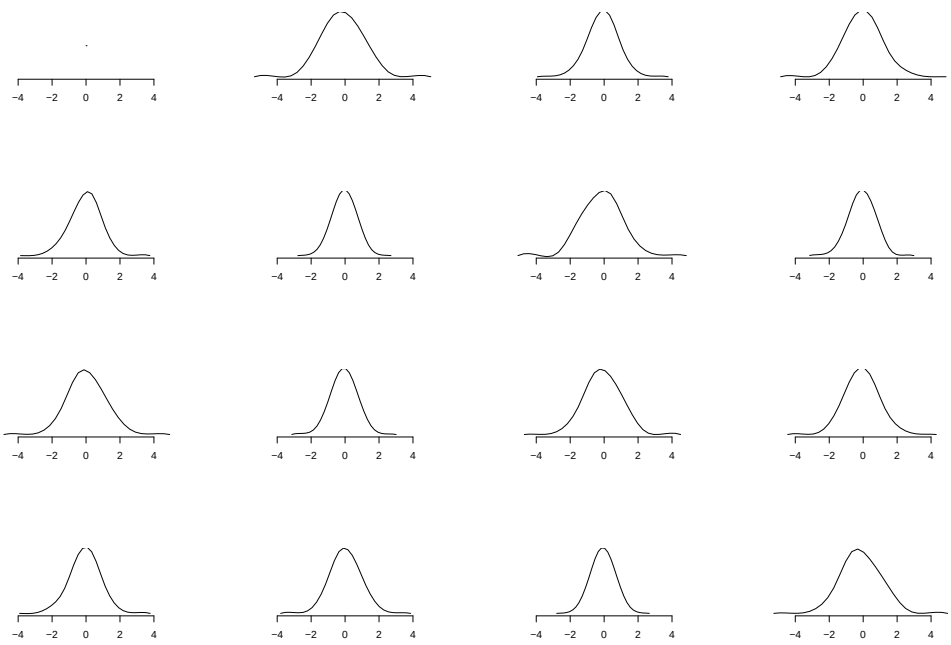


Figure 8.10: *Five dimensional data, dimensions 2 - 5 independently distributed* $N(0,1)$*, dimension 1 is Geometric* (1.0)*. Axes are value, (x) and normalised density (y) (not displayed).*

In Figure 8.10 the 1st dimension is a thin spike when compared to the other dimensions of the data which appear very similar to those in Figure 8.1.

8.2 Rats

The Rats example is taken from Gelfand et al. (1990, section 6), and concerns 30 young rats whose weights were measured weekly for five weeks. Part of the data is shown below, where Y_{ij} is the weight of the i^{th} rat measured at age x_j.

Weights Y_{ij} of rat i on day x_j

$x_j =$	8	15	22	29	36
Rat 1	151	199	246	283	320
Rat 2	145	199	249	293	354
	.....				
Rat 30	153	200	244	286	324

A plot of the 30 growth curves suggests some evidence of downward curvature.

The model is essentially a random effects linear growth curve

$$Y_{ij} \sim \text{Normal}(\alpha_i + \beta_i(x_j - x_{bar}), \tau_c) \tag{8.1}$$

$$\alpha_i \sim \text{Normal}(\alpha_c, \tau_a) \tag{8.2}$$

$$\beta_i \sim \text{Normal}(\beta_c, \tau_b) \tag{8.3}$$

where $x_{bar} = 22$, and τ represents the precision $(1/variance)$ of a Normal distribution. Note the absence of a parameter representing correlation between α_i and β_i (unlike Gelfand et al., 1990). The x_j values are standardised around their mean to reduce dependence between α_i and β_i in their likelihood: in fact, for the full balanced data, complete independence is achieved. (Note that, in general, prior independence does not force the posterior distributions to be independent.)

$\alpha_c, \tau_a, \beta_c, \tau_b$ and τ_c are given independent "noninformative" priors. Particular interest focuses on the intercept at zero time (birth), denoted $\alpha_0 = \alpha_c - \beta_c\, x_{bar}$. However, for the purposes of this demonstration, the α_i parameters are examined. From the analysis the posterior is expected to be largely Normal and so any projection from the parameter space should show Normality.

Figures 8.11 and 8.12 show the density estimations given by CODA (the BUGS analysis tool) for the 6 parameters α_1 to α_6, Figure 8.11 being the standard smooth estimate and Figure 8.12 being the coarse version derived from Silverman (1986, pp 45-47). Figure 8.13 shows the adaptive BKDE estimates for the same 6 parameters.

Figure 8.14 shows the first six steps of the 1D Grand Tour with non-adaptive BKDE (produced by the `tour1s` routine given in Appendix B), the first step being α_1 as that axis is the tour starting point. Finally Figure 8.15 shows the first six steps of a tour from the whole data set α_1 to α_{30}

As expected the adaptive BKDE estimates (Figure 8.13) and the smooth CODA estimate (Figure 8.11) show basically the same information. The non-adaptive BKDE (Figures 8.14 and 8.15) show the same basic information but with less detail.

Only the coarse estimates show a markedly different view, however, given the nature of the data and the agreement of the other four estimates, it is assumed that this should be discarded as under smoothed. Note that in this case the BKDE agrees with the smooth CODA estimate and with the expected smoothness of the data, without the necessity of choosing between the two different bandwidths.

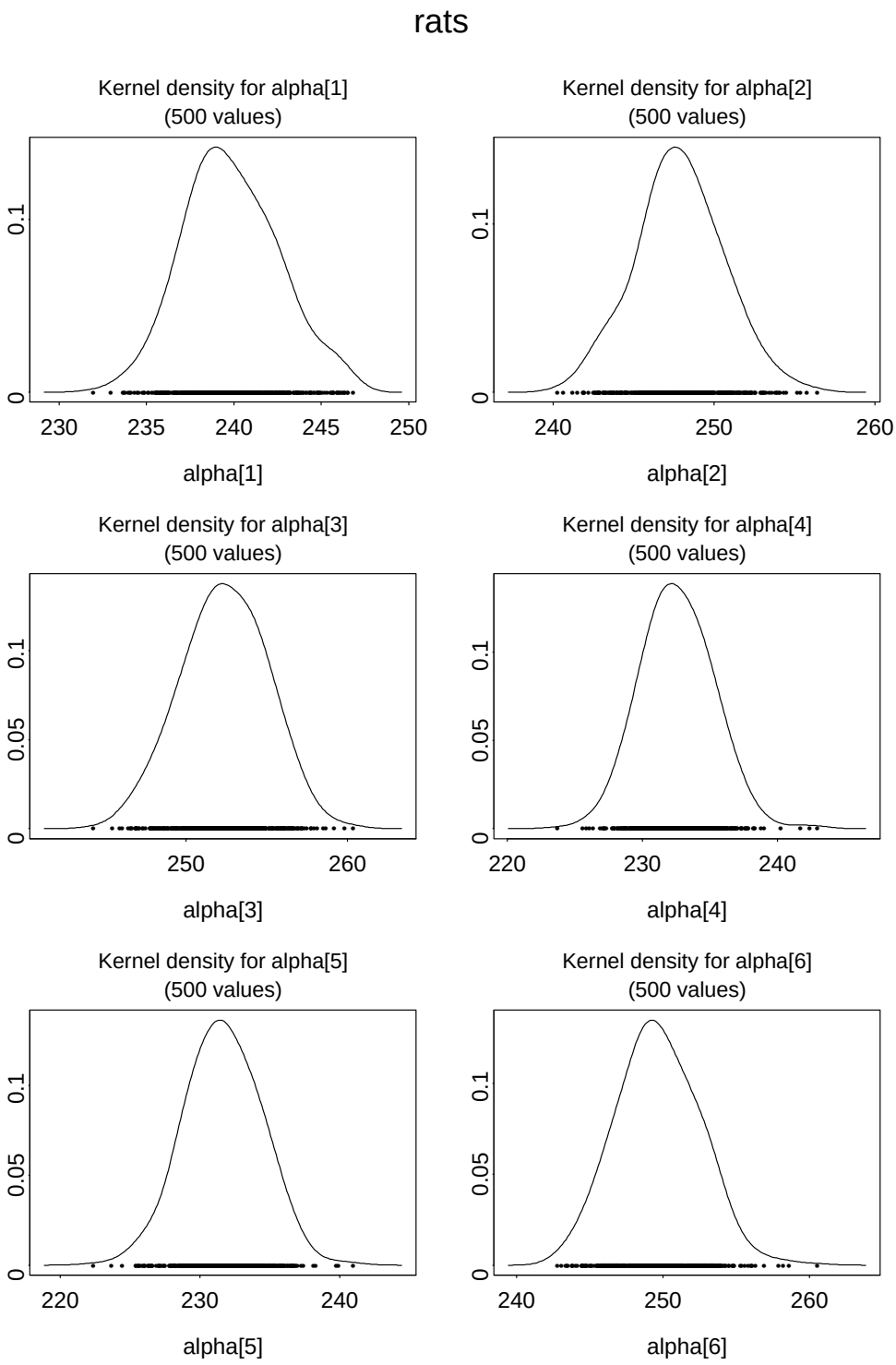


Figure 8.11: *Rats example CODA standard density estimates for α_1 to α_6. Axes are standard BUGS axes.*

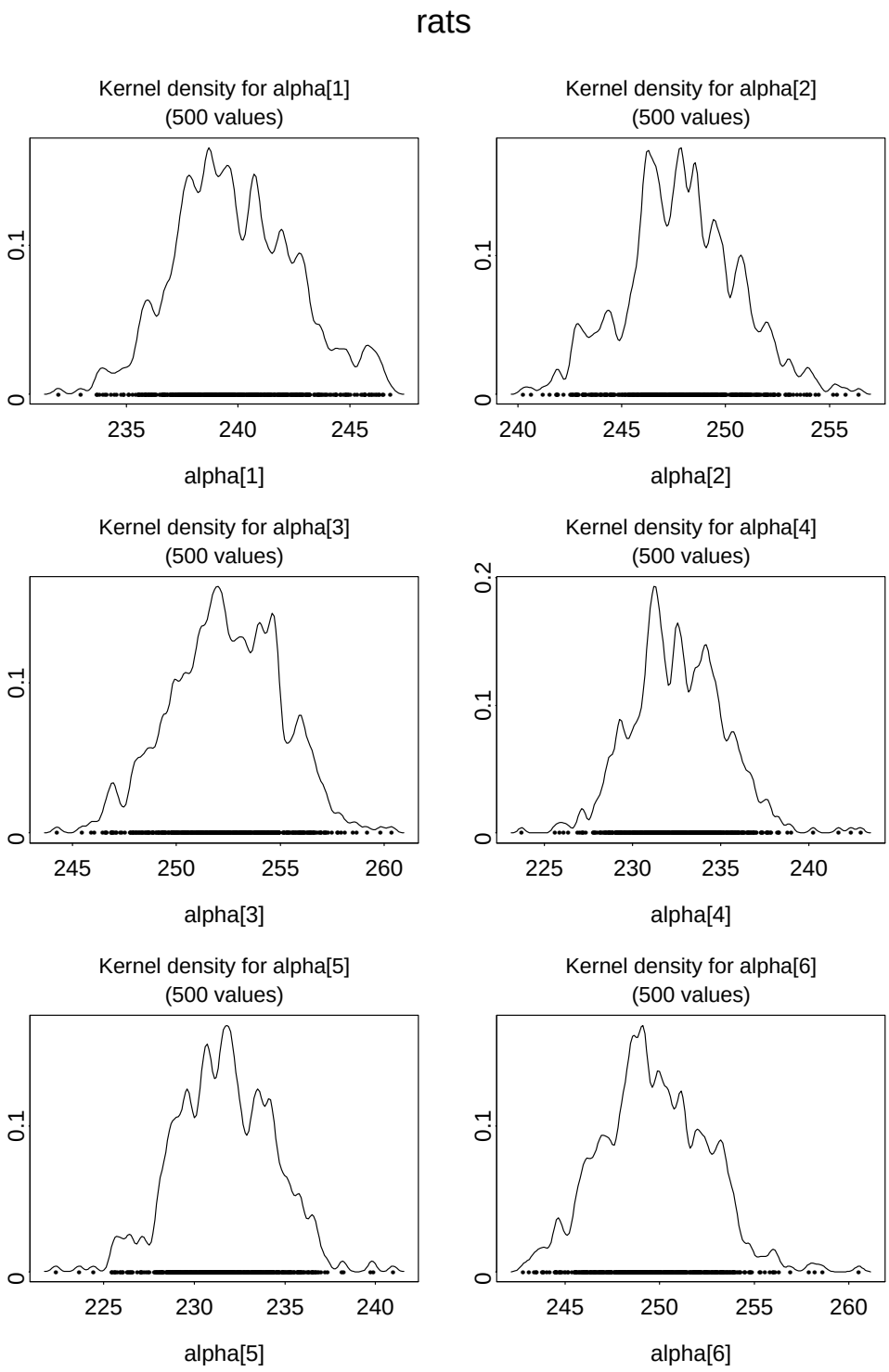


Figure 8.12: *Rats example CODA coarse density estimates for α_1 to α_6. Axes are standard BUGS axes.*

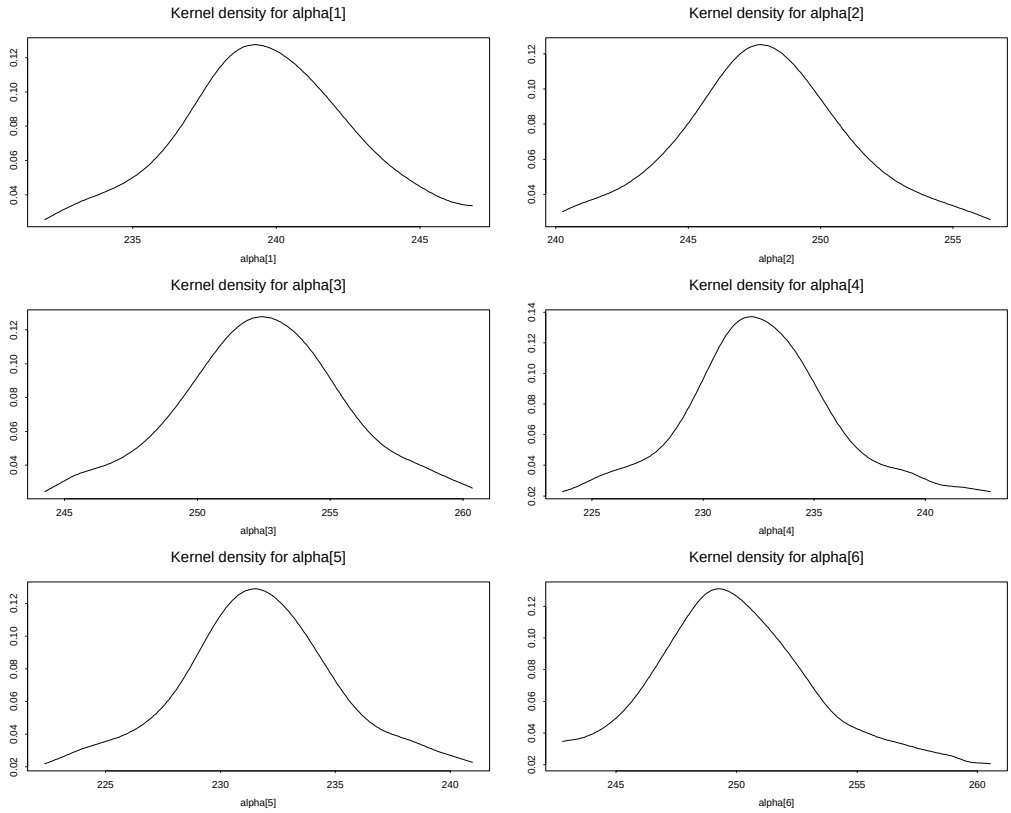


Figure 8.13: *Rats example variable bandwidth BKDE estimates for* α_1 *to* α_6*. Axes are value (x) and predictive density (y).*

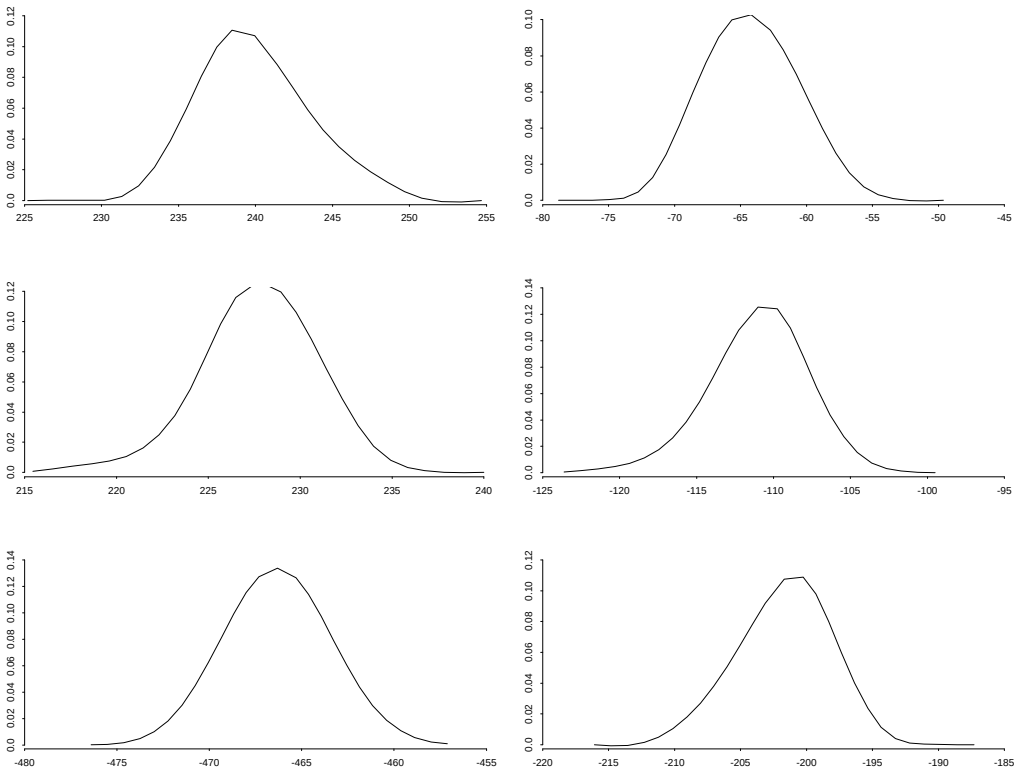


Figure 8.14: *Rats example Grand Tour with BKDE estimates for α_1 to α_6. Axes are value (x) and predictive density (y).*

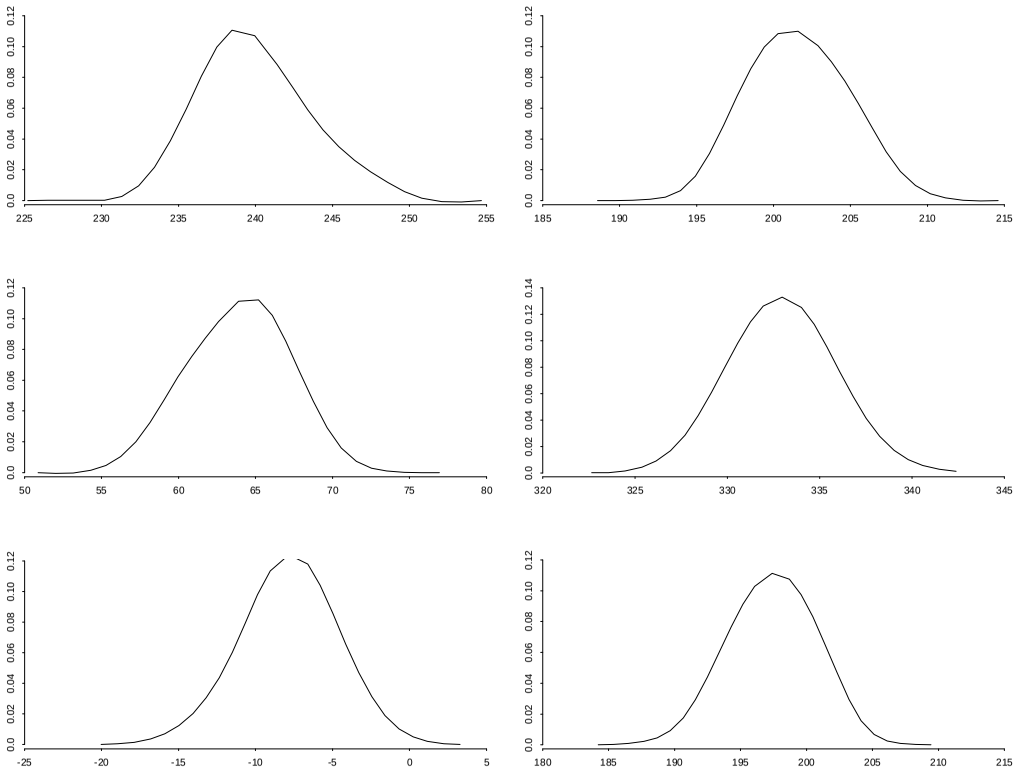


Figure 8.15: *Rats example Grand Tour with BKDE estimates for* α_1 *to* α_{30}*. Axes are value (x) and predictive density (y).*

8.3 Surgical

This example considers mortality rates in 12 hospitals performing cardiac surgery in babies. The data are shown in Figure8.1.

Hospital	Operations	Deaths
A	47	0
B	148	18
C	119	8
D	810	46
E	211	8
F	196	13
G	148	9
H	215	31
I	207	14
J	97	8
K	256	29
L	360	24

Table 8.1: *Mortality rates, 12 hospitals performing cardiac surgery in babies.*

The number of deaths r_i for hospital i are modelled as a binary response variable with 'true' failure probability p_i:

$$r_i \sim \text{Binomial}(p_i, n_i) \tag{8.4}$$

The random effects model is used and interest is in the parameters p_i for each hospital. Figures 8.16 and 8.17 again show the two different density estima-

tions given by CODA. Figure 8.18 showing the adaptive BKDE estimates. Figures 8.19 and 8.20 show the first six steps of the Grand Tour for 6 and 12 hospitals respectively. Note that the BKDE here is non-adaptive, producing a faster but less informative output.

In this case the smooth CODA estimate is similar to those produced by the non-adaptive BKDE in the Grand Tour examples (Figures 8.19 and 8.20). However, in the adaptive BKDE, Figure 8.18 there is more detail, the coarse CODA estimate (Figure 8.17) actually being more useful (or at least more in agreement with BKDE) here than the smooth estimate. Again the adaptive BKDE shows detail with no requirement to choose the bandwidth (or to choose between two different estimations).

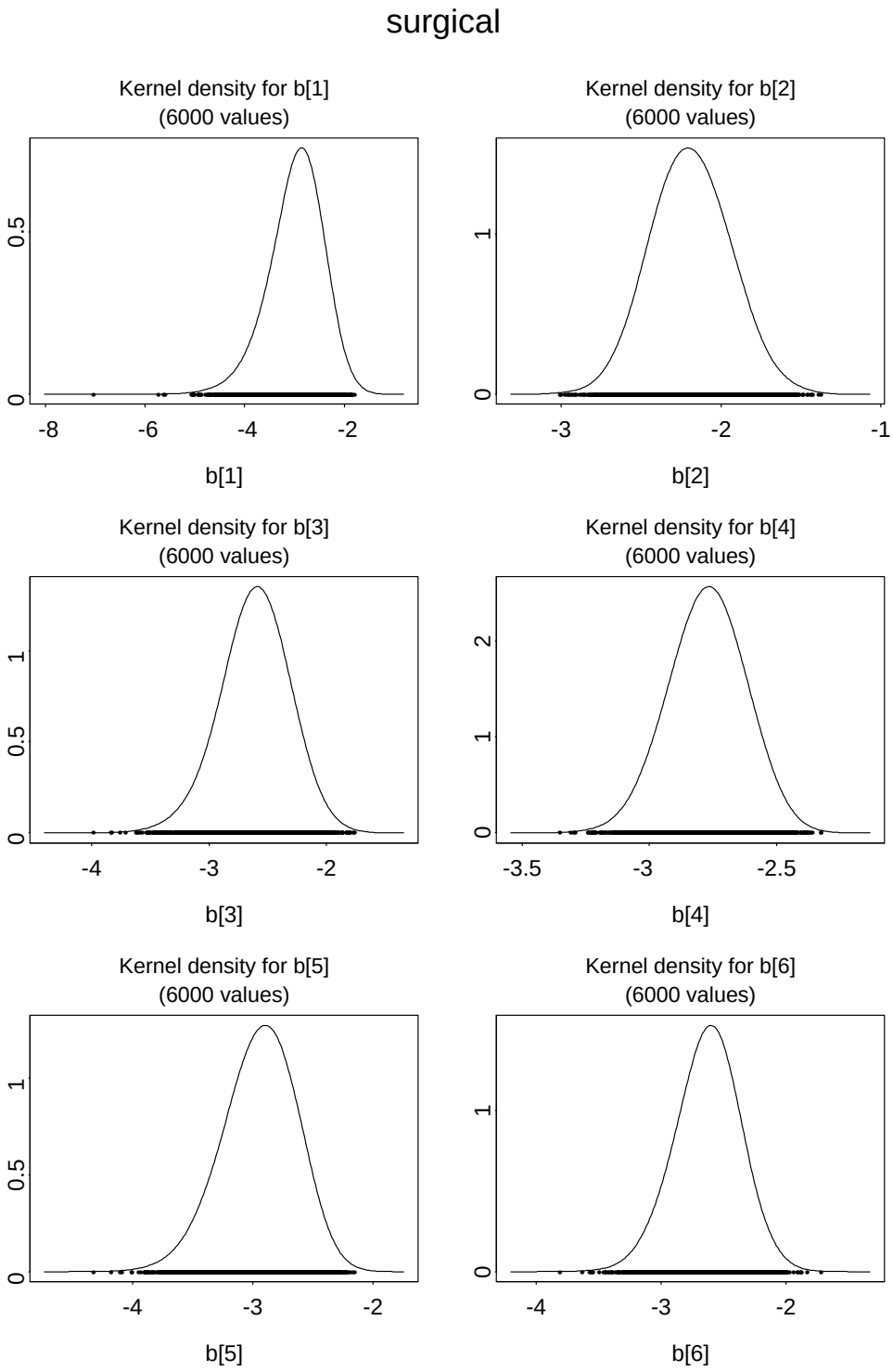


Figure 8.16: *Surgical example CODA standard density estimates for hospitals 1 to 6. Axes are standard BUGS axes.*

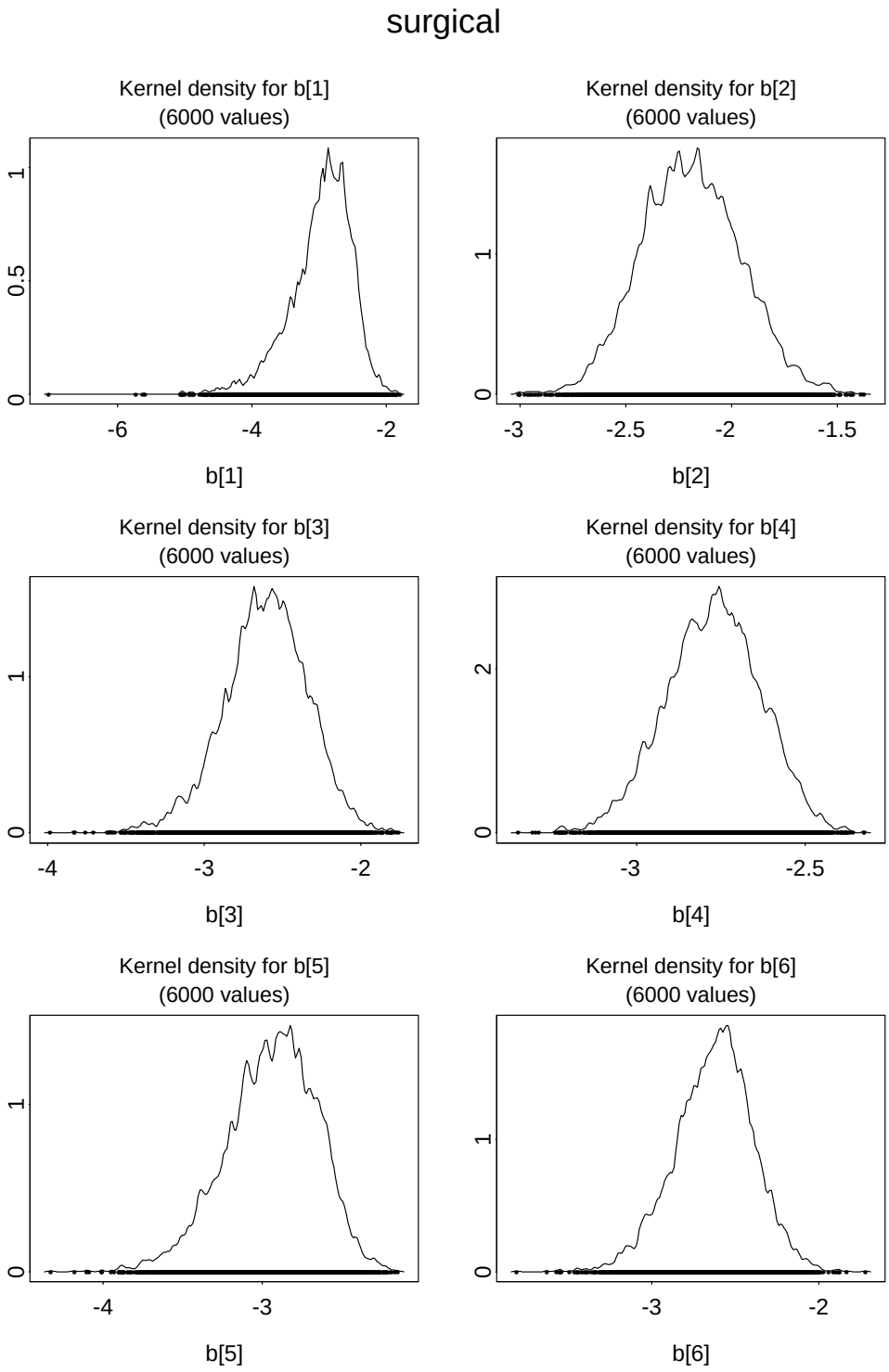


Figure 8.17: *Surgical example CODA coarse density estimates for hospitals 1 to 6. Axes are standard BUGS axes.*

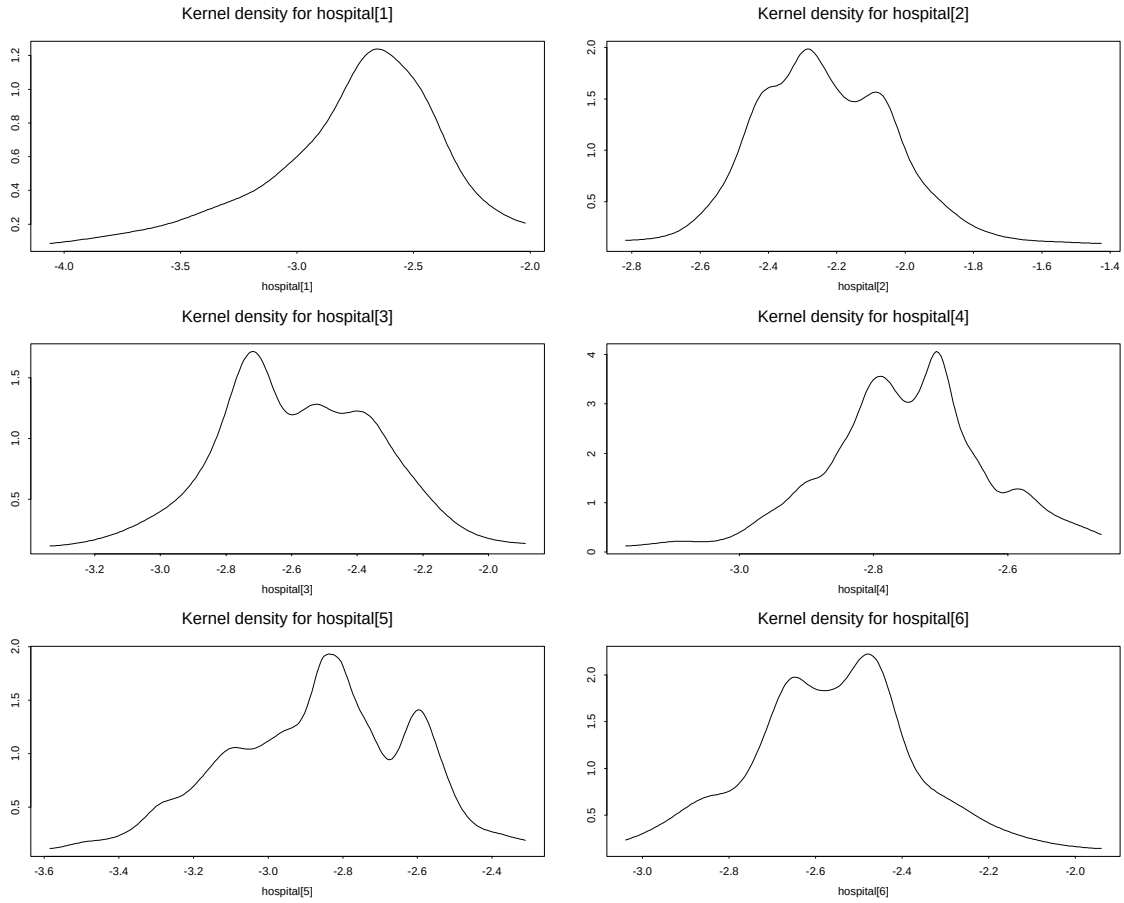


Figure 8.18: *Surgical example variable bandwidth BKDE estimates for hospitals 1 to 6. Axes are value (x) and predictive density (y).*

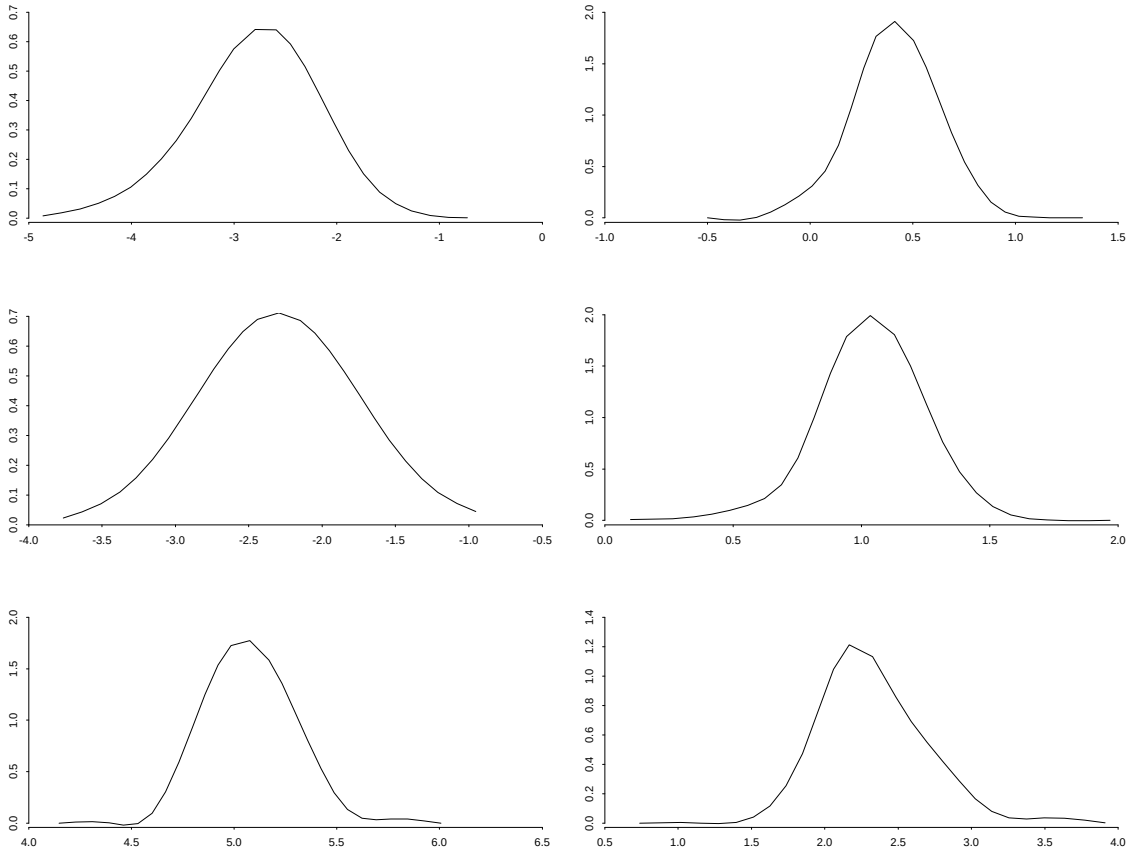


Figure 8.19: *Surgical example Grand Tour with BKDE estimates for hospitals 1 to 6. Axes are value (x) and predictive density (y).*

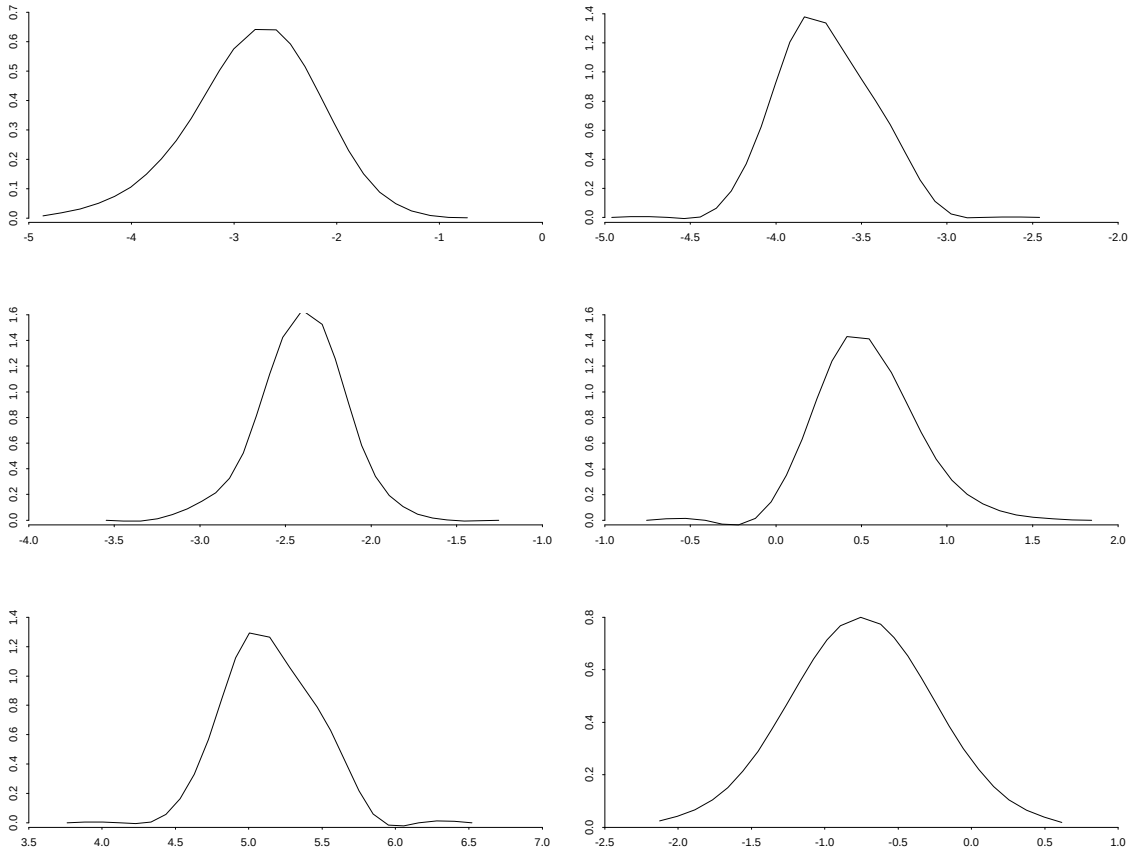


Figure 8.20: *Surgical example Grand Tour with BKDE estimates for hospitals 1 to 12. Axes are value (x) and predictive density (y).*

8.4 Summary

Comparing the plots of the Rats and the Surgical examples, the two are seen to come from two very different problems. In the case of the Rats problem the posterior distribution is expected to be multivariate Normal and the smooth estimate from CODA shows a marked similarity to those produced by both the adaptive and non-adaptive BKDE. The coarse estimate has the same general shape but is much less smooth.

In the surgical example the smooth CODA estimate still appears to be Normal, however, the adaptive BKDE and the coarse estimate show more detail, the BKDE giving a locally smoother estimate. The non-adaptive BKDE gives a similar estimate to the smooth CODA estimate.

As discussed in section 6.3 an adaptive estimate gives a better result for data with areas of both low and high density, such as the Old Faithful data. Making the bandwidth parameters part of the Bayesian formulation of the problem makes the BKDE particularly suitable for automated systems and also removes choice of bandwidth from the operator.

Chapter 9

Conclusions

This thesis presents a detailed investigation of graphical representation techniques as applied to a methodology for the examination of high dimensional objects, with case studies developed to demonstrate the use of such a methodology. It begins with summaries of Bayesian Theory and its applications, MCMC methods, display methods and the frequentist view of KDE. It then considers how KDE can be modified to provide a more satisfactory formulation, from a Bayesian perspective and to enable automated density estimation in the context of a display system such as the Grand Tour.

As this work was progressing it became apparent that many researchers, in many different disciplines, need to analyse their data in this way, but they may neither have, nor need to have, a detailed understanding of the mathematics involved. What they do need is to be able to visualise their data in some form. Hence, as part of this work, software was produced in R to provide a usable visualisation of BKDE. A large number of examples is provided to demonstrate how this software can allow easy visualisation of BKDE for a variety of types of dataset.

9.1 Overview

The initial aim of this project, driven by the interests of those involved and of the sponsoring company, Serif (UK) Ltd., was to review existing technology that could be of use in the visual representation of high dimensional functions, specifically those functions that might arise from some Bayesian examination of a problem. This aim was achieved via the combination of MCMC and the Grand Tour with the new Bayesian Kernel Density Estimation technique. These objects arise, basically, in two forms, a mathematical description of an object generated by a mathematical analysis, or increasingly, a sample from an object generated by an MCMC analysis.

Data of high dimensionality can be displayed in various ways and all of these in some way address the issue of dimension reduction. The progression from a mathematically expressed function to a sample from that function is easily handled by Monte Carlo Markov Chain (MCMC) methods. Once a sample is obtained several methods of dimensional reduction are available to either produce a summary view of the sample or, more usefully here, to produce a series of views from differing perspectives within the state space of the sample. All that is left is some way of displaying the dimensionally reduced data in a more informative way than a dot plot or line graph and the idea of recovering, and displaying, the underlying density, using kernel density estimation (KDE) seems to be an obvious approach.

Chapter 2 starts by giving a short introduction to Bayesian theory that should be enough for any reader to understand the Bayesian parts of the thesis. Chapters 3, 4, and 5 give introductions to MCMC, the Grand Tour and kernel density estimation respectively. In addition a survey of the methods and their origins is provided along with discussion of their suitability to

the work at hand.

Of the techniques available for MCMC, Gibbs sampling and Metropolis Hastings stand out for various reasons, not least that between them they provide techniques for sampling from the most common descriptive specifications for posterior densities. The conditions under which each of these is appropriate are:

Gibbs samplers given a target density specified as a complete set of conditional densities Gibbs samplers, for example that used to generate Figure 3.3, can produce a sample from the target density.

Metropolis Hastings samplers A fully specified density, for example

$$f(x,y) = (2\pi\sigma\sigma_1)^{-1} \exp\left[-\frac{1}{2}\left\{\frac{(x-\mu_1)^2}{\sigma_1^2} + \frac{(y-\mu)^2}{\sigma^2}\right\}\right].$$

allows a sample to be drawn by rejection sampling.

Combination of Gibbs and Metropolis Hastings samplers. For example given $p(x|y_1, y_2)$ and $p(y_1, y_2) = f(y_1, y_2)$ then a combination of Gibbs and Metropolis Hastings sampling provides a route to a sample. (see Gilks and Best, 1995, for a practical approach to combining samplers) .

Given a sufficiently large data set some method of viewing the sample in order to appraise it, and hence the underlying density, is needed. Projection methods (sometimes called dimension reduction methods) are more useful for this type of work, standing out for two reasons; 1) the view produced is without distortion as far as possible and 2) is a marginal projection of the data, and they lend themselves to density recovery in the form of KDE (see, for example, Asimov, 1985; Jones and Sibson, 1987).

Conventional KDE has two disadvantages, it is ad hoc in the extreme and a large amount of user input is required in choosing the bandwidth. In a situation where large numbers of projections are viewed in sequence the choice of a bandwidth for each view becomes problematical. This leads to the Bayes KDE in Chapter 6. BKDE is introduced along with a discussion of the need for an automated, accurate system for KDE, and a mathematical derivation of the new estimator is given. Both fixed and variable kernel treatments are discussed.

Examples of the BKDE being used, both for real data and constructed data, are included as well as a short discussion of the extension of the method to bivariate data. A brief discussion of the integration of the Grand Tour and BKDE is given in Chapter 7.

Several more examples are given in Chapter 8, in the form of both constructed, five dimensional, data sets, with views derived from the Grand Tour with BKDE, and two examples (the Rats example and the Surgical example) taken from WinBugs, allowing comparison of the density estimation provided by WinBugs with BKDE.

9.2 Further work and Interesting Papers

The BKDE presented here is essentially simple in concept but computationally expensive. New approaches to KDE, numerical integration or the Grand Tour will make this work more efficient and useful. A library for R incorporating some of the work referenced below should make the methods more accepted.

The BKDE relies on a system of numerical integration that is versatile but

has difficulty in dealing with high orders of variable. Many researchers have established research in the area of integration for use in Bayesian statistics. The R version of J Naylor's Gauss-Hermite functions (Naylor and Smith, 1982) needs further work to make it more universally useful. Papers such as Grey (2009) have proposed alternatives and these need investigating for alternatives to Gauss-Hermite rules.

Several authors are working on adaptive bandwidth selection, for example Ahmad and Amezzine (2007); Brewer (2000); Wand and Jones (1995) and more are working on comparisons between various KDE methods for example Assenza et al. (2008); Archambeau et al. (2006); Lian (2009). These take the comparison further than Berlinet and Devroye (1994) and could lead to insight into BKDE.

One of the problems of many KDEs is the difficulty presented by thick-tailed distributions. Bolance et al. (2008) present a transformation kernel density estimator that is suitable for heavy-tailed distributions. Alternative kernel functions such as the skew Student-t-Normal (StN) distribution, have been suggested (Barbosa et al., 2008) showing that it is a good alternative for modeling heavy-tailed data with a strong asymmetrical nature.

Various new approaches to KDE have been proposed, all are significantly different to that here, for example Duong and Hazelton (2003); Duong (2007); Gray and Moore (2003); Hazelton and Marshall (2009); Jebara et al. (2007); Ker and Ergun (2005). Comparison between them should lead to a more universal model.

Extension to more than one dimension for KDE is simple using the formulation here but leads to a large increase in computational load. A different approach to multivariate KDE is proposed by Zhang et al. (2006). KDE with

censored data is of interest, Kulasekera and Padgett (2006) as is grouped data (Lambert and Eilers, 2009), soft clustering (Lopez-Rubio and Ortiz-de Lazcano-Lobato, 2008) and density estimation on limited a range such as $[0, 1]$ (Jones and Henderson, 2007).

Approximations to Bayesian predictive distributions that do not rely on Gauss-Hermite integration could be an improvement, see for example Snelson and Ghahramani (2005). In addition, modifications to the Grand Tour by Huh and K. (2002) might lead to a more efficient formulation of the R routines used here.

All of the above lead to different KDEs that will perform differently to BKDE. Some of them have different, automated bandwidth selection. As observed in applying BKDE to the data from Berlinet and Devroye (1994) in Chapter 6,

> The overwhelming conclusion to be drawn from it is that no one KDE will do well at all densities and some experimentation with method is needed. However, the BKDE in one of its forms produced acceptable estimates of a large number of the densities without the need for human intervention. As a method KDE compares well to several others, in terms of producing a reasonable output, and can be considered at least the equal of most.

The new kdes in the above papers will perform differently to BKDE and to each other. Weither or not they are better, in some sense, depends on the use being made of them. Without producing a test such as that in Berlinet and Devroye (1994) it seems pointless to comment further.

In terms of the current work BKDE can be considered to outperform any

method without some automatic bandwidth calculation. The Bayesian formulation of BKDE gives it some advantage over other methods as far as theoretical underpinning is concerned, however, it is always possible to find a data set on which one method outperforms another. In summary, for the current work, the requirement was for a density estimator that provides a systematic procedure for bandwidth, and hence kernel, choice, BKDE readily provides that.

The variant of Projection Pursuit used here is the most basic of the many Grand Tour algorithms available. The requirements for the Grand Tour in this work were simply that it be easy to write in R, that it run on a reasonable machine and that the produced functions be easy to use. To this end, once the basic rotation matrix had been worked out and the Grand Tour run in R no further investigation into improvements was carried out. A recent overview of the Grand Tour can be found in Everitt (2005). No further comment about Projection Pursuit or The Grand Tour seems appropriate here.

Finally a new Bayesian analysis of the radiocarbon data is presented in Buck et al. (2006), which proposes a replacement for the piecewise linear formulation shown in Figure 6.3. the application of Bayes' curve estimation to these data could lead to a better understanding of radiocarbon dating and make revisiting the problem of Naylor and Smith (1988) worthwhile.

Bibliography

I. S. Abramson. On bandwidth variation in kernel estimates – a square root law. *The Annals of Statistics*, 10:1217–1223, 1982.

I. A. Ahmad and M. Amezzine. A general and fast convergent bandwidth selection method of kernel estimator. *Journal of Nonparametric Statistics.*, 19(4):165–187, 2007.

J. Aitchison and I. R. Dunsmore. *Statistical prediction analysis*. Cambridge University Press, Cambridge, 1975.

D. F. Andrews. Plots of high dimensional data. *Biometrics*, 28:125–136, 1972.

C. Archambeau, M. Valle, A. Assenza, and M. Verleysen. Assessment of probability density estimation methods: Parzen window and finite gaussian mixtures. In *Proceedings of IEEE International symposium on circuits and systems*, volume 1-11, pages 3245–3248, May 21-24 2006. Kos Isl, Greece.

E. H. Ashton, M. J. R. Healy, and S. Lipton. The descriptive use of discriminant functions in physical anthropology. *Proceedings of the Royal Society B*, 146:552–572, 1957.

D. Asimov. The grand tour - a tool for viewing multidimensional data. *SIAM Journal of Scientific, Statistical Computing*, 6(1):128–143, Jan 1985.

A. Assenza, M. Valle, and M. Verleysen. A comparative study of various probability density estimation methods for data analysis. *International journal of computational intelligence systems*, 1(2):188–201, June 2008.

C. Barbosa, R. Celso, H. Bolfarine, and J. Gomes P., Raimundo. Bayesian density estimation using skew Student-t-Normal mixtures. *Computational Statistics and Data Analysis*, 52(12):5075–5090, Aug 15 2008.

G. A. Barnard. Statistical inference. *Journal of the Royal Statistical Society B*, 11(2):115–139, 1949.

T. Bayes. An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society A*, 53:370–418, 1763.

R. A. Becker and W. S. Cleveland. Brushing scatterplots. *Technometrics*, 29:127–142, 1987.

R. H. Berk. Limiting behaviour of posterior distributions when the model is incorrect. *The Annals of Statistics*, 37:51–58, 1966.

A. Berlinet and L. Devroye. A comparison of kernel density estimates. *Publications de lInstitut de Statistique de lUniversite de Paris*, 38:3–59, 1994.

C. Bolance, M. Guillen, and J. P. Nielsen. Inverse beta transformation in kernel density estimation. *Statistics and Probability Letters*, 78(13):1757–1764, S 15 2008.

M. J. Brewer. A Bayesian model for local smoothing in kernel density estimation. *Statistics and Computing.*, 10:299–309, 2000.

S. Brooks and G. Roberts. Diagnosing convergence of mcmc algorithms. Technical report, Cambridge University, 1995.

C. E. Buck, C. D. Litton, D. G. P. Aguilar, and A. OHagan. Bayesian nonparametric estimation of the radiocarbon calibration curve. *Bayesian Analysis*, 1(2):265–288, 2006.

H. Chernoff. Using faces to represent points in k-dimensional space. *Journal of the American Statistical Association*, 79:807–822, 1973.

S. Chib and E. Greenberg. Understanding the Metropolis-Hastings algorithm. Technical report, Washington University, 1994.

R. M. Clark. A calibration curve for radiocarbon dates. *Antiquity*, 49:251–266, 1979.

W. S. Cleveland and R. McGill. The many faces of the scatterplot. *Journal of the American Statistical Association*, 79:807–822, 1984.

CRAN. The r project for statistical computing, r-faq. World wide web., June 2009. URL `http://cran.r-project.org/doc/FAQ/R-FAQ.htm`.

B. Cunliffe. Danebury: An Iron-Age hill fort in Hampshire. *Research Report 2, Council for British Archaeology, London*, 1984.

P. J. Davis and P. Rabinowitz. *Numerical integration*. Academic press Ltd., London, 2nd edition, 1984.

B. P. Dawkins. Investigating the geometry of a p-dimensional data set. Technical report, Victoria University of Wellington, Mar 1993.

B. P. Dawkins. Investigating the geometry of a p-dimensional data set. *Journal of the American Statistical Association*, 90(429):350–359, 1995.

P. Dellaportas and A. F. M. Smith. Bayesian inference for generalized linear models via Gibbs sampling. *Applied Statistics*, 42, 1993.

T. Duong. Kernel density estimation and kernel discriminant analysis for multivariate data in r. *Journal of Statistical Software*, 21(7), Oct 2007.

T. Duong and M. L. Hazelton. Plug-in bandwidth matrices for bivariate kernel density estimation. *Nonparametric Statistics*, 15(1):1730, 2003.

V. A. Epanechnikov. Nonparametric estimation of a multidimensional probability density. *Theoretical Probability, Applications*, 14:153–158, 1969.

B. S. Everitt. *An R and S-Plus Companion to Multivariate Analysis*. Springer, 2005.

W. Feller. *An introduction to probability theory and its applications*. Wiley, New York, 3rd edition, 1970.

S. Fienberg. Graphical methods in statistics. *The American Statistician*, 33: 165–178, 1979.

P. Fiorini and A. Inselberg. Configuration space representation in parallel coordinates. *Proceedings of the IEEE International Conference on Robotics and Automation*, 2:1215 – 122, 1989.

K. Fukunaga. *Introduction to statistical pattern recognition*. Academic Press, New York, Boston(Mass), 1972.

A. E. Gelfand and A. F. M. Smith. Sampling based approaches to calculating marginal densities. *Journal of the American Statistical Association*, 85 (410):398–409, 1990a.

A. E. Gelfand and A. F. M. Smith. Sampling based approaches to calculating marginal densities. *jasa*, 85(410):398–409, 1990b.

A. E. Gelfand, S. E. Hills, A. Racine-Poon, and A. F. M. Smith. Illustration of Bayesian inference in normal data models using Gibbs sampling. *Journal of the American Statistical Association*, 85(412):972–985, 1990.

S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *ieeep*, 6:721–741, 1984.

W. R. Gilks and N. G. Best. Adaptive rejection Metropolis sampling within Gibbs sampling. *Journal of the Royal Statistical Society C*, 44(4):455–472, 1995.

W. R. Gilks, S. Richardson, and D. J. Spiegelhalter (editors). *Markov Chain Monte Carlo in practice*. Chapman and Hall, London, 1996.

A. G. Gray and A. W. Moore. Rapid evaluation of multiple density models. In *In Artificial Intelligence and Statistics*, 2003.

A. Grey. Adaptive Monte Carlo methods - high-dimensional integration without Markov chains. World wide web., July 2009. URL `http://www.cs.cmu.edu/ agray/`.

G. H. Hardy and E. M. Wright. *An introduction to the theory of numbers*. Oxford University Press., Oxford, 1954.

W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrics*, 57(1):97–109, 1970.

M. L. Hazelton and J. C. Marshall. Linear boundary kernels for bivariate density estimation. *Statistics and Probability Letters*, 79(8):999–1003, Apr 15 2009.

M. Y. Huh and Kim K. Visualization of multidimensional data using modifications of the grand tour. *Journal of Applied Statistics*, 29(5):721–728, July 2002.

T. Jebara, Y. Song, and J. Thadani. Independent similarly distributed assumptions for semiparametric density estimation. Department of Computer Science, Columbia University, New York, NY 10027, USA, 2007.

M. C. Jones and D. A. Henderson. Kernel-type density estimation on the unit interval. *Biometrica*, 94(4):977–984, 2007.

M. C. Jones and R. Sibson. What is projection pursuit. *Journal of the Royal Statistical Society A*, 150(1):1–36, 1987.

M. C. Jones, J. S. Marron, and S. J. Sheather. *Progress in data-based bandwidth selection for kernel density estimation.* Chapel Hill, N.C. : Dept. of Statistics, 1992.

M. C. Jones, J. S. Marron, and S. J. Sheather. A brief survey of bandwidth selection for density estimation. *Journal of the American Statistical Association*, 91:401–407, Feb 1994.

A. P. Ker and A. T. Ergun. Empirical Bayes nonparametric kernel density estimation. *Statistics and Probability Letters*, 75(4):315–324, Dec 15 2005.

K. B. Kulasekera and W. J. Padgett. Bayes bandwidth selection in kernel density estimation with censored data. *Journal of Nonparametric Statistics*, 18(2):129–143, Feb 2006.

S. Kullback. *Information theory and statistics.* Dover Publications, London, 2nd edition, 1997.

P. Lambert and P. H. C. Eilers. Bayesian density estimation from grouped continuous data. *Computational Statistics and Data Analysis*, 53(4):1388–1399, 2009.

J. A. Lee and M. Verleysen. *Nonlinear Dimensionality Reduction*. Springer, Nov 2007.

H. Lian. Cross-validation for comparing multiple density estimation procedures. *Statistics and Probability Letters*, 79(1):112–115, Jan 1 2009.

D. V. Lindley. *Introduction to probability and statistics*, volume 2 - Inference. Cambridge University Press., Cambridge, 1965.

E. Lopez-Rubio and J. M. Ortiz-de Lazcano-Lobato. Soft clustering for non-parametric probability density function estimation. *Pattern Recognition Letters*, 29(16):2085–2091, Dec 1 2008.

D. J. Lunn, A. Thomas, N. Best, and D. Spiegelhalter. Winbugs – Bayesian modelling framework: concepts, structure, and extensibility. *Statistics and Computing*, 10:325–337, 2000.

K. V. Mardia, J. T. Kent, and J. M. Bibby. *Multivariate Analysis*. London: Academic Press, 1979.

F. H. C. Marriott and G. Eslava. Some criteria for projection pursuit. *Statistics and Computing*, 4(1):13–20, 1994.

K. T. McDonald. Volume and surface area of an n–sphere, 2003. Internet note from the Joseph Henry Laboratories, Princeton University.

N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, and A. H. Teller. Equations of state calculated by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

G. E. Moore. Cramming more components onto integrated circuits. *Electronics Magazine*, 4, 1965.

J. C. Naylor. *Some numerical aspects of Bayesian inference.* PhD thesis, University of Nottingham, October 1982.

J. C. Naylor and J. E. H. Shaw. Bayes 4 user guide. Private, The Nottingham Trent University, 1983.

J. C. Naylor and A. F. M. Smith. An archaelogical inference problem. *Journal of the American Statistical Association*, 83(4):588–595, 1988.

J. C. Naylor and A. F. M. Smith. Applications of a method for the efficient computation of posterior distributions. *Applied Statistics*, 31:214–225, 1982.

J. R. Norris. *Markov Chains.* Cambridge University Press, Cambridge, 2nd edition, 1997.

A. O'Hagan. *Kendall's advanced theory of statistics*, volume 2B - Bayesian inference. Edward Arnold., London, 1st edition, 1994.

B. U. Park, J. S. Marron, and W. C. Kim. Asymptotically best bandwidth selectors in kernel density estimation. *Statistics and Probability Letters*, 19:119–127, 1994.

C. Posse. An effective two-dimensional projection pursuit algorithm. *Communications in Statistics, Simulation and Computation*, 19(4):1143–1165, 1990.

L. J. Savage. Subjective probability and statistical practice. In *The foundations of statistical inference.*, pages 9–35. Methuen and Wiley, 1962.

D. W. Scott. *Multivariate density estimation, Theory, practice and visualisation.* Number 0-471-54770-0 in Wiley series in probability and mathematical statistics. Wiley, New York, 1992.

D. W. Scott. Averaged shifted histograms: Effective nonparametric density estimation. *The Annals of Statistics*, 13:1024–1040, 1985.

D. W. Scott. Nonparametric probability density estimation for data analysis in several dimensions. In *Proceedings of the twenty-eighth conference on the design of experiments in army research, development and testing*, pages 387–397, 1983.

D. W. Scott and G. R. Terrell. Biased and unbiased cross-validation in density estimation. *Journal of the American Statistical Association*, 82 (400):1131–1146, 1987.

J. E. H. Shaw. A quasirandom approach to integration in Bayesian statistics. Technical report, University of Nottingham, 1986.

B. W. Silverman. *Density estimation.* Chapman and Hall, London, 1986.

A. F. M. Smith. Bayesian computational methods. *Philosophical Transactions of the Royal Society A*, 337:369–386, 1991.

A. F. M. Smith and J. M. Bernardo. *Bayesian Theory.* Wiley and Sons., Chichester, 1994.

A. F. M. Smith and A. E. Gelfand. Bayesian statistics without tears: a sampling - resampling perspective. *The American Statistician*, 146(2):84–88, 1992.

A. F. M. Smith and G. O. Roberts. Bayesian computation via the Gibbs sampler and related Markov-chain and Monte-Carlo methods. *Journal of the Royal Statistical Society*, 55(1):3–23, 1993.

A. F. M. Smith, A. M. Skene, J. E. H. Shaw, and J. C. Naylor. Progress with numerical and graphical methods for practical Bayesian statistics. *The Statistician*, 36:75–82, 1987.

E. Snelson and Z. Ghahramani. Compact approximations to Bayesian predictive distributions. In *ACM International Conference Proceeding Series; Proceedings of the 22nd international conference on Machine learning*, volume 119, pages 840 – 847. ACM New York, NY, USA, 2005.

J. W. Tukey. *Exploratory data analysis*. Addison-Wesley, London, 1977.

W. N. Venables and B. D. Ripley. *Modern Applied Statistics with S*. Springer-Verlag, 2002.

M. P. Wand and M. C. Jones. *Kernel smoothing*. Chapman and Hall, London, 1995.

E. J. Wegman. Hyperdimensional data analysis using parallel coordinates. *Journal of the American Statistical Association*, 85(411):664–675, 1990.

C. Weihs. Multivariate exploratory data-analyses and graphics - a tutorial. *Journal of Chemometrics*, 7(5):305–340, 1993.

S. Weisberg. *Applied linear regression*. Wiley, Sanford, 1980.

WinBugs. The bugs project. World wide web., June 2009. URL `http://www.mrc-bsu.cam.ac.uk/bugs/`.

X. Zhang, M. L. King, and R. J. Hyndman. A Bayesian approach to bandwidth selection for multivariate kernel density estimation. *Computational Statistics and Data Analysis*, 50:3009 3031, 2006.

Appendix A

The Grand Tour in S-Plus

The following are the collection of S-Plus routines that allow the production of the Grand Tour, with the kernel density estimation and contour plotting. Examples are shown in Figures 4.1, 7.1 and 7.2.

A.1 The simple Grand Tour

This simple wrapper function uses some of the functions below and produces a simple output such as that shown in Figure 4.1.

A typical call might be

```
simpleGT(x,xlim=c(-1000,1000), ylim=c(-500,500),steps=100),
```

the only parameter that differs from those in section A.2 is the "steps" parameter, this simply defines the number of steps through n-space to be taken and displayed.

```
simpleGT(x, step = sqrt(5), steps = 10, xlim, ylim)
{
#rotate project and display latest at 20/4/95
    size <- ncol(x)
#changed for size info
    xn <- as.matrix(x)    #
#xp is the matrix that is used as the plane to project onto
# in this case 2d
    xp <- rep(0, 2 * size)
    dim(xp) <- c(2, size)
    xp[1, 1] <- 1
    xp[2, 2] <- 1
    temp <- xn    #
# centre the data by subtracting the mean
    for(i in 1:ncol(temp))
        temp[, i] <- temp[, i] - mean(temp[, i])    #
# set the limits for the display
    if(missing(xlim))
        xlim <- c(-2000, 2000)
    if(missing(ylim))
        ylim <- c(-2000, 2000)    #
# test is the matrix R - the composite rotation
    test <- rotation(size, step = step)    #
# apply the rotation/projection steps times
    for(i in 1:steps) {
#
# project and display
        xx <- t(xp %*% t(temp))
        plot(xx[, 1], xx[, 2], xlim = xlim, ylim = ylim)    #
# rotate the projection plane
        xp <- t(test %*% t(xp))
    }    #
#return the final projection plane
    xp
```

```
}
```

A.2 The full Grand Tour

The main wrapper function that is used as the user interface, a typical call to this might be

```
GT(x, xlim=c(-1000,1000), ylim=c(-500,500))
```

The parameters to the function are

1. x The file name for the data,
2. step default value sqrt(5) the rotation step size,
3. xlim default value c(-0.02, 0.02) the x-axes limits, ylim default value c(-0.02, 0.02) the x-axes limits,
4. square default value T if True display the orientation square,
5. gui default value "openlook" select the GUI, allows use of the functions on other platforms, e.g. MS Windows,
6. xp if present, specifies the start position for the axes, the function returns the final value of the axes when it exits, this allows breaking the session and starting again in the same projection,
7. grey default value F start up in Greyscale projection if True,
8. cont default value F start up in Contour plot if True.

```
GT(x, step = sqrt(5), xlim = c(-0.02, 0.02), ylim = c(-0.02, 0.02),
        square = T, gui = "openlook", xp, grey = F, cont = F)
{
# rotate project and display latest at 30/4/95
# mouse control a la spin, on windows at least
# some of this is inspired by the progdraw  library
# package, it used to rely on the two functions box and
# frame from that package - but not now 30/4/95 ;-}.
# the gui default seems a bit dodgey
#
    plotarea <- c(0.1, 0.75, 0.1, 0.75)
    indarea <- c(0.75, 0.95, 0.1,
        0.3)
    allarea <- c(0, 1, 0, 1)
    flag <- 0    #
# count the columns in the input file
    size <- ncol(x)    #
# generate an n indicator
    sq <- indicator(size)    #
# centre the sq, a better visual solution might be possible
    for(i in 1:ncol(sq[[1]])) {
        mn <- (mean(sq[[1]][, i]) + mean(sq[[2]][, i]))/2
        sq[[1]][, i] <- sq[[1]][, i] - mn
        sq[[2]][, i] <- sq[[2]][, i] - mn
    }
    xn <- as.matrix(x)    #
# xp is the matrix that is used as the plane to project onto
# in this case 2d, starting at a nice simple 1,1
# type of direction
    if(missing(xp)) {
        xp <- rep(0, 2 * size)
        dim(xp) <- c(2, size)
        xp[1, 1] <- 1
        xp[2, 2] <- 1
```

```
    }
    temp <- xn    #
# centre the data by subtracting the mean
    for(i in 1:ncol(temp)) {
        temp[, i] <- temp[, i] - mean(temp[, i])
    }
# test is the matrix R - the composite n-dimensional rotation
    test <- rotation(size, step = step)    #
# set the display
    switch(gui,
        athena = X11(),
        motif = motif(),
        openlook = openlook(),
        windows = win.graph(),
        stop("Unsupported graphical interface"))    #
# save the par settings and set up the exit cleanup
    oldpar <- par(pty = "s")
    on.exit(par(oldpar))
    assign("Draw.window", dev.cur(), where = 0)
    on.exit({
        dev.off(Draw.window)
        remove("Draw.window", where = 0)
    }
    )    #
# project the first frame and display it
    xx <- t(xp %*% t(temp))
    par(new = F, fig = c(0, 1, 0, 1), plt = plotarea)
    newpar <- par()
    if(grey) {
        xk <- kernel(xx, xx)
        image.xyz(xx[, 1], xx[, 2], xk, xlim = xlim, ylim = ylim)
    }
    else if(cont) {
        xk <- kernel(xx, xx)
```

```
        contour.xyz(xx[, 1], xx[, 2], xk, xlim = xlim, ylim = ylim)
    }
    else {
        plot(xx[, 1], xx[, 2], xlim = xlim, ylim = ylim)
    }
    par(new = F, fig = allarea, plt = indarea)
    if(square)
        indicate(xp, sq)
    par(new = F, plt = allarea)
    DrawPalette()    #
# go round looking at mouse clicks
    repeat {
# get the mouse position when clicked
        pos <- locator(1)    #
# quit
        if(pos$x > 0.9) {
            if(pos$y > 0.9) {
                return(xp)
            }
            else if(pos$y > 0.8) {
# step forward once
# rotate the axes
                xp <- t(test %*% t(xp))    #
# project and display
                xx <- t(xp %*% t(temp))
                par(new = F, plt = plotarea)
                if(grey) {
                  xk <- kernel(xx, xx)
                  image.xyz(xx[, 1], xx[, 2], xk, xlim = xlim,
                    ylim = ylim)
                }
                else if(cont) {
                  xk <- kernel(xx, xx)
                  contour.xyz(xx[, 1], xx[, 2], xk, xlim = xlim,
```

```
                    ylim = ylim)
                }
                else {
                  plot(xx[, 1], xx[, 2], xlim = xlim, ylim =
                    ylim)
                }
                par(new = F, fig = allarea, plt = indarea)
                if(square)
                  indicate(xp, sq)
                par(new = F, fig = allarea, plt = allarea)    #
# redraw the switches
                DrawPalette()
                next
            }
            else if(pos$y > 0.7) {
# step backwards once
# invert the rotation matrix if needed
                if(!flag) {
                  invtest <- solve(test)
                  flag <- 1
                }
# project and display etc, same as above
                xp <- t(invtest %*% t(xp))    #
# project and display
                xx <- t(xp %*% t(temp))
                par(new = F, plt = plotarea)
                if(grey) {
                  xk <- kernel(xx, xx)
                  image.xyz(xx[, 1], xx[, 2], xk, xlim = xlim,
                    ylim = ylim)
                }
                else if(cont) {
                  xk <- kernel(xx, xx)
                  contour.xyz(xx[, 1], xx[, 2], xk, xlim = xlim,
```

```
                    ylim = ylim)
                }
                else {
                  plot(xx[, 1], xx[, 2], xlim = xlim, ylim =
                    ylim)
                }
                par(new = F, fig = allarea, plt = indarea)
                if(square)
                  indicate(xp, sq)
                par(new = F, fig = allarea, plt = allarea)
                DrawPalette()
                next
            }
            else if(pos$y > 0.6) {
# step forward 3 times
# project and display etc, same as above 10 times
                for(i in 1:3) {
# rotate the axes
                  xp <- t(test %*% t(xp))    #
# project and display
                  xx <- t(xp %*% t(temp))
                  par(new = F, plt = plotarea)
                  if(grey) {
                    xk <- kernel(xx, xx)
                    image.xyz(xx[, 1], xx[, 2], xk, xlim = xlim,
                      ylim = ylim)
                  }
                  else if(cont) {
                    xk <- kernel(xx, xx)
                    contour.xyz(xx[, 1], xx[, 2], xk, xlim =
                      xlim, ylim = ylim)
                  }
                  else {
                    plot(xx[, 1], xx[, 2], xlim = xlim, ylim =
```

```
                    ylim)
                }
                par(new = F, fig = allarea, plt = indarea)
                if(square)
                  indicate(xp, sq)
              }
              par(new = F, fig = allarea, plt = allarea)    #
# redraw the switches
              DrawPalette()
              next
            }
            else if(pos$y > 0.5) {
# step back 10 times
# invert the rotation matrix
              if(!flag) {
                invtest <- solve(test)
                flag <- 1
              }
# project and display etc, same as above 10 times
              for(i in 1:3) {
# project and display etc, same as above
                xp <- t(invtest %*% t(xp))
# project and display
                xx <- t(xp %*% t(temp))
                par(new = F, plt = plotarea)
                if(grey) {
                  xk <- kernel(xx, xx)
                  image.xyz(xx[, 1], xx[, 2], xk, xlim = xlim,
                    ylim = ylim)
                }
                else if(cont) {
                  xk <- kernel(xx, xx)
                  contour.xyz(xx[, 1], xx[, 2], xk, xlim =
                    xlim, ylim = ylim)
```

```
                }
                else {
                  plot(xx[, 1], xx[, 2], xlim = xlim, ylim =
                    ylim)
                }
                par(new = F, fig = allarea, plt = indarea)
                if(square)
                  indicate(xp, sq)
              }
              par(new = F, fig = allarea, plt = allarea)
              DrawPalette()
              next
            }
            else if(pos$y > 0.4) {
# step forward 100 times
# project and display etc, same as above 10 times
              for(i in 1:10) {
# rotate the axes
                xp <- t(test %*% t(xp))    #
# project and display
                xx <- t(xp %*% t(temp))
                par(new = F, plt = plotarea)
                if(grey) {
                  xk <- kernel(xx, xx)
                  image.xyz(xx[, 1], xx[, 2], xk, xlim = xlim,
                    ylim = ylim)
                }
                else if(cont) {
                  xk <- kernel(xx, xx)
                  contour.xyz(xx[, 1], xx[, 2], xk, xlim =
                    xlim, ylim = ylim)
                }
                else {
                  plot(xx[, 1], xx[, 2], xlim = xlim, ylim =
```

```
                    ylim)
                }
                par(new = F, fig = allarea, plt = indarea)
                if(square)
                  indicate(xp, sq)
              }
              par(new = F, fig = allarea, plt = allarea)    #
# redraw the switches
              DrawPalette()
              next
            }
            else if(pos$y > 0.3) {
              if(grey) {
                grey <- F
              }
              else {
                grey <- T
                cont <- F
              }
            }
            else if(pos$y > 0.2) {
              if(cont) {
                cont <- F
              }
              else {
                cont <- T
                grey <- F
              }
            }
            else if(pos$y > 0.1) {
              par(new = F, plt = plotarea)
              if(grey) {
                xk <- kernel(xx, xx)
                image.xyz(xx[, 1], xx[, 2], xk, xlim = xlim,
```

```
                    ylim = ylim)
                }
                else if(cont) {
                  xk <- kernel(xx, xx)
                  contour.xyz(xx[, 1], xx[, 2], xk, xlim = xlim,
                    ylim = ylim)
                }
                else {
                  plot(xx[, 1], xx[, 2], xlim = xlim, ylim =
                    ylim)    #
                }
                par(new = F, fig = allarea, plt = indarea)
                if(square)
                  indicate(xp, sq)
                par(new = F, fig = allarea, plt = allarea)    #
# redraw the switches
                DrawPalette()
                next
            }
            else if(pos$y > 0 && pos$y < 0.1) {
# redraw without the switches
                DrawPalette(col = 0)
                dev.print()
                DrawPalette()
                next
            }
        }
#this is a test        else break
    }
}
```

The next function "rotation" produces the n-dimensional rotation matrix that is applied to the axes on each step.

The rotation matrix is derived as follows:

A rotation in 2D looks like $\begin{pmatrix} cos\ \theta & sin\ \theta \\ -sin\ \theta & cos\ \theta \end{pmatrix}$ in p dimensions it is made up of a product of matrices like

$$\begin{pmatrix} cos\ \gamma_{12} & sin\ \gamma_{12} & & & 0 \\ -sin\ \gamma_{12} & cos\ \gamma_{12} & & & \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \end{pmatrix}$$

with the general form

$$\begin{pmatrix} 1 & & & & & & & & \\ & \ddots & & & & & & & \\ & & 1 & & & & & & \\ & & & cos\ \gamma_{ij} & \dots & sin\ \gamma_{ij} & & & \\ & & & & 1 & & & & \\ & & & \vdots & \ddots & \vdots & & & \\ & & & & & 1 & & & \\ & & & -sin\ \gamma_{ij} & \dots & cos\ \gamma_{ij} & & & \\ & & & & & & 1 & & \\ & & & & & & & \ddots & \\ & & & & & & & & 1 \end{pmatrix}$$

The matrix for the total rotation is given by $R_{12}(\gamma_{12}), \dots, R_{1p}(\gamma_{1p}), R_{23}(\gamma_{23}), \dots, R_{2p}(\gamma_{2p})$ where $R_{ij}(\gamma_{ij})$ is the matrix that rotates the subspace (X_i, X_j) through the angle γ_{ij} and corresponds to the identity matrix except for the elements:

$$(i,i) = (j,j) = cos(\gamma_{ij})$$

$$(i, j) = -(j, i) = sin(\gamma_{ij})$$

Labeling $\gamma_{12}, \ldots, \gamma_{1p}, \gamma_{23}, \ldots, \gamma_{2p}$ as $\gamma_1, \ldots, \gamma_L$ with $L = 2p - 3$, the angles are given by

$$\gamma_i = STEP\sqrt{P_i}, i = 1, \ldots, L$$

where P_i is the i^{th} prime number and $STEP$ is any irrational number.

R is the product of all the R_{ij} matrices.

A sequence of rotations is obtained by repeatedly applying the matrix R with a fixed angle and projecting each result onto a plane (X_1, X_2). Marriott and Eslava Marriott and Eslava (1994) use a step of $STEP = \sqrt{5}$ which "is large enough to produce pseudo-random projections".

```
rotation(n, step = sqrt(5))
{
    p <- 2 * n - 3
    primes <- primen(p)
    rot <- diag(n)
    test <- 0
    count <- p
    for(j in n:3) {
        gammai <- step * sqrt(primes[count])
        current <- diag(n)
        current[2, 2] <- cos(gammai)
        current[j, j] <- cos(gammai)
        current[2, j] <- sin(gammai)
        current[j, 2] <-  - sin(gammai)
        rot <- current %*% rot
        count <- count - 1
    }
    for(j in n:2) {
        gammai <- step * sqrt(primes[count])
```

```
        current <- diag(n)
        current[1, 1] <- cos(gammai)
        current[j, j] <- cos(gammai)
        current[1, j] <- sin(gammai)
        current[j, 1] <-  - sin(gammai)
        rot <- current %*% rot
        count <- count - 1
    }
    rot
}
```

In this example if x is of dimension 5 then the projection matrix used is

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix}$$

It is suggested in several papers that the use of the first 1, 2 or 3 rows of the data matrix as a projection matrix is possible, however as it is convenient if the axes are orthogonal I prefer the above technique.

The next function does an extremely simple kernel density estimation on the projection. It generates h the smoothing parameter of the estimation from the variance/co-variance matrix of the projection. See Silverman [Silverman (1986)] for more details.

```
kernel(a, x, h)
{
    p <- nrow(a)
    s <- nrow(x)
    y <- rep(0, p)      #
```

```
#if there is no h specified, make one up
    if(missing(h))
        h <- 2 * sqrt(var(x)) * 1.06 * p^(-0.2)
    h <- solve(h)       #
#treat x as a scalar
    for(i in 1:p) {
        ex <- (x - a[i,  ]) %*% h
        ex <- ex * ex
        tx <- ex[, 1] + ex[, 2]
        y[i] <- sum(1 - tx[tx < 1], na.rm = T)
    }
    y
}
```

The next two functions wrap the standard S-Plus functions image and contour and combine them with `interp` that is used to obtain a regular grid for their estimations.

```
contour.xyz(x, y, z, ...)
{
    i <- interp(x, y, z)
    contour(i$x, i$y, i$z, labex = 0, ...)    #
}
image.xyz(x, y, z, ...)
{
    i <- interp(x, y, z)
    image(i$x, i$y, i$z, labex = 0, ...)    #
}
```

finally, "DrawPalette" and "indicate" simply draw the indicator and the mouse menu.

```
DrawPalette(...)
```

```
{
# Draw the palette for the drawing tools, including
# quit - to provide exit from the function:
    text(0.95, 0.95, "quit", ...)    #
# step - to advance the tour
    text(0.95, 0.85, "step", ...)    #
# back - to go back one step
    text(0.95, 0.75, "back", ...)    #
# +10 to step 10 times
    text(0.95, 0.65, "+3", ...)    #
# -10 to back 10 times
    text(0.95, 0.55, "-3", ...)    #
# +10 to step 10 times
    text(0.95, 0.45, "+10", ...)    #
# toggle greyscale
    text(0.95, 0.35, "grey", ...)    #
# toggle contour
    text(0.95, 0.25, "contour", ...)    #
# redraw
    text(0.95, 0.15, "redraw", ...)    #
# print
    text(0.95, 0.05, "Print", ...)    #
}

indicate(xp, ind)
{
# xp is a rotation matrix,
# ind is the indicator, endpoint list
    size <- nrow(ind[[1]])
    xx <- t(xp %*% t(ind[[1]]))
    plot(xx[, 1], xx[, 2], xlim = c(-1.3, 1.3), ylim = c(-1.3, 1.3), axes
         = F, xlab = "", ylab = "")
    lines(c(xx[1, 1], xx[size, 1]), c(xx[1, 2], xx[size, 2]), type = "l")
    for(i in 2:(size - 1)) {
```

```
        lines(c(xx[i, 1], xx[size, 1]), c(xx[i, 2], xx[size, 2]), type
            = "l")
    }
    points(xx[size, 1], xx[size, 2], pch = 15)
    points(xx[1, 1], xx[1, 2], pch = 15)
    xx <- t(xp %*% t(ind[[2]]))
    points(xx[, 1], xx[, 2])
    lines(c(xx[1, 1], xx[size, 1]), c(xx[1, 2], xx[size, 2]), type = "l")
    for(i in 2:(size - 1)) {
        lines(c(xx[i, 1], xx[size, 1]), c(xx[i, 2], xx[size, 2]), type = "l")
    }
}
```

Appendix B

The One and Two dimensional Tour using BKDE

B.1 The one dimensional Tour

Likelihood for the BKDE estimation of the bandwidth discussed in chapter 6 is supplied by the two functions `lik1` and `lik2`. `lik1` takes a vector of $\boldsymbol{h}$ values and uses a for loop to generate the log-likelihood of the data set with respect to each value. `kde1.1` and its associated functions use the `apply` command to split up the vector and return a vector of corresponding values.

`dest1` and `dest1.2` supply a BKDE for a data set given a value for h.

Comments within the functions give the operating instructions.

```
# lik1 version 1 by W K Kaye 4/8/99
#
# a set of functions that provide the likelihood used
# in igh to get the posterior for h
# lik1, kde1.1 and kde1.2
# and a second set, dest1 and dest1.2  that do a kde
```

```
# once h has been found
# finally rmlik1 rmoves all the above and itself
#
# parameters are:
# h bandwidth or a vector of bandwidths
# x a point
# X a data set
#
"lik1"<-function(h, X=data)
{
# main likelihood for BKDE 1D
    n<-length(X)
    dim(X)<-n
    dim(h)<-length(h)
    ret<-rep(0,length(h))
# call kde1.1 for each data element i>1
# likelihood depends on previous values
    for(i in 2:n)
        ret<-ret+log(kde1.1(h,X[i],X[1:i-1]))
    ret<-ret-max(ret)
# the kde routines work in log_n
    exp(ret)
    #ret
}
"kde1.1"<-function(h, x, X=data)
{
# apply here separates the components of the h vector
    dim(X)<-length(X)
    dim(h)<-length(h)
    ret<-rep(0,length(h))
    ret<-apply(h,1,kde1.2,x,X)
    ret
}
"kde1.2"<-function(h, x, X=data)
```

```
{
# apply here separates the components of the data vector
# note that this needs a symmetrical kernel
# and that h is actually log(h)
    dim(X)<-length(X)
    ret<-sum(apply(X,1,dnorm,x,exp(h)))
    ret
}
#density estimation once igh has been used to find h
#postm$mu should be passed to dest, note it expects the log of h
"dest1"<-function(x, h, X = data)
{
    dim(X) <- length(X)
    dim(x) <- length(x)
    ret <- apply(x, 1, dest1.2, h, X)
    ret
}
"dest1.2"<-function(x, h, X = data)
{
#note that this needs the LOG of h !!!!!!!!!
    dim(X) <- length(X)
    ret <- sum(apply(X, 1, dnorm, mean = x, sd = exp(h)))
    ret
}
"rmlik1"<-function()
{
    rm(lik1, kde1.1, kde1.2, dest1, dest1.2, rmlik1)
}
```

The functions for BKDE use the Gauss-Hermite integration functions written by Naylor (1982). The S-Plus versions of these follow for reference:

```
"igh"<- function(pdf, n = 4, postm = list(mu = 0, sd = 1), ...)
```

```
{
    xw <- ighRule(n, postm$mu, postm$sd)
    dim(xw$x) <- length(xw$x)
    f0 <- pdf(xw$x, ...)
    f0 <- f0/sum(f0 * xw$wt)
    f1 <- xw$x * f0
    postm$mu <- sum(f1 * xw$wt)
    f2 <- (xw$x - postm$mu) * f1
    postm$sd <- sqrt(sum(f2 * xw$wt))
    postm
}
".ighConst"<-
list(key = c(1, 2, 3, 5, 7, 10, 13, 17, 21, 26, 31, 37, 43, 50, 57, 65, 73, 82,
    91, 101), x = structure(.Data = c(0, 0.70710678120000003, 0,
    1.2247448714, 0.52464762330000003, 1.6506801239, 0, 0.95857246460000001,
    2.0201828704999998, 0.43607741189999999, 1.335849074,
    2.3506049736999999, 0, 0.81628788289999998, 1.6735516288000001,
    2.6519613567999998, 0.38118699020000002, 1.1571937124,
    1.9816567567000001, 2.9306374203000001, 0, 0.72355101879999995,
    1.4685532891999999, 2.2665805845000002, 3.1909932018, 0.34290132722,
    1.0366108297900001, 1.7566836493, 2.5327316742299999, 3.43615911884, 0,
    0.65680956688000003, 1.3265570844900001, 2.0259480158300001,
    2.7832900997799999, 3.66847084656, 0.31424037625000001,
    0.94778839124000003, 1.59768263515, 2.2795070805000002,
    3.0206370251200001, 3.8897248978699999, 0, 0.60576387916999996,
    1.22005503659, 1.8531076516, 2.5197356856800002, 3.2466089783699998,
    4.1013375961799996, 0.29174551066999999, 0.87871378733000005,
    1.4766827311399999, 2.0951832585100001, 2.7484707249899998,
    3.4626569335999999, 4.30444857047, 0, 0.56506958326000001,
    1.13611558521, 1.71999257519, 2.3257324861700002, 2.9671669279100001,
    3.6699503733999999, 4.4999907073100003, 0.27348104614000002,
    0.82295144913999996, 1.3802585392, 1.95178799092, 2.5462021578499998,
    3.17699916198, 3.8694479048599999, 4.6887389393100003, 0,
    0.53163300134000002, 1.06764872574, 1.61292431422, 2.1735028266700001,
```

```
2.7577629156999999, 3.3789320911399998, 4.0619466758799998,
4.8713451936699999, 0.25826775052000001, 0.77668291926999999,
1.30092085839, 1.8355316042600001, 2.38629908917, 2.9613775055299998,
3.5737690684899999, 4.24811787357, 5.0483640088700001, 0,
0.50352016342000006, 1.01036838713, 1.52417061939, 2.0492317098499999,
2.5911337897900002, 3.1578488183500002, 3.7621873519600002,
4.4285328065999998, 5.2202716905399997, 0.24534070829999999,
0.73747372855000004, 1.2340762154, 1.7385377121200001, 2.25497400209,
2.7888060584300001, 3.3478545673800002, 3.9447640401199999,
4.6036824495499999, 5.38748089001), .Dim = 110), w = c(
1.7724538482000001, 1.4611411827, 1.1816359005999999, 1.3239311752,
1.0599644828999999, 1.2402258177000001, 0.94530872050000003,
0.9865809968, 1.1814886254999999, 0.87640133440000001,
0.93558055760000003, 1.1369083327, 0.81026461760000001,
0.82868730329999996, 0.89718460020000002, 1.1013307295999999,
0.76454412869999999, 0.79289004839999999, 0.86675260659999998,
1.0719301442, 0.72023521560000003, 0.73030245270000005,
0.76460812509999998, 0.84175270150000003, 1.047003581,
0.68708185394999999, 0.70329632310000001, 0.74144193193999997,
0.82066612640000003, 1.02545169137, 0.65475928690999996,
0.66096041943999995, 0.68121188106999997, 0.72195362473000002,
0.80251686884999995, 1.00652678617, 0.62930787437000002,
0.63962123201999999, 0.66266277327000001, 0.70522036611000005,
0.78664393946, 0.98969904709000001, 0.60439318791999996,
0.60852958369999999, 0.62171605528999996, 0.64675946332000001,
0.69061803483999995, 0.77258082335, 0.97458039563999999,
0.58406169052000001, 0.59110666704000003, 0.60637973912999998,
0.63290060647000002, 0.67770675919000001, 0.75998708739999998,
0.96087870303, 0.56410030873000006, 0.56702115345000004,
0.57619335027999996, 0.59302744975999999, 0.62066260353000002,
0.66616600511000001, 0.74860736602, 0.94836897083000005,
0.54737520504000003, 0.55244195737000001, 0.56321782908999996,
0.58124727539999999, 0.60973695825999996, 0.65575567288000003,
0.73824562228000001, 0.93687449288000002, 0.53091793761999995,
```

```
    0.53307065457000002, 0.53976311390999998, 0.55177735307999998,
    0.57073929412000002, 0.59989273266999998, 0.64629170021000004,
    0.72874837058999997, 0.92625413998999995, 0.51684583648000004,
    0.52063494667999999, 0.52858944291999999, 0.54157867865999998,
    0.56127904554999997, 0.59095300346000001, 0.63763017201000005,
    0.71999338311000005, 0.91639353754999997, 0.50297488828000003,
    0.50461533135000003, 0.50967937510000005, 0.51863319370000005,
    0.53240236055000001, 0.55269462096999999, 0.58277952976000003,
    0.62965663264000005, 0.71188187434000005, 0.90719879608999998,
    0.49092150067000001, 0.49384338526999999, 0.49992087134000002,
    0.50967902712000002, 0.52408035095000005, 0.54485174236,
    0.57526244285000006, 0.62227869618999998, 0.70433296117999999,
    0.89859196144999998))
"ighRule"<-
function(n, mu = 0, sigma = 1)
{
    x <- c(rep(0, n))
    wt <- x
    i <- seq(0, ceiling(n/2) - 1)
    j <- .ighConst$key[n] + i
    left <- ceiling(n/2) - i
    right <- floor(n/2) + i + 1
    root2s <- 1.4142135618 * sigma
    x[right] <- .ighConst$x[j]
    x[left] <-  - x[right]
    wt[right] <- .ighConst$w[j]
    wt[left] <- wt[right]
    list(n = n, x = mu + root2s * x, wt = root2s * wt)
}
"rmigh" <- function()
{
    rm(igh, ighRule, .ighConst, rmigh)
}
```

Finally the `tour1` function carries out the Grand Tour with the BKDE estimation of the density.

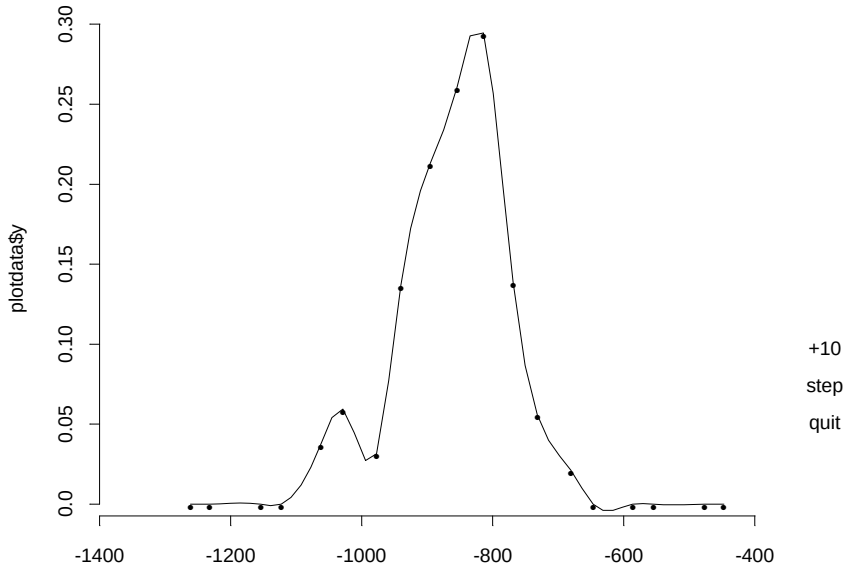


Figure B.1: *Typical screen-shot of the* `tour1` *display.*

```
"tour1"<-function(X=data, xp, m = 10, step = sqrt(5), gui = "windows")
{
# this will display projections in sequence
#
# X is data
# xp is projection matrix
# m is number of points for the plot
# step is the tour step, should be irrational
# gui is the window system being used
#
# number of dimensions
    size<-ncol(X)
# projection matrix
```

```
    if(missing(xp))
    {
        xp <- rep(0, size)
        dim(xp) <- size
        xp[1] <- 1
    }
# rotation matrix
    rot <- rotation(size, step = step)
# open a window with 16 frames
    switch(gui,
        athena = X11(),
        motif = motif(),
        openlook = openlook(),
        windows = win.graph(),
        stop("Unsupported graphical interface"))
# setup clean up at exit time
# save the par settings
    oldpar<-par()
    on.exit(
    {
        dev.off()
        par(oldpar)
    })

    plotdata<-project(rot, xp, X, m)
    par(fig=c(0,1,0,1),plt=c(0.1,0.8,0.1,0.8))
    plot(plotdata, type = "n", bty = "n", xaxs = "e", yaxs = "e")
    lines(spline(plotdata), col = 1)
    redraw <- 1
    while(redraw > 0)
    {
        redraw <- redraw - 1
        if(redraw == 0)
        {
```

```
            DP1()
            redraw<-mouseclick()
            if(redraw == 0)
                return(list(xp = xp, x = plotdata$x, y = plotdata$y))
        }
# rotate the axes
        xp <- t(rot %*% t(xp))
        plotdata<-project(rot, xp, X, m)
        par(fig=c(0,1,0,1),plt=c(0.1,0.8,0.1,0.8), new = F)
        plot(plotdata, type = "p", bty = "n", xaxs = "e", yaxs = "e")
        lines(spline(plotdata), col = 1)
    }
}
"project"<-function(rot, xp, X, m)
{
# project the data
    xx <- t(xp %*% t(X))
    dim(xx) <- length(xx)
# calculate h
    postm <- igh(lik1, X=xx)
    p1<-ighRule(m/2, mean(xx), sqrt(var(xx)))
    p2<-ighRule(m/2-1, mean(xx), sqrt(var(xx)))
    plotdata<-list(x=sort(c(p1$x,p2$x)))
    plotdata$y<-dest1(plotdata$x,postm$mu,xx)
    plotdata
}
"mouseclick"<-function()
{
    par(fig=c(0,1,0,1),plt=c(0.8,0.95,0.1,0.95))
    pos<-locator(1)
    if(pos$y > 0.7)
        ret <- 10
    else if(pos$y > 0.5)
        ret <- 1
```

```
    else
        ret <- 0
    ret
}
"DP1"<-function()
{
    par(fig=c(0,1,0,1),plt=c(0.8,0.95,0.1,0.95))
    plot(c(0,1),c(0,3),type="n",xlab="",ylab="",bty="n",axes=F)
    text(0.5,0.4,"quit")
    text(0.5,0.6,"step")
    text(0.5,0.8,"+10")
}
"tour1s"<-function(X=data, xp, n = 16, m = 10, step = sqrt(5), gui = "windows")
{
# this will display the first n projections 4 at a time
#
# X is data
# xp is projection matrix
# n is number of projections
# m is number of points for the plot
# step is the tour step, should be irrational
# gui is the window system being used
#
# number of dimensions
    size<-ncol(X)
# projection matrix
    if(missing(xp))
    {
        xp <- rep(0, size)
        dim(xp) <- size
        xp[1] <- 1
    }
# rotation matrix
    rot <- rotation(size, step = step)
```

```
# open a window with 4 frames
    switch(gui,
        athena = X11(),
        motif = motif(),
        openlook = openlook(),
        windows = win.graph(),
        stop("Unsupported graphical interface"))
    par(mfrow=c(2,2))
# setup clean up at exit time
    on.exit(
    {
        dev.off()
    })
    redraw <- n
    while(redraw > 0)
    {
# project the data
        xx <- t(xp %*% t(X))
        dim(xx) <- length(xx)
# calculate h
        postm <- igh(lik1, X=xx)
        plotdata<-ighRule(m, mean(xx), sqrt(var(xx)))
        plotdata$y<-dest(plotdata$x,postm$mu,xx)
        plot(plotdata, type = "n", ask = T, bty = "n", xaxs = "e", yaxs = "e")
        lines(spline(plotdata))
        redraw <- redraw - 1
# rotate the axes
        if(redraw > 0) xp <- t(rot %*% t(xp))
    }
    pos <- locator(1)
    list(xp = xp, x = plotdata$x, y = plotdata$y)
}
"rmtour1"<-function()
{
```

```
    rm(tour1, tour1s, DP1, mouseclick, rmtour1, project)
}
```

Appendix C

A Bayes4 implementation of Bayesian Kernel Density Estimation

In order to implement the Bayesian Kernel Density Estimation (BKDE) technique the results from chapter 6 equations (6.7) and (6.9) are needed. These are obtained using the Bayes4 system developed by Naylor and Smith (1982), (see also Naylor, 1982; Naylor and Shaw, 1983). The Bayes4 system has been ported to C++ from the original Fortran 77 version. Versions of the BKDE are given both for the original Fortran libraries and for the newer C++ versions. The older version allows better comparison with the Bayes4 manual (Naylor and Shaw, 1983).

These programs implement 1D kernel density estimation (KDE) and are the basis for the much simpler version used in S-Plus to integrate with the Grand Tour.

The documentation within the code should be sufficient to enable implementation given the Bayes4 system.

C.1 Fortran 77 libraries

These require a single, user written c program that is then compiled and linked with the Bayes4 header file and library. This code follows a set pattern and is simply modified from the sample code supplied with the Bayes4 system.

C.1.1 Fixed bandwidth KDE

This version implements a simple fixed bandwidth KDE, giving densities for both the mean and standard deviation of the final value of nh. In addition it gives a predictive density for the data on a series of points chosen to span the effective range of the density but with more points towards its 'centre'.

```
/************************************************/
/*                                              */
/* Bayes 4 analysis returning an estimate for   */
/* h the bandwidth for a KDE given a data sample. */
/* with predictive routines and data standardised */
/* with mu and sd                               */
/* Tidied up a lot                              */
/* 16/5/96 D. Kaye                              */
/*                                              */
/************************************************/

#include <stdio.h>
#include <math.h>
#include <string.h>
#include "bayes.h"

/************************************************/
/*                                              */
/* constants                                    */
```

```
/*                                                    */
/******************************************************/

#define ASIZE 5000                  /* size for input arrays*/
#define PREDNO 100                  /* number of predictive points*/

/* const for 1/root(2 pi) */
const double S2MPI = 0.3989422804 ;     /* 1 / root(2pi)*/

/******************************************************/
/*                                                    */
/* functions                                          */
/*                                                    */
/******************************************************/

/* Bayes4 - set up file pointers */
extern FILE *getfilep_( int * );
/* Normal pdf */
double K(double, double, double);
/* kde */
double kernel(float, float [], int, double);


/******************************************************/
/*                                                    */
/* global variables                                   */
/*                                                    */
/******************************************************/

char title[76];                     /* array for title*/ float
x[ASIZE],                           /* input array */
    xt[ASIZE],                      /* transformed input array */
    xpred[PREDNO];                  /* array for predicted values*/
int nx;                             /* number of input points*/
```

```
double h,
/* bandwidth*/
     mu = 0.0,                    /* prior parameters*/
     sd = 1.0,                    /* prior parameters*/
     dmu = 0.0,                   /* data parameters*/
     dsd = 0.0,                   /* data parameters*/
     dmin = 0.0,                  /* data parameters*/
     dmax = 0.0,                  /* data parameters*/
     drange = 0.0;                /* data parameters*/
FILE *sout, *pin;                 /* io channels for ip and log*/

/************************************************/
/*                                              */
/* standard Bayes4 main                         */
/*                                              */
/************************************************/

main()
{
    f_init();
    bayld_();
    bayes_();
    bayend_();
    f_exit();
}

/************************************************/
/*                                              */
/* set up file handles - standard Bayes4        */
/*                                              */
/************************************************/

void bfgpio_c(FILE **ppin, FILE **psout)
{
```

```
    int kpin, ksout;
    bfgpio_(&kpin, &ksout);
    *psout = getfilep_(&ksout);
    *ppin = getfilep_(&kpin);
}

/*************************************************/
/*                                               */
/* required by Bayes4                            */
/* set up the problem                            */
/*                                               */
/*************************************************/

void probld_(char *vname, int *npar, int lname)
{
    int i = 0;
    double x2sum = 0.0, xsum = 0.0, u;

    /* set up file pointers */
    bfgpio_c(&pin, &sout);

    /* initialise range variables */
    fscanf(pin,"%f ", &x[0]);
    dmin = x[0];
    dmax = x[0];
    dmu = x[0];

    /* input data and find range etc */
    for ( nx=1; fscanf(pin,"%f ", &x[nx])==1; nx++)
    {
        if(x[nx] < dmin)
            dmin = x[nx];
        if(x[nx] > dmax)
            dmax = x[nx];
```

```
        u = x[nx] - dmu;
        xsum += u;
        x2sum += u * u;
    }
    dmu += xsum / nx;
    dsd = sqrt(1.0 / (nx - 1) * (x2sum - xsum * xsum / nx));

    /* output count and mean etc */
    printf(" %d items read: (%7.2f) ... (%7.2f)\n",
        nx, x[0],
        x[nx-1]);
    printf("sample mean (%7.4f), sample sd (%7.4f)\n", dmu, dsd);

    /*normalise */
    for(i = 0; i < nx; i++)
    {
        xt[i] = (x[i] - dmu) / dsd;
    }
    drange = dmax - dmin;

    printf("\n");

    *npar = 1;
    (void) strncpy(&vname[0],"h-width",lname);
}

/************************************************/
/*                                              */
/* required by Bayes4                           */
/* inverse transforms                           */
/*                                              */
/* in this case h must be positive so we work   */
/* with log(h)                                  */
/*                                              */
```

```
/************************************************/

void btftrn_(float *theta, int *npar, float *rcon)
{
    h = exp(theta[0]);
    *rcon = 1;
}

/************************************************/
/*                                              */
/* normal density at point xx, N(u,s)           */
/*                                              */
/*                                              */
/* the kernel                                   */
/*                                              */
/************************************************/

double K(double xx, double u, double s)
{
    double z = (xx - u) / s;
    return S2MPI / s * exp(-0.5 * z * z);
}

/************************************************/
/*                                              */
/* KDE                                          */
/* at point xi, using set of n points X and     */
/* normal kernel N(X[i],hh)                     */
/*                                              */
/************************************************/

double kernel(float xi, float X[], int n, double hh)
{
    int i = 0;
```

```
    double sum = 0.0;

    for(i = 0; i < n; i++)
    {
        sum += K(xi, X[i], hh);
    }
    sum /= n;
    return sum;
}

/*************************************************/
/*                                               */
/* prior, required by Bayes4                     */
/* input parameter vector and number             */
/* return prior for theta, default = 1           */
/*                                               */
/* the normal prior allows expressing belief in  */
/* the smoothness of the density, mu is          */
/* smoothness sd is strength of belief           */
/*                                               */
/*************************************************/

float prior_(float *theta, int *npar)
{
        return K(theta[0], mu, sd);
        /*return 1;*/
}

/*************************************************/
/*                                               */
/* log likelihood of data                        */
/* uses answer to rturn log likelihood of data   */
/* given theta and number of parameters          */
/*                                               */
```

```
/************************************************/

void loglik_(float *theta, int *npar, float *answer, int *ok)
{
    int i, j;
    double sum = 0.0;
    double lik = 0.0;

    for (i = 1; i < nx; i++)
    {
    lik = lik + log(kernel(xt[i], xt, i, h));
    }
    *answer = lik;
}

/************************************************/
/*                                              */
/* all the following are optional for predictive  */
/* or special function analysis The default       */
/* versions of these functions are useful, so     */
/* leave as here only uncomment the next line     */
/* to define some specific predictive analysis    */
/*                                              */
/************************************************/
#define PREDICTIVE_FUNCTIONS
#ifdef PREDICTIVE_FUNCTIONS

/************************************************/
/*                                              */
/* required for Bayes4 special functions          */
/* output routine                               */
/*                                              */
/************************************************/
```

```
double bxval_c(int j)
{
    extern float bxval_();
    float x;
    int tmp = j+1;
    x =  bxval_(&tmp);
    return( x );
}

/*************************************************/
/*                                               */
/* required for Bayes4 special functions         */
/* initialise special function system            */
/*                                               */
/*************************************************/

void bxinit_(int *nofun)
{
    /* step for the alternative to bcher */
    double step = drange / (PREDNO - 1);
    int i = 0;

    *nofun = PREDNO;

    /* set up the xpred values */
    xpred[0] = dmin;
    for(i = 1; i < PREDNO; i++)
    {
        xpred[i] = xpred[i-1] + step;
    }
}

/*************************************************/
/*                                               */
```

```
/* required for Bayes4 special functions          */
/* compute gi                                     */
/*                                                */
/************************************************/

void bxfun_(float *theta, int *npar, float *funs, int *nofun)
{
    int i;

    h *= dsd;

    for (i = 0; i < PREDNO; i++)
    {
    funs[i] = kernel(xpred[i], x, nx, h);
    }
}

/************************************************/
/*                                                */
/* required for Bayes4 special functions          */
/* output routine                                 */
/*                                                */
/************************************************/

void bxout_(int *pnfun)
{
    int i;
    float density[PREDNO];
    float max = 0;
    int temp = PREDNO;

    char title[] = "predx";
    for(i = 0;  i < PREDNO; i++)
    {
```

```
        density[i] = bxval_c(i);
        if (density[i] > max) max = density[i];
    }
    fprintf(stdout,"Predictive distribution for X\n");
    boden1_(xpred, density, &temp, &max, title, 8);
}

#endif
```

C.1.2 Variable bandwidth KDE

This version implements a variable bandwidth KDE in which bandwidth depends on an inverse relationship with a simple fixed bandwidth KDE. It gives a predictive density for the data on a series of points chosen to span the effective range of the density but with more points towards its 'centre'.

```
/*************************************************/
/*                                               */
/* Bayes 4 analysis returning an estimate for    */
/* h the bandwidth for a KDE given a data sample. */
/* with predictive routines and data standardised */
/* with mu and sd using variable kernel          */
/* \propto 1/sqrt(\hat f) as a second parameter   */
/* \hat f used in a simple non adaptive kernel   */
/* Tidied up a lot                               */
/* 15/5/96 D. Kaye                               */
/*                                               */
/*************************************************/


#include <stdio.h>
#include <math.h>
#include <string.h>
```

```
#include "bayes.h"                  /* local version! */

/************************************************/
/*                                              */
/* constants                                    */
/*                                              */
/************************************************/
#define ASIZE 5000                  /* size for input arrays               */
#define PREDNO 50                   /* number of predictive points         */

/************************************************/
/*                                              */
/* functions                                    */
/*                                              */
/************************************************/

/* Bayes4 - set up file pointers */
extern FILE *getfilep_( int * );
/* Normal pdf */
double K(double, double, double);
/* kde */
double kernel(float, float [], int, double);

/************************************************/
/*                                              */
/* global variables                             */
/*                                              */
/************************************************/

char title[76];                     /* array for title*/
float x[ASIZE],                     /* input array*/
     xt[ASIZE],                     /* transformed input array      */
     xpred[PREDNO],                 /* array for predicted values   */
     hf,                            /* fixed h                      */
```

```
     hv;                              /* variable h               */
int nx;                               /* number of input points*/
double mu = 0.0,                      /* prior mean*/
     sd = 1.0,                        /* prior sd                 */
     lh = 0.0,                        /* prior for hf             */
     lsd = 1.0,                       /* prior for hf             */
     dmu = 0.0,                       /* data mean                */
     dsd = 0.0,                       /* data parameters              */
     dmin = 0.0,                      /* data minimum             */
     dmax = 0.0,                      /* data maximum             */
     drange = 0.0;                    /* data range               */
FILE *sout, *pin;                     /* io channels for files*/

/* const for 1/root(2 pi) */
const double S2MPI = 0.3989422804 ;

/*************************************************/
/*                                               */
/* standard Bayes4 main                          */
/*                                               */
/*************************************************/

main()
{
    f_init();
    bayld_();
    bayes_();
    bayend_();
    f_exit();
}

/*************************************************/
/*                                               */
/* set up file handles - standard Bayes4         */
```

```
/*                                                    */
/******************************************************/

void bfgpio_c(FILE **ppin, FILE **psout)
{
    int kpin, ksout;
    bfgpio_(&kpin, &ksout);
    *psout = getfilep_(&ksout);
    *ppin = getfilep_(&kpin);
}

/******************************************************/
/*                                                    */
/* required by Bayes4                                 */
/* set up the problem                                 */
/*                                                    */
/******************************************************/

void probld_(char *vname, int *npar, int lname)
{
    int i = 0;
    double x2sum = 0.0, xsum = 0.0, u;

    /* set up file pointers */
    bfgpio_c(&pin, &sout);

    /*printf("\ninput - mean sd mean sd\n");
    scanf("%f %f %f %f", &mu, &sd, &lh, &lsd);*/

    /* initialise range variables */
    fscanf(pin,"%f ", &x[0]);
    dmin = x[0];
    dmax = x[0];
    dmu = x[0];
```

```
    /* input data and find range etc */
    for ( nx=1; fscanf(pin,"%f ", &x[nx])==1; nx++)
    {
        if(x[nx] < dmin)
            dmin = x[nx];
        else if(x[nx] > dmax)
            dmax = x[nx];
        u = x[nx] - dmu;
        xsum += u;
        x2sum += u * u;
    }
    dmu += xsum / nx;
    dsd = sqrt(1.0 / (nx - 1) * (x2sum - xsum * xsum / nx));

    /* output count and mean etc */
    printf(" %d items read: (%7.2f) ... (%7.2f)\n",
        nx, x[0],
        x[nx-1]);
    printf("sample mean (%7.4f), sample sd (%7.4f)\n", dmu, dsd);

    /* normalise */
    for(i = 0; i < nx; i++)
    {
        xt[i] = (x[i] - dmu) / dsd;
    }
    drange = dmax - dmin;

    printf("\n");

    *npar = 2;
    (void) strncpy(&vname[0],"h-var",lname);
    (void) strncpy(&vname[8],"h-fix",lname);
}
```

```
/************************************************/
/*                                              */
/* required by Bayes4                           */
/* inverse transforms                           */
/*                                              */
/* in this case h must be positive so we work   */
/* with log(h)                                  */
/*                                              */
/************************************************/

void btftrn_(float *theta, int *npar, float *rcon)
{
    hv = exp(theta[0]);
    hf = exp(theta[1]);
    *rcon = 1;
}

/************************************************/
/*                                              */
/* normal                                       */
/* density at point xx, N(u,s)                  */
/*                                              */
/************************************************/

double K(double xx, double u, double s)
{
    double z = (xx - u) / s;
    return S2MPI / s * exp(-0.5 * z * z);
}

/************************************************/
/*                                              */
/* KDE                                          */
```

```
/* at point xi, using set of n points X and       */
/* normal kernel N(X[i],hh)                       */
/*                                                */
/************************************************/

double kernel(float xi, float X[], int n, double hh)
{
    int i = 0;
    double sum = 0.0;

    for(i = 0; i < n; i++)
    {
        sum += K(xi, X[i], hh);
    }
    sum /= n;
    return sum;
}

/************************************************/
/*                                                */
/* prior, required by Bayes4                      */
/* input parameter vector and number              */
/* return prior for theta, default = 1            */
/*                                                */
/* the normal prior allows expressing belief in   */
/* the smoothness of the density, mu is           */
/* smoothness sd is strength of belief            */
/*                                                */
/************************************************/

float prior_(float *theta, int *npar)
{
        return K(theta[0],mu,sd)*K(theta[1],lh,lsd);
        /*return 1;*/
```

```
}

/************************************************/
/*                                              */
/* log likelihood of data                       */
/* uses answer to rturn log likelihood of data  */
/* given theta and number of parameters         */
/*                                              */
/************************************************/

void loglik_(float *theta, int *npar, float *answer, int *ok)
{
    int i;
    double lik = 0.0;

    for (i = 2; i < nx; i++)
    {
        lik += log(kernel(xt[i], xt, i,
            hv/sqrt(kernel(xt[i], xt, i, hf))));
    }
    *answer = lik;
}

/*****************************************************************************/
/************************************************/
/*                                              */
/* all the following are optional for predictive  */
/* or special function analysis The default       */
/* versions of these functions are useful, so     */
/* leave as here only un-comment the next line    */
/* to define some specific predictive analysis    */
/*                                              */
/************************************************/
#define PREDICTIVE_FUNCTIONS
```

```
#ifdef PREDICTIVE_FUNCTIONS

/************************************************/
/*                                              */
/* required for Bayes4 special functions        */
/* output routine                               */
/*                                              */
/************************************************/

double bxval_c(int j)
{
    extern float bxval_();
    float x;
    int tmp = j+1;
    x =  bxval_(&tmp);
    return( x );
}

/************************************************/
/*                                              */
/* required for Bayes4 special functions        */
/* initialise special function system           */
/*                                              */
/************************************************/

void bxinit_(int *nofun)
{
    /* step for the alternative to bcher */
    double step = drange / (PREDNO - 1);
    int i = 0;

    *nofun = PREDNO;

    /* set up the xpred values */
```

```
    xpred[0] = dmin;
    for(i = 1; i < PREDNO; i++)
    {
        xpred[i] = xpred[i-1] + step;
    }
}

/************************************************/
/*                                              */
/* required for Bayes4 special functions        */
/* compute gi                                   */
/*                                              */
/************************************************/

void bxfun_(float *theta, int *npar, float *funs, int *nofun)
{
    int i;

    hv *= dsd;
    hf *= dsd;

    for (i = 0; i < PREDNO; i++)
    {
        funs[i] = kernel(xpred[i], x, nx,
            hv/sqrt(kernel(xpred[i], x, nx, hf)));
    }
}

/************************************************/
/*                                              */
/* required for Bayes4 special functions        */
/* output routine                               */
/*                                              */
/************************************************/
```

```
void bxout_(int *pnfun)
{
    int i;
    float density[PREDNO];
    float max = 0;
    int temp = PREDNO;

    char title[] = "predx";
    for(i = 0;  i < PREDNO; i++)
    {
        density[i] = bxval_c(i);
        if (density[i] > max) max = density[i];
    }
    fprintf(stdout,"Predictive distribution for X\n");
    boden1_(xpred, density, &temp, &max, title, 8);

}

#endif
```

C.2 C++ libraries

The C++ libraries were converted from the original Fortran 77 using a mechanical translator. The requirement is for a header file (for example bkde.h) containing class definitions specific to the problem and an accompanying c++ file (for example bkde.cc) containing the function definitions for the class as well as the main function for the program.

C.2.1 Fixed bandwidth KDE

This version implements a simple fixed bandwidth KDE, giving densities for both the mean and standard deviation of the final value of nh. In addition it gives a predictive density for the data on a series of points chosen to span the effective range of the density but with more points towards its 'centre'.

First `bkde.h`

```
// bkde.h
// Define class for bkde problem

#ifndef BKDE
#define BKDE
#include <math.h>

const int ASIZE = 5000;
const int PredPoints = 19;
const int nPred = 1;
const double SNCON = M_2_SQRTPI/(2*M_SQRT2);
const double S2MPI = M_2_SQRTPI/(2*M_SQRT2);
// 1/sqrt(2pi) = 2/sqrt(pi) / (2*sqrt(2))

class bkde : public Bayes4
{
      int nx;
      double x[ASIZE];
      double xt[ASIZE];
      double xpred[ASIZE];
      double h;
      double mu;
      double sd;
      double dmu;
      double dsd;
```

```
        double dmin;
        double dmax;
        double drange;
        //
        // normal at (xx)
        //
        // the kernel
        //
        double K(double xx, double u, double s)
        {
              double z = (xx - u) / s;
              return S2MPI/s * exp(-0.5 * z * z);
        }
        // KDE
        double kernel(double xi, double X[], int n, double hh);
        // special function analysis stuff
        double pm;
        double psd;
public:
        bkde():mu(0), sd(1) {};
        void load(char*, int&);
        double log_likelihood();
        double prior();
        double ftran(float*);
// special function analysis stuff
        void spec_start(int&);
        void spec_fun(float*);
        void spec_out();
};
#endif
```

Followed by bkde.cc

```
#include "/home/danny/bayes4/b4/include/Bayes4.h"
#include "bkde.h"
#include <stdio.h>
#include <math.h>
#include <string.h>


int main()
{
bkde regcoef;
doBayes(&regcoef);
return 0;
}

#include <stdio.h>

void bkde::load(char* pnames, int& npar)
{
char title[80];
int i=0;
double x2sum=0.0, xsum=0.0, u;
// set up title
fgets(title, 80, pin);
  fprintf(sout,"%s\n",title);
  printf("%s\n",title);
  // initialise range variables
  fscanf(pin,"%lf", &x[0]);
dmin = x[0];
dmax = x[0];
dmu = x[0];
// input data and find stats
for ( nx=1; fscanf(pin,"%lf", &x[nx])==1 ; nx++)
```

```
{
if(x[nx] < dmin) dmin = x[nx];
if(x[nx] > dmax) dmax = x[nx];
u = x[nx] - dmu;
xsum += u;
x2sum += u * u;
}
dmu += xsum / nx;
dsd = sqrt(1.0 / (nx - 1) * (x2sum - xsum * xsum / nx ));
  printf(" %d items read: %lf ... %lf\n", nx, x[0], x[nx-1]);
printf(" sample mean = %7.4f, sample standard deviation = %7.4f \n", dmu,
dsd);
// normalise
for(i = 0; i < nx; i++) xt[i] = (x[i] - dmu) / dsd;
drange = dmax - dmin;
printf(" drange = %7.4f, dmin = %7.4f, dmax = %7.4f\n\n", drange, dmin,
dmax);
  npar = 1;
  Load_Name("h-width",0, pnames);
}

//
// inverse transforms
//
// h must be positive so work with log h
//
double bkde::ftran(float* theta)
{
h = exp(theta[0]);
//printf("\ntheta = %7.4f\n",h);
return 1;
}

//
```

```
// KDE
//
// at point xi, using set of points X and
// Normal Kernel
//
double bkde::kernel(double xi, double X[], int n, double hh)
{
int i = 0;
double sum = 0.0;

for(i = 0; i < n; i++) sum += K(xi, X[i], hh);
sum /= n;
return sum;
}

//
// Prior
//
// Normal prior
//
double bkde::prior()
{
//double n = K(h, mu, sd);
return 1;
}

//
// Log likelihood of data
//
double bkde::log_likelihood()
{
//printf("\n\nxxx\n\n");
int i;
double lik = 0.0;
```

```
for(i = 1; i < nx; i++) lik += log(kernel(xt[i], xt, i, h));
return lik;
}

// Now for Member Functions for predictive analysis

void bkde::spec_start(int& number_funs)
{
double step = drange / (PredPoints - 1);
int i = 0;

number_funs = nPred * PredPoints;

xpred[0] = dmin;
for (int i = 1; i < PredPoints; i++) xpred[i] = xpred[i - 1] + step;
}

void bkde::spec_fun(float* funvals)
{
int i = 0;
for (int i = 0; i < PredPoints; i++)
*funvals++ = kernel(xpred[i], x, nx, h);
}

void bkde::spec_out()
{
int i = 0;
double density[PredPoints];
double max = 0.0;
int temp = PredPoints;
char title[] = "predx";

fprintf(sout, "Predictive Distribution for X\n");
```

```
for (i = 0; i < PredPoints; i++)
{
density[i] = SpecValue(i);
if(max < density[i]) max = density[i];
}
Output_Density(xpred, density, PredPoints, "Length");
}
```

It is apparent that even with the inconvenience of two files the C++ version is both shorter and easier to use.

C.2.2 Variable bandwidth KDE

This version implements a variable bandwidth KDE in which bandwidth depends on an inverse relationship with a simple fixed bandwidth KDE. It gives a predictive density for the data on a series of points chosen to span the effective range of the density but with more points towards its 'centre'.

First `vbkde.h`

```
// bkde.h
// Define class for bkde problem

#ifndef BKDE
#define BKDE

#include <math.h>


const int ASIZE = 5000;
```

```
const int PredPoints = 19;
const int nPred = 1;
const double SNCON = M_2_SQRTPI/(2*M_SQRT2);
const double S2MPI = M_2_SQRTPI/(2*M_SQRT2);
// 1/sqrt(2pi) = 2/sqrt(pi) / (2*sqrt(2))

class bkde : public Bayes4
{
      int nx;
      double x[ASIZE];
      double xt[ASIZE];
      double xpred[ASIZE];
      double h;
      double mu;
      double sd;
      double dmu;
      double dsd;
      double dmin;
      double dmax;
      double drange;
      //
      // normal at (xx)
      //
      // the kernel
      //
      double K(double xx, double u, double s)
 {
            double z = (xx - u) / s;
            return S2MPI / s * exp(-0.5 * z * z);
      }
      // KDE
      double kernel(double xi, double X[], int n, double hh);
      // special function analysis stuff
      double pm;
```

```
      double psd;
public:
      bkde():mu(0), sd(1) {};
      void load(char*, int&);
      double log_likelihood();
      double prior();
      double ftran(float*);
// special function analysis stuff
      void spec_start(int&);
      void spec_fun(float*);
      void spec_out();
};

#endif
```

Finally vbkde.cc

```
#include "/home/danny/bayes4/b4/include/Bayes4.h"
#include "bkde.h"
#include <stdio.h>
#include <math.h>
#include <string.h>


int main()
{
bkde regcoef;
doBayes(&regcoef);
return 0;
}

#include <stdio.h>

void bkde::load(char* pnames, int& npar)
```

```
{
char title[80];
int i=0;
double x2sum=0.0, xsum=0.0, u;
// set up title
fgets(title, 80, pin);
  fprintf(sout,"%s\n",title);
  printf("%s\n",title);
  // initialise range variables
  fscanf(pin,"%lf", &x[0]);
dmin = x[0];
dmax = x[0];
dmu = x[0];
// input data and find stats
for ( nx=1; fscanf(pin,"%lf", &x[nx])==1 ; nx++)
{
if(x[nx] < dmin) dmin = x[nx];
if(x[nx] > dmax) dmax = x[nx];
u = x[nx] - dmu;
xsum += u;
x2sum += u * u;
}
dmu += xsum / nx;
dsd = sqrt(1.0 / (nx - 1) * (x2sum - xsum * xsum / nx / nx));
  printf(" %d items read: %lf ... %lf\n", nx, x[0], x[nx-1]);
printf(" sample mean = %7.4f, sample standard deviation = %7.4f \n", dmu,
dsd);
// normalise
for(i = 0; i < nx; i++) xt[i] = (x[i] - dmu) / dsd;
drange = dmax - dmin;
printf(" drange = %7.4f, dmin = %7.4f, dmax = %7.4f\n\n", drange, dmin,
dmax);
  npar = 1;
  Load_Name("h-width",0, pnames);
```

```
}

//
// inverse transforms
//
// h must be positive so work with log h
//
double bkde::ftran(float* theta)
{
h = exp(theta[0]);
return h;
}

//
// KDE
//
// at point xi, using set of points X and
// Normal Kernel
//
double bkde::kernel(double xi, double X[], int n, double hh)
{
int i = 0;
double sum = 0.0;

for(i = 0; i < n; i++) sum += K(xi, X[i], hh);
sum /= n;
return sum;
}

//
// Prior
//
// Normal prior
//
```

```
double bkde::prior()
{
double n = K(h, mu, sd);
return 1;
}

//
// Log likelihood of data
//
double bkde::log_likelihood()
{
//printf("\n\nxxx\n\n");
int i, j;
double sum = 0.0;
double lik = 0.0;

for(i = 1; i < nx; i++) lik += log(kernel(xt[i], xt, i, h));
return lik;
}

// Now for Member Functions for predictive analysis

void bkde::spec_start(int& number_funs)
{
double step = drange / (PredPoints - 1);
int i = 0;

number_funs = nPred * PredPoints;

xpred[0] = dmin;
for (int i = 1; i < PredPoints; i++) xpred[i] = xpred[i - 1] + step;
}

void bkde::spec_fun(float* funvals)
```

```
{
int i = 0;
for (int i = 0; i < PredPoints; i++)
funvals[i] = kernel(xpred[i], x, nx, h);
}

void bkde::spec_out()
{
int i = 0;
double density[PredPoints];
double max = 0.0;
int temp = PredPoints;
char title[] = "predx";

fprintf(sout, "Predictive Distribution for X\n");

for (i = 0; i < PredPoints; i++)
{
density[i] = SpecValue(i);
if(max < density[i]) max = density[i];
}
Output_Density(xpred, density, PredPoints, "Length");
}
```

C.3 Building the system

A sample makefile is given here, it is able to check on the state of the Bayes4 system and rebuild if necessary.

```
# bkde

PROG = bkde

BOME = ../..

default_target: $(PROG)

include $(BOME)/common.makefile

OBJ = bkde.o

$(PROG): $(OBJ) $(LIB)
g++ -o $@ $(OBJ) $(LIBS)

$(OBJ): bkde.h Normal2.h $(INCFILES)

clean:
-$(RM) $(PROG) *.o

clean_all:
$(MAKE) -C$(BOME) clean
```

Appendix D

Benchden KDE test densities

Alain Berlinet and Luc Devroye Berlinet and Devroye (1994) published a set of 28 test densities for density estimation, in the following each of the densities is shown in six views:

1. A histogram.
2. An estimated density for a sample of 10000 points using the inbuilt R function `density()`.
3. An estimated density for a sample of 1000 points using BKDE.
4. An estimated density for a sample of 1000 points using adaptive BKDE.
5. An estimated density for a sample of 100 points using BKDE.
6. An estimated density for a sample of 100 points using adaptive BKDE.

The densities were chosen to be a test of an estimators ability to deal with difficult densities and the overriding message from the paper (Berlinet and

Devroye, 1994) is that there is no density estimator that handles all estimation problems well.

In the examples here a histogram and the estimator built in to R are given to provide comparisons to the four BKDE outputs. In some of the densities, for example the inverse exponential density, dnum=20, the values are so widely spaced that it is not possible to estimate the density without an extremely large number of samples. In these cases a limited range has been used to allow the estimation of the "interesting" portion of the density.

For each diagram the value of `dberdev(seq(-3,3,0.01),dnum=1)`[1] is overlaid on the estimate.

In Berlinet and Devroye Berlinet and Devroye (1994) each density estimator is tested by averaging the results of 20 tests each using a sample of $n = 100$. Here the estimators were tested against a single sample with $n = 1000$ for the Histogram, another with $n = 1000$ for the R estimator and two samples with $n = 100$ and $n = 1000$ for the two BKDE estimators. No particular selection was made and each sample was as generated, this can lead to some difference from the reference density as in the $n = 100$ samples used with the BKDE estimators (e.g. D.5).

[1] Where the value of dnum varies from 1 to 28 giving the appropriate line for each density.

D.1 Uniform Density

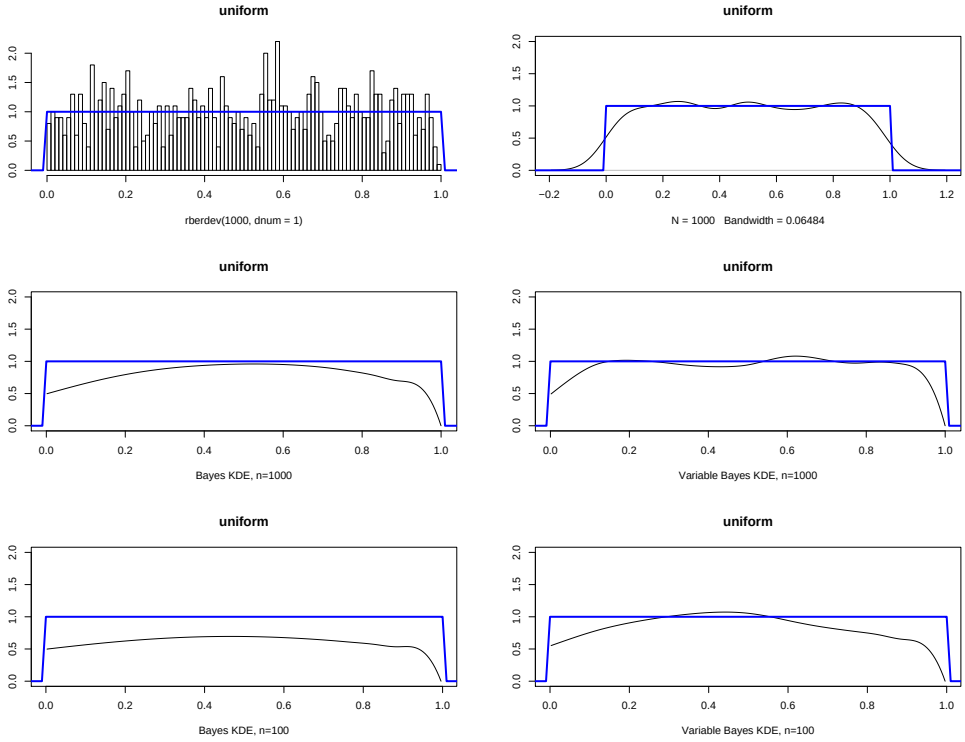


Figure D.1: *Uniform density.*

D.2 Exponential Density

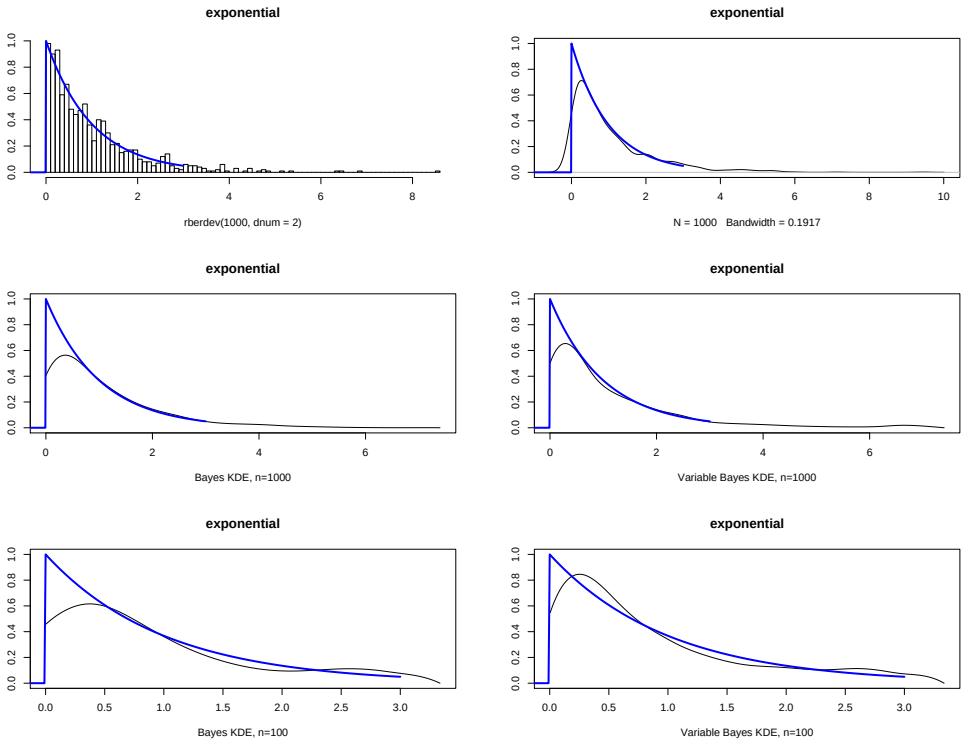


Figure D.2: *Exponential density.*

D.3 Maxwell Density

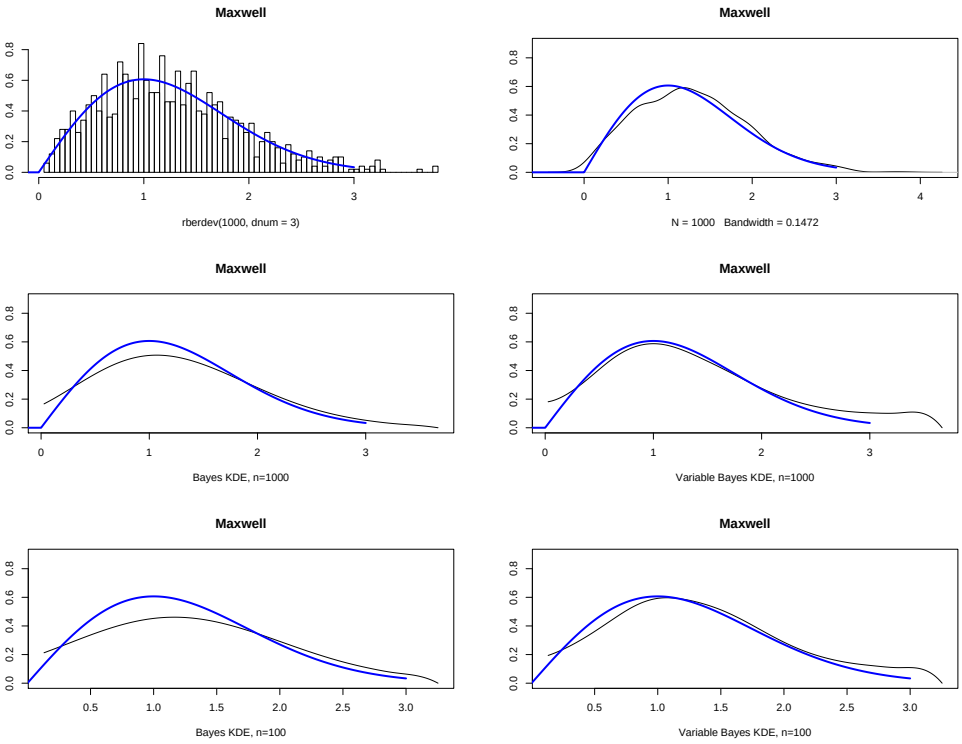


Figure D.3: *Maxwell density.*

D.4 Double Exponential Density

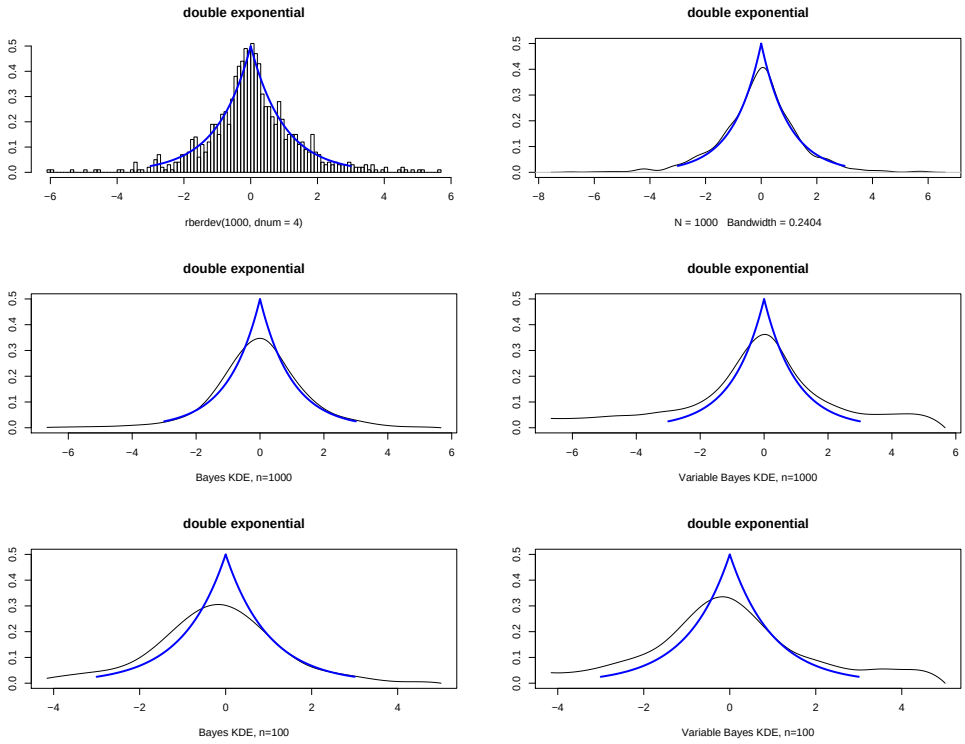


Figure D.4: *Double Exponential density.*

D.5 Logistic Density

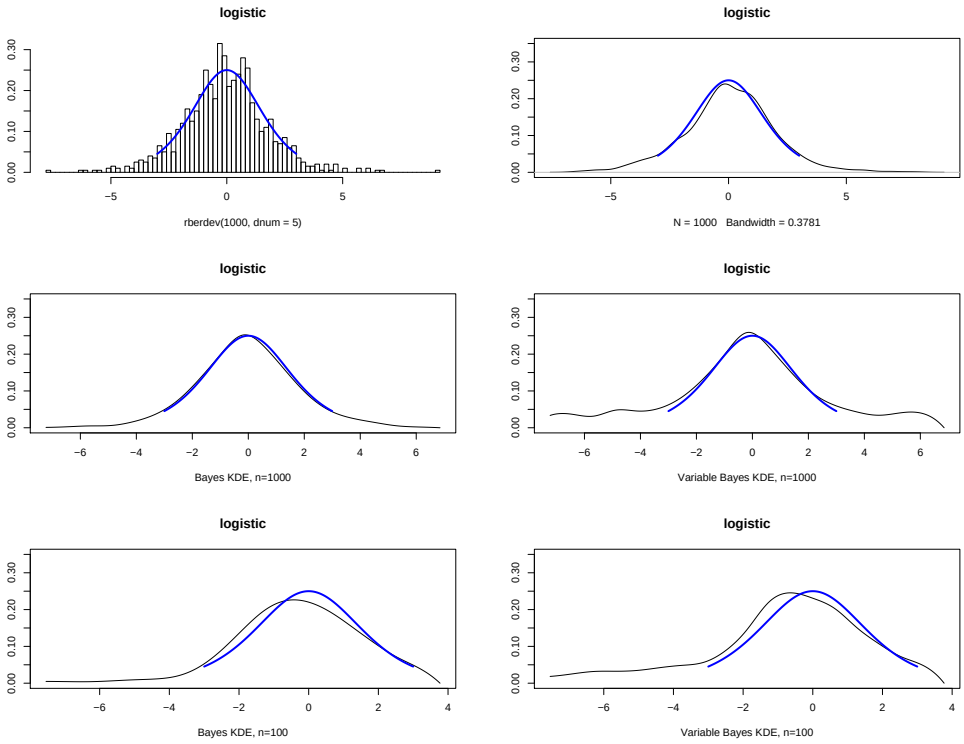


Figure D.5: *Logistic density.*

D.6 Cauchy Density

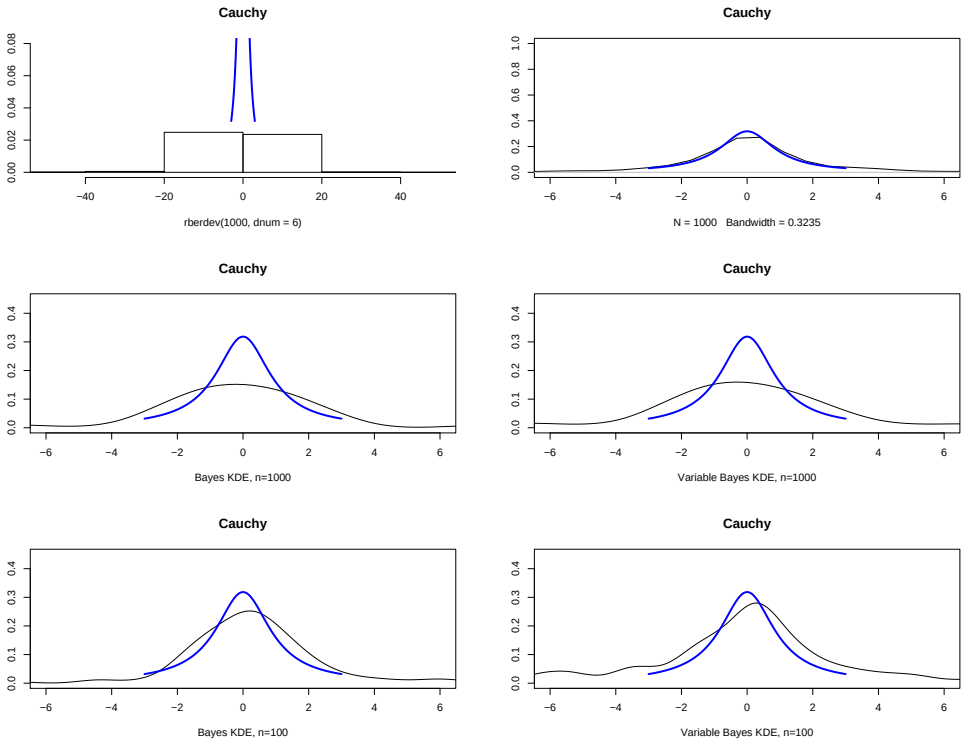


Figure D.6: *Cauchy density.*

D.7 Extreme Value Density

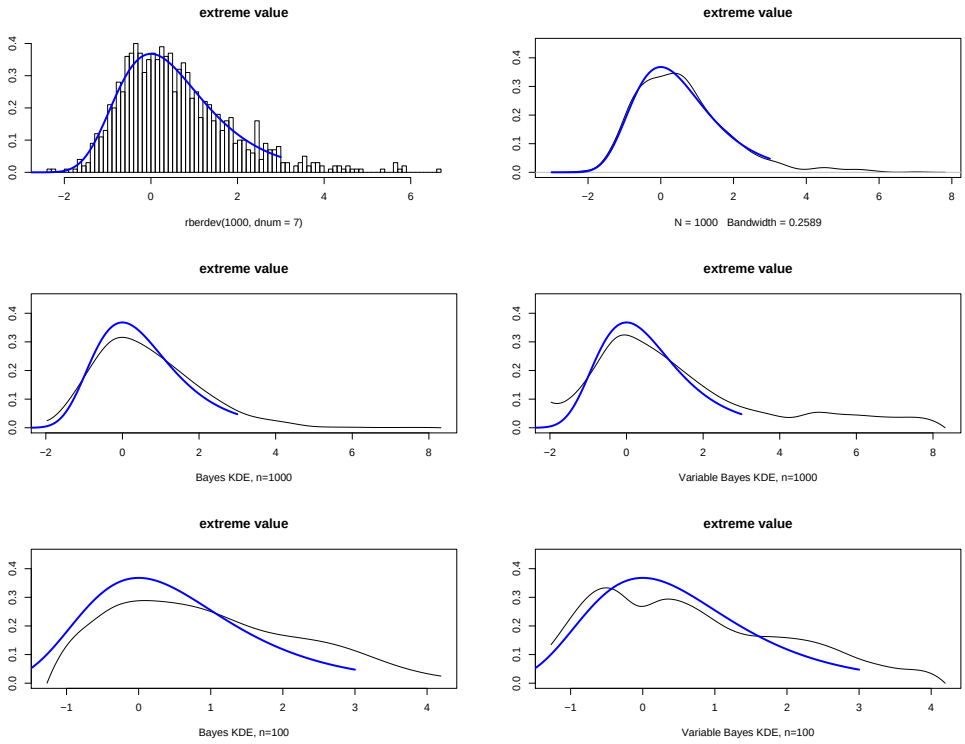


Figure D.7: *Extreme Value density.*

D.8 Infinite Peak Density

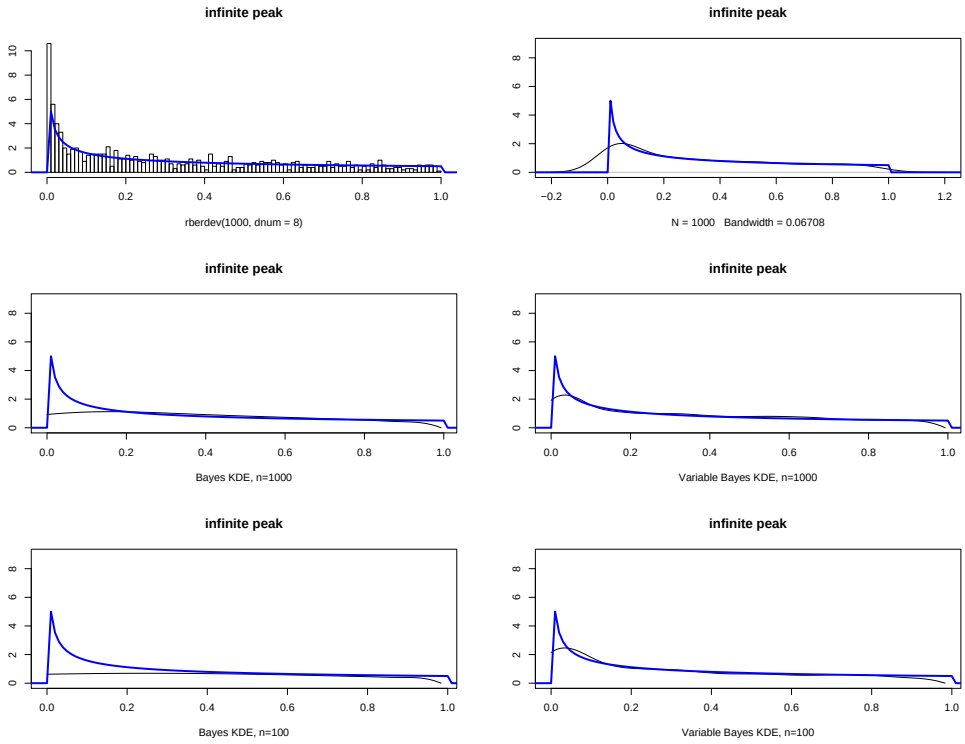


Figure D.8: *Infinite Peak density.*

D.9 Pareto Density.

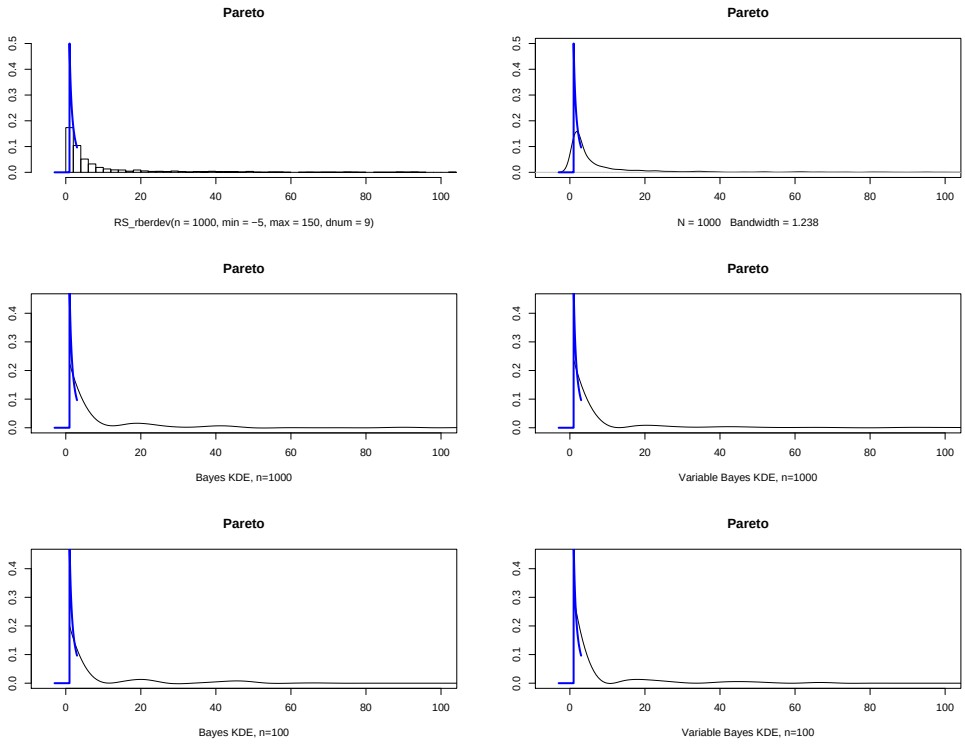


Figure D.9: *Pareto density.*

D.10 Symmetric Pareto Density

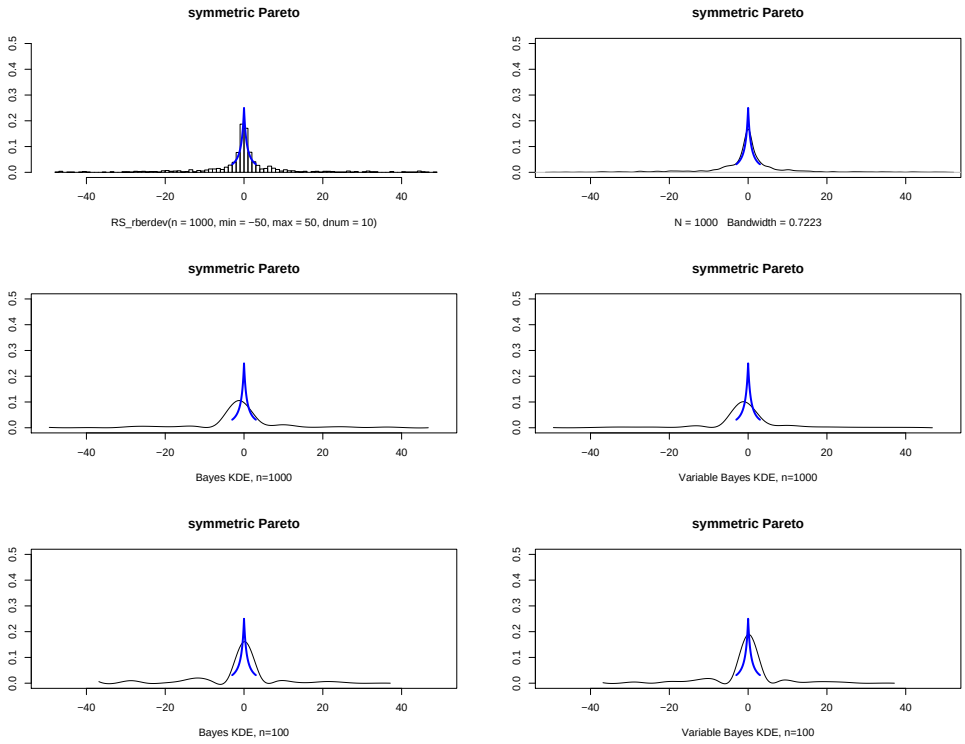


Figure D.10: *Symmetric Pareto density.*

D.11 Normal Density

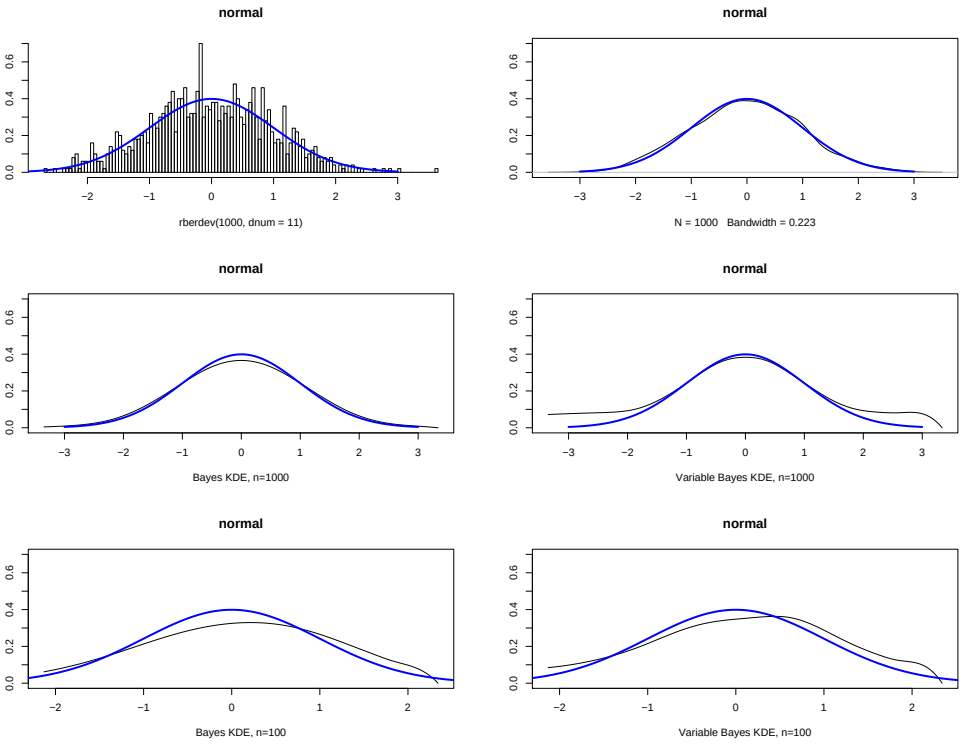


Figure D.11: *Normal density.*

D.12 Lognormal Density

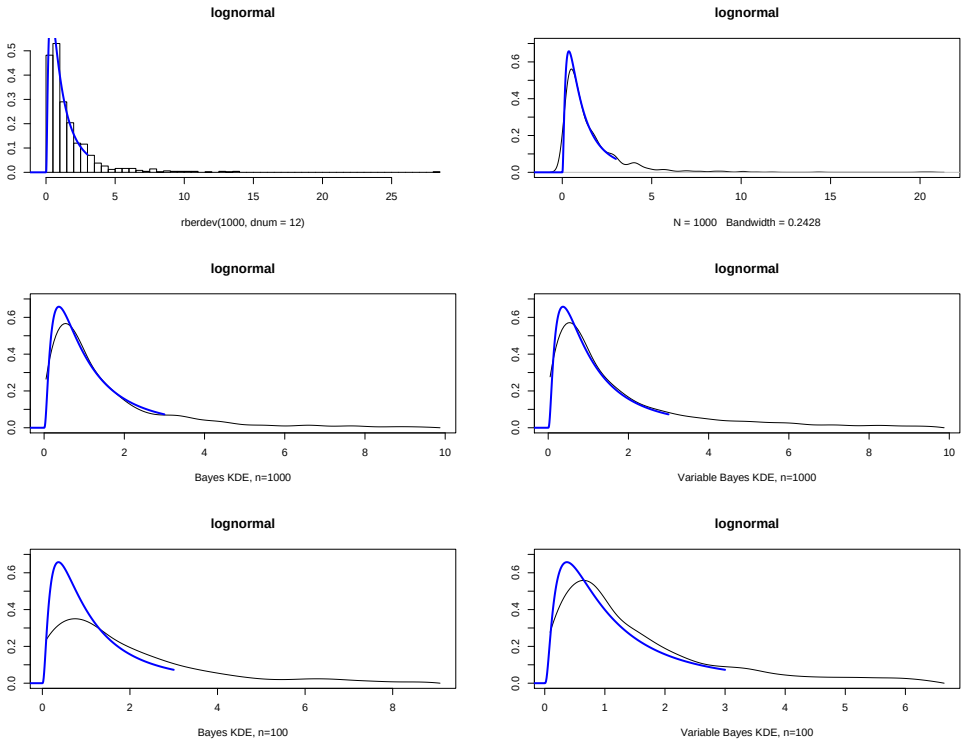


Figure D.12: *Lognormal density.*

D.13 Uniform Scale Mixture Density

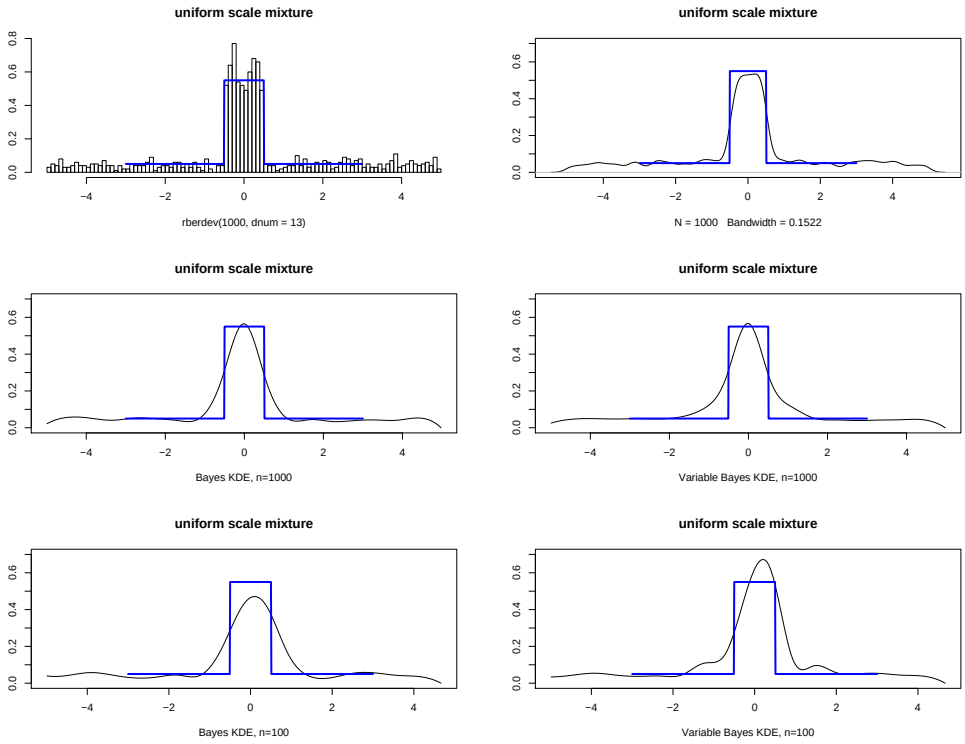


Figure D.13: *Uniform Scale Mixture density.*

D.14 Matterhorn Density

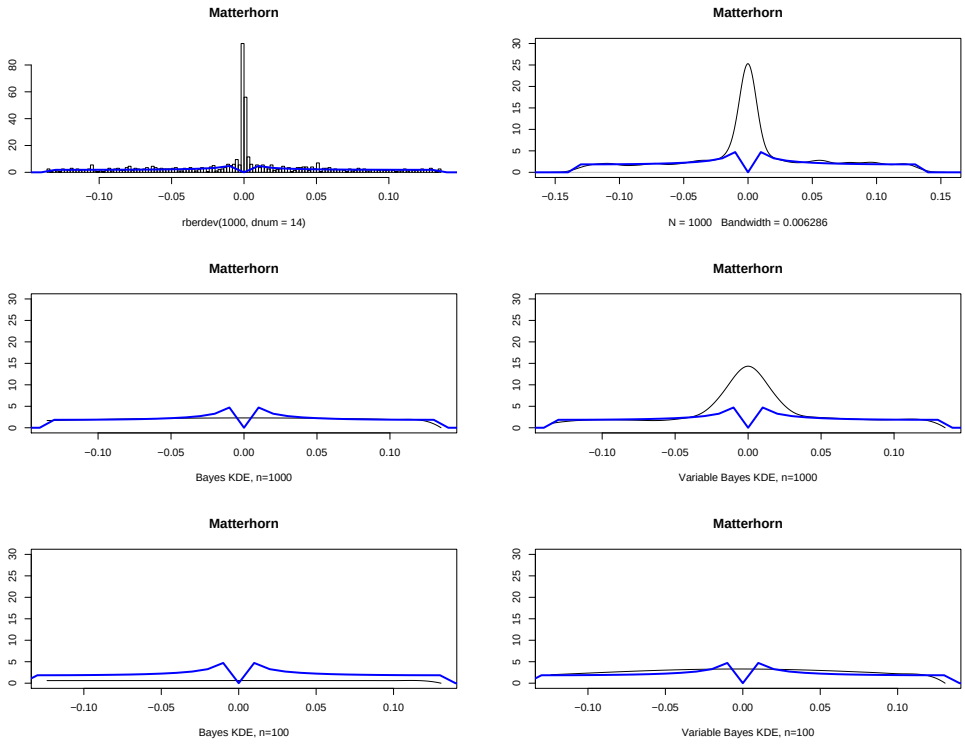


Figure D.14: *Matterhorn density.*

D.15 Logarithmic Peak Density

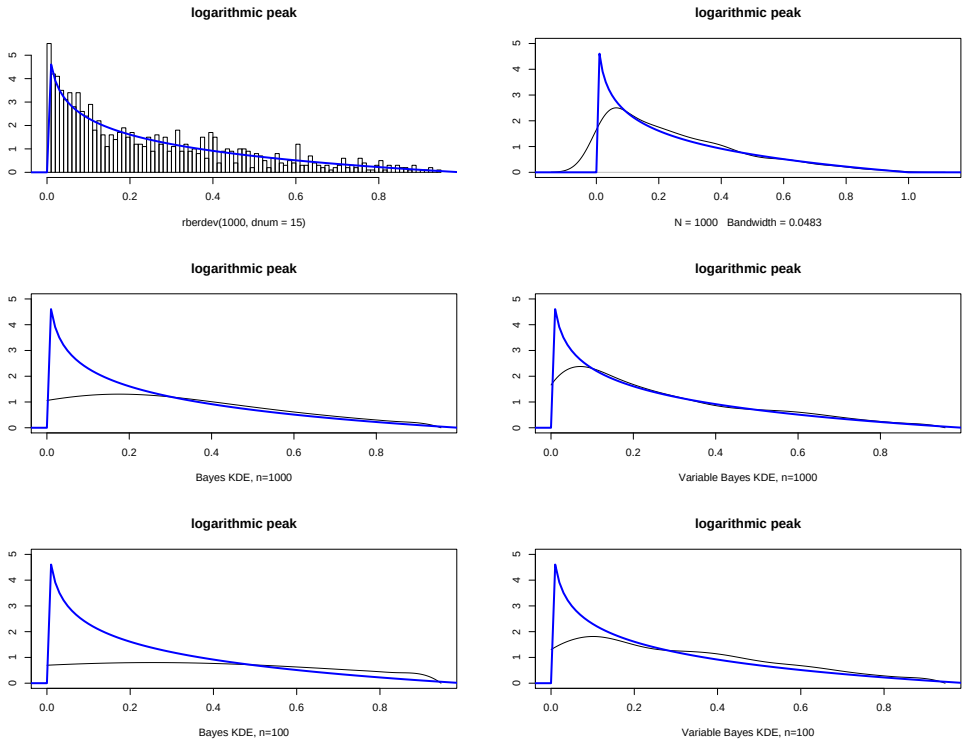


Figure D.15: *Logarithmic Peak density.*

D.16 Isosceles Triangle Density

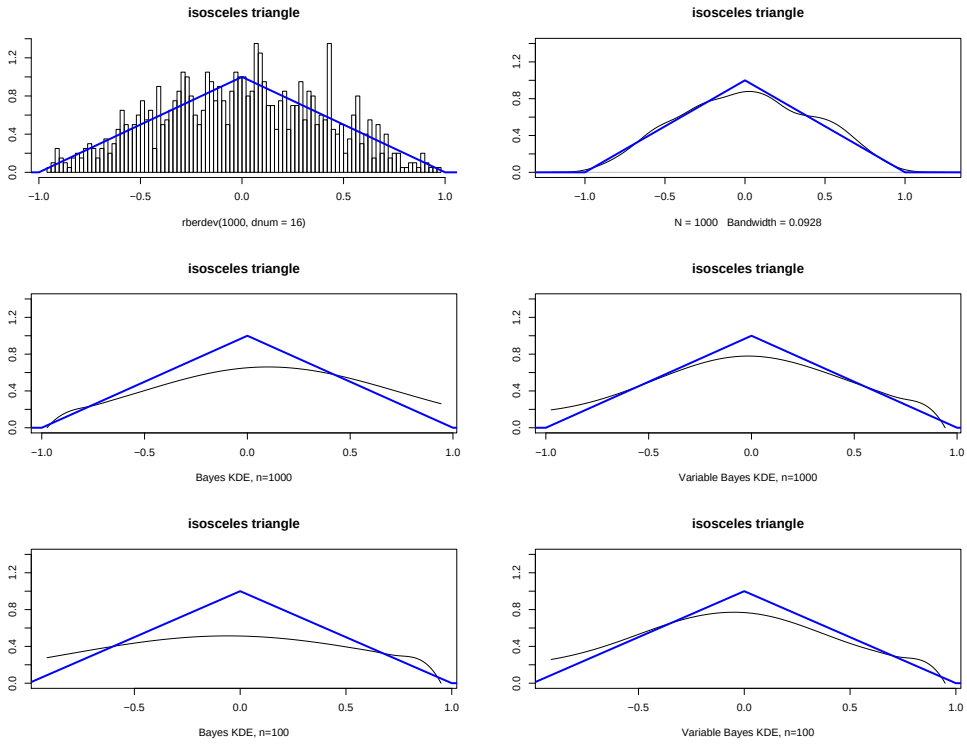


Figure D.16: *Isosceles Triangle density.*

D.17 Beta (2,2) Density

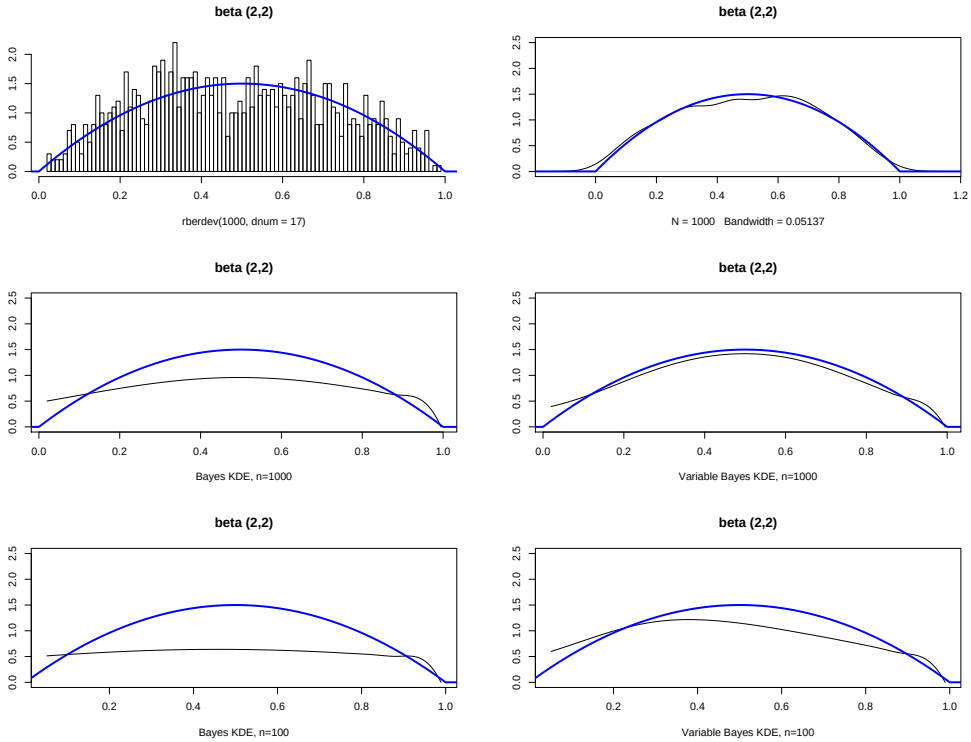


Figure D.17: *Beta (2,2) density.*

D.18 Chi-square (1) Density

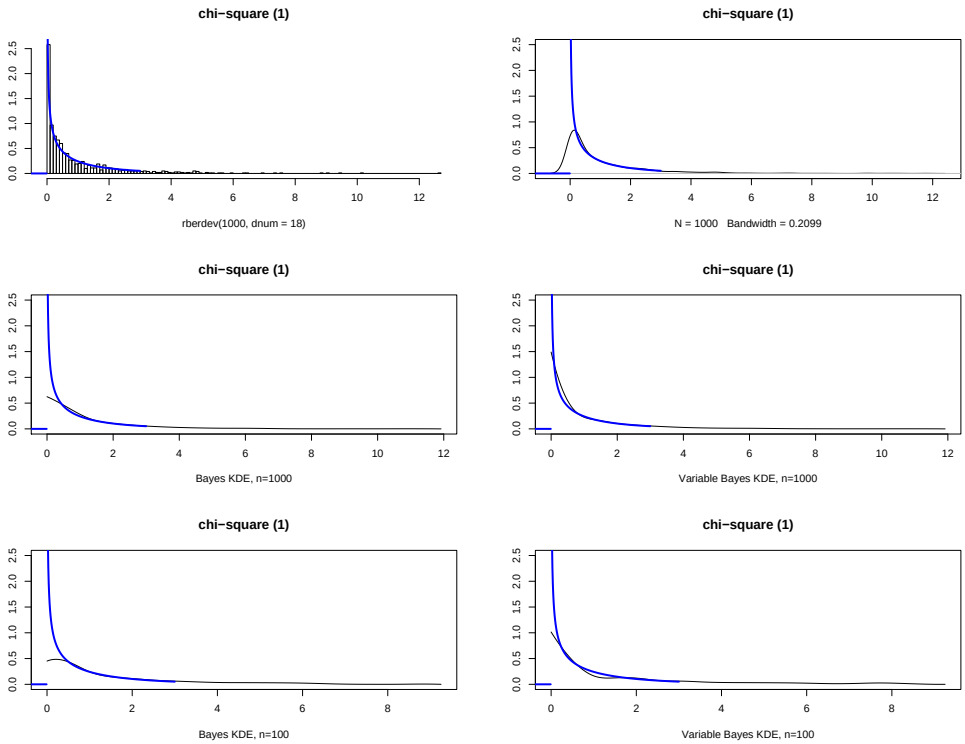


Figure D.18: *Chi-square (1) density.*

D.19 Normal Cubed Density

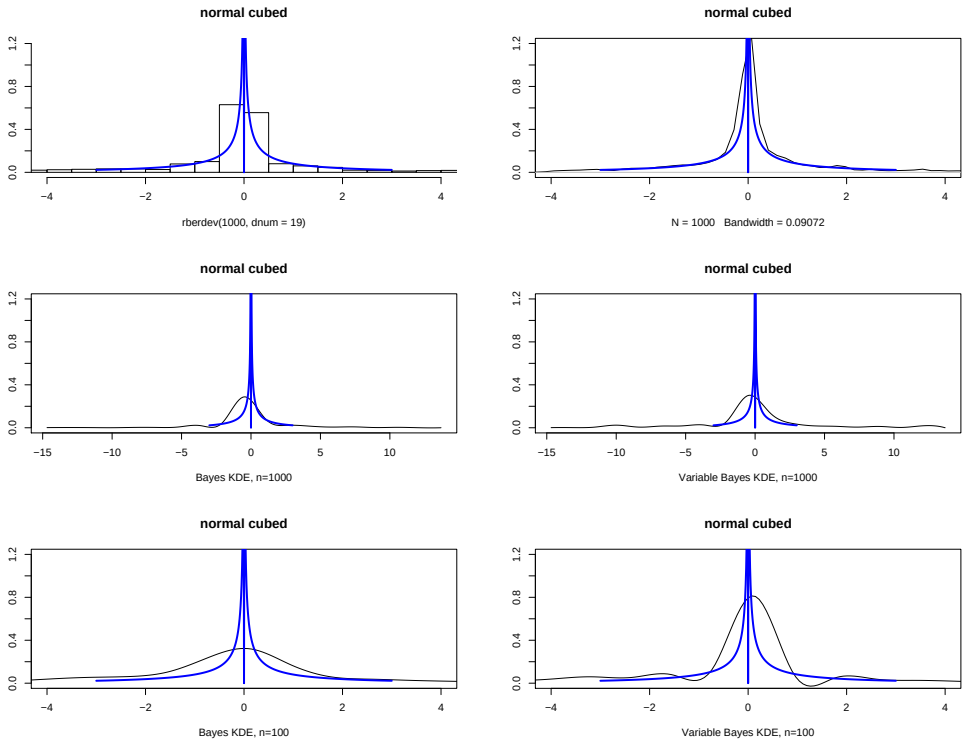


Figure D.19: *Normal Cubed density.*

D.20 Inverse Exponential Density

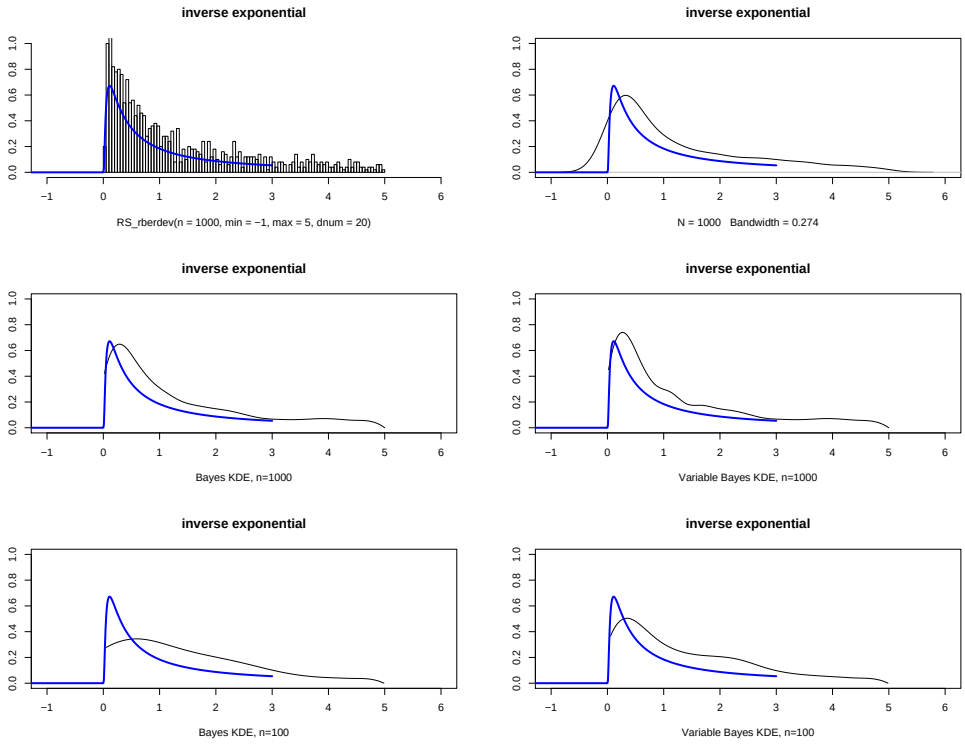


Figure D.20: *Inverse Exponential density.*

D.21 Marronite Density

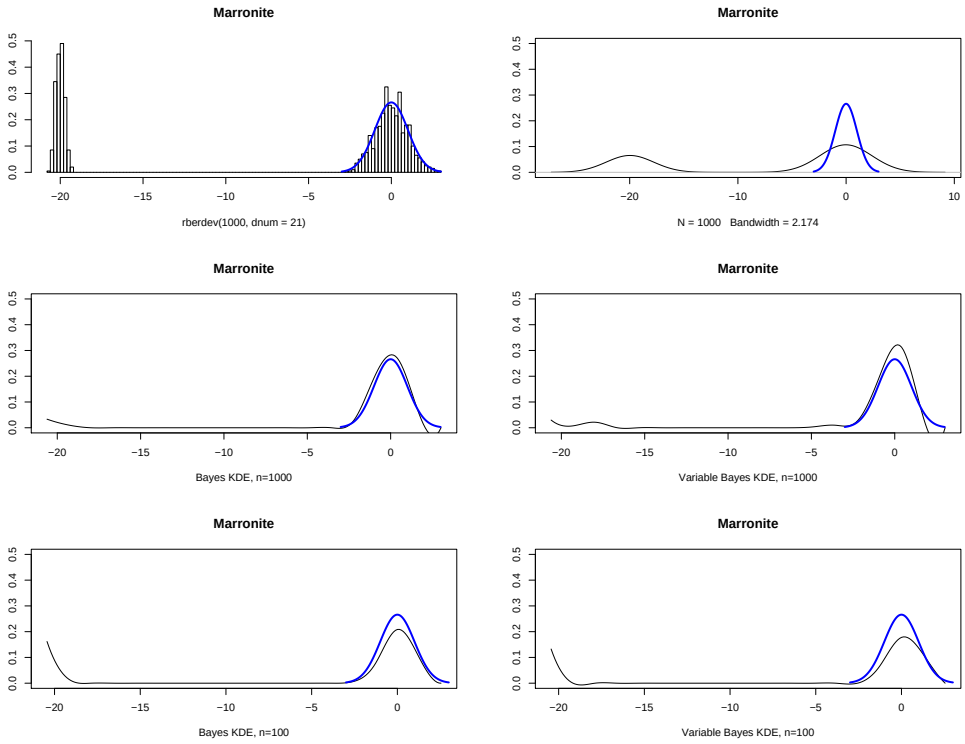


Figure D.21: *Marronite density.*

D.22 Skewed Bimodal Density

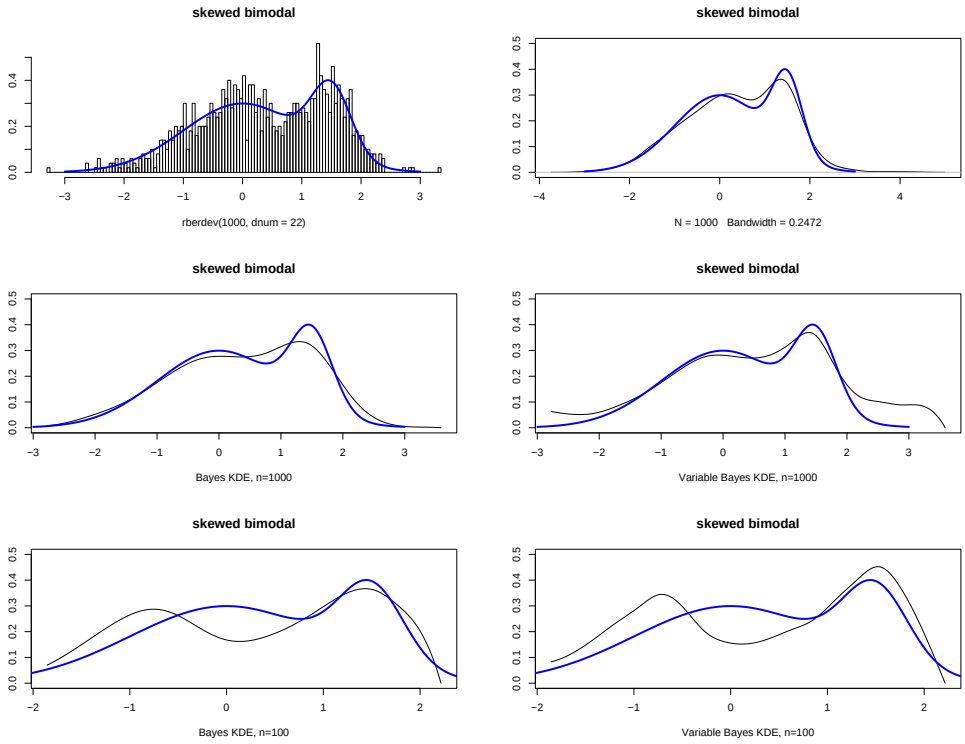


Figure D.22: *Skewed Bimodal density.*

D.23 Claw Density

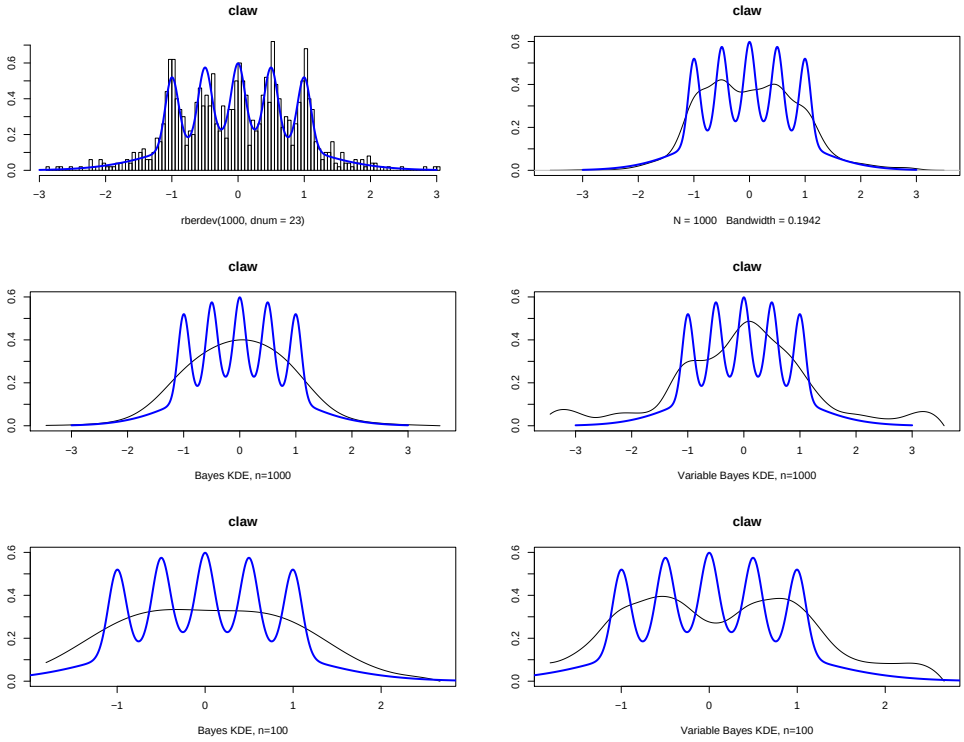


Figure D.23: *Claw density.*

D.24 Smooth Comb Density

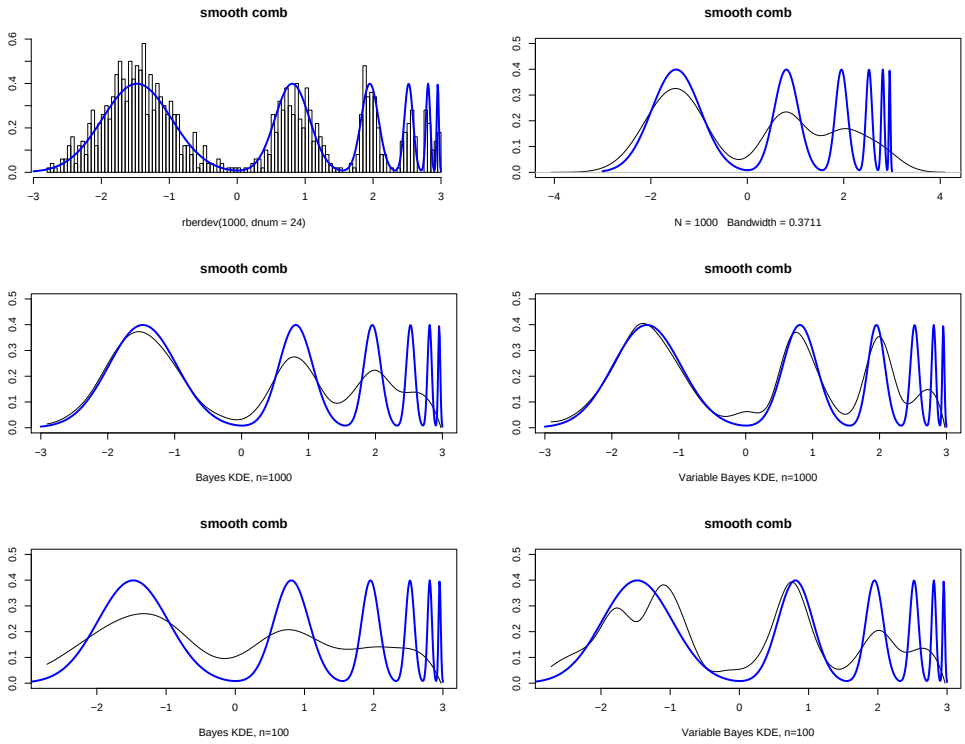


Figure D.24: *Smooth Comb density.*

D.25 Caliper Density

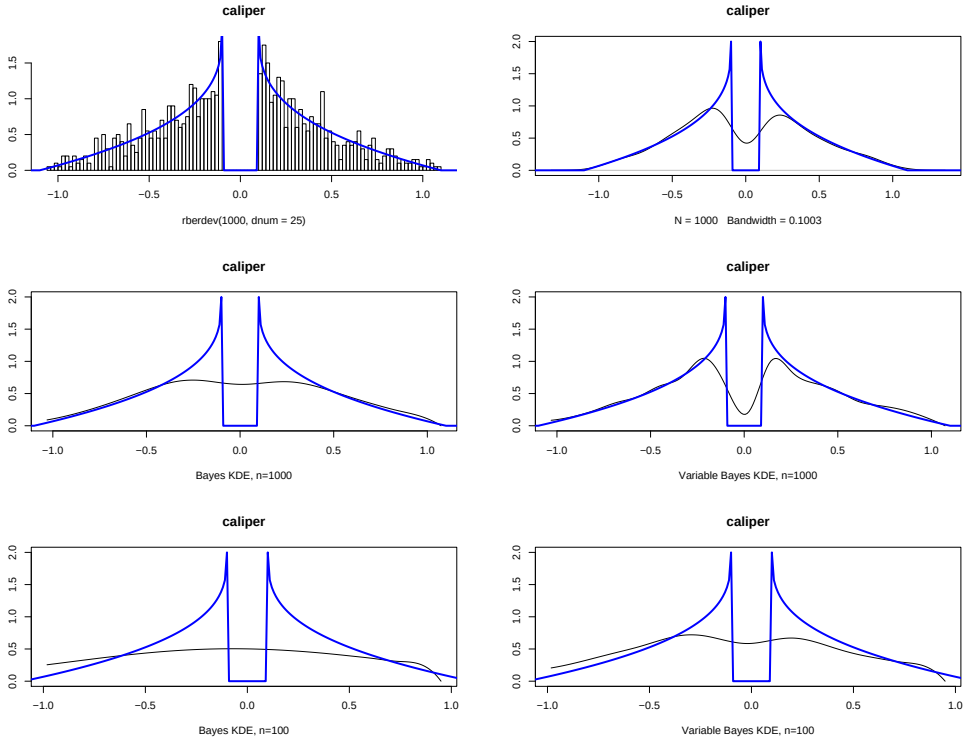


Figure D.25: *Caliper density.*

D.26 Trimodal Uniform Density

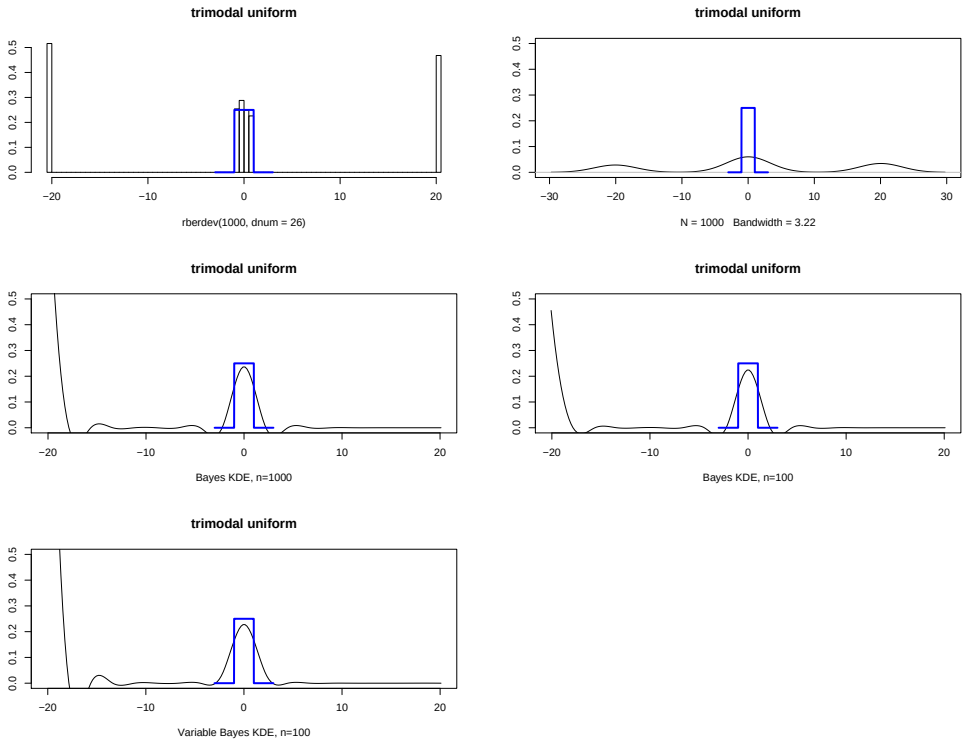


Figure D.26: *Trimodal Uniform density.*

D.27 Sawtooth Density

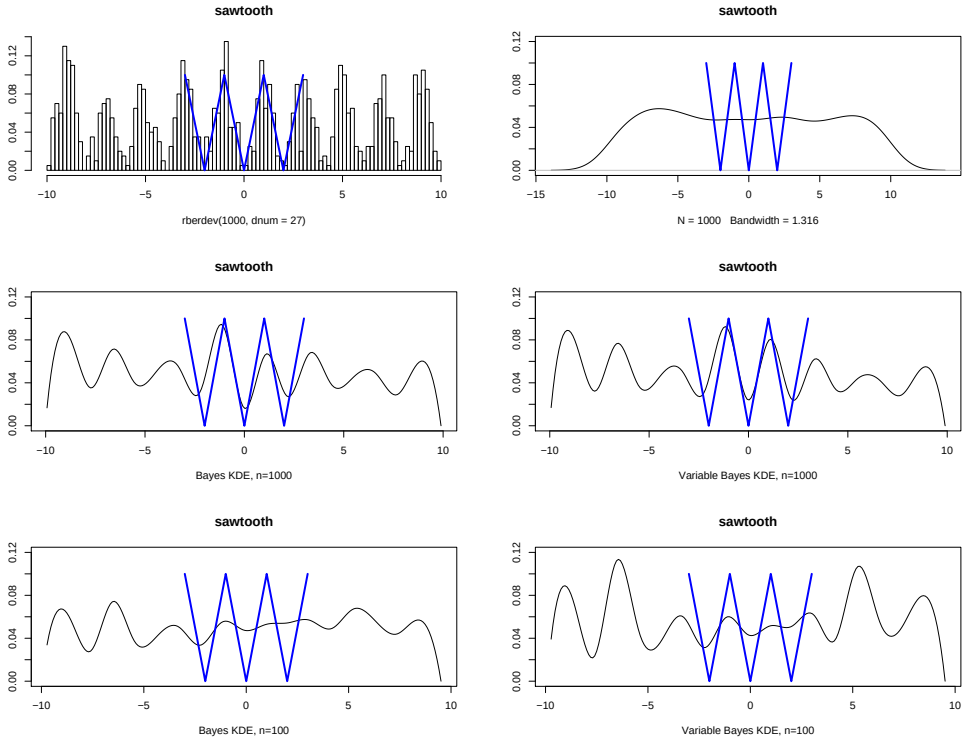


Figure D.27: *Sawtooth density.*

D.28 Bilogarithmic Peak Density

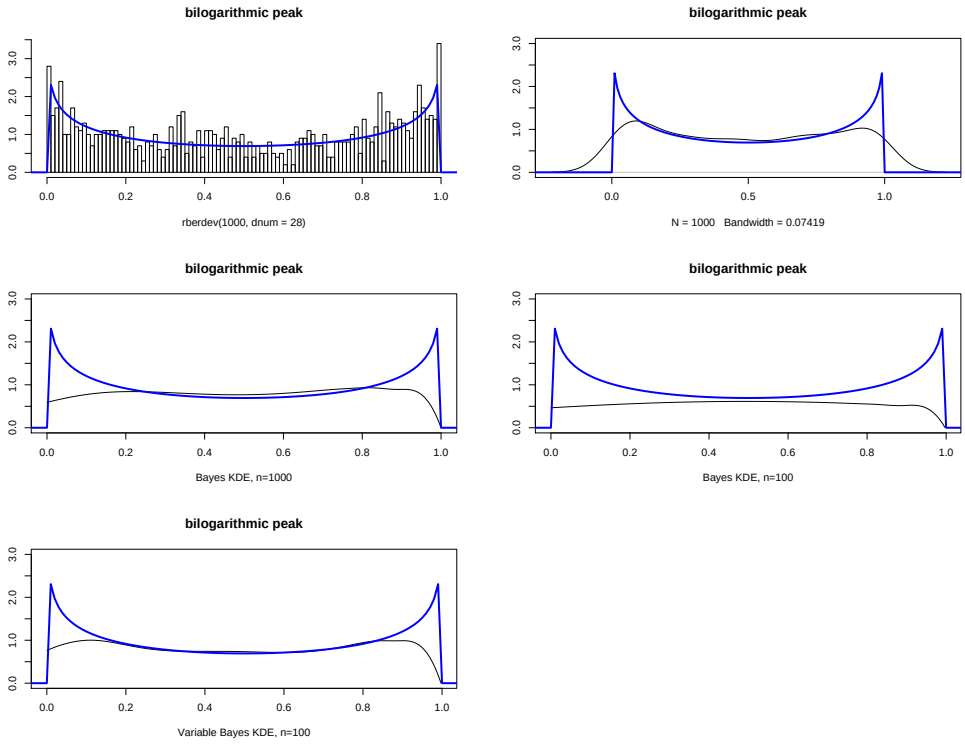


Figure D.28: *Bilogarithmic Peak density.*

Appendix E

Data

E.1 The Old Faithful Data

4.37	4.7	1.68	1.75	4.35	1.77
4.25	4.1	4.05	1.9	4	4.42
1.83	1.83	3.95	4.83	3.87	1.73
3.92	3.2	2.33	4.57	3.58	3.7
4.25	3.58	3.67	1.9	4.13	4.53
4.1	4.12	4	4.93	3.68	1.85
3.83	1.85	3.8	3.8	3.33	3.73
1.67	4.63	1.83	2.03	2.72	4.03
1.73	3.1	4.62	1.88	3.52	3.77
3.43	2	3.73	4.6	2.93	4.65
4.18	4.58	3.5	4.62	4.03	1.97
4.6	4	3.75	4	4.33	1.82
1.67	3.5	4.2	4.43	1.9	4.08

3.43	1.77	4.5	1.8	3.7	2.5
2.27	2.93	4.63	4	1.97	3.93
4.07	4.5	2.25	4.25	4.08	3.92
4.73	3.72	4.5	4.4	4.58	3.5
1.8	4.28	4.33 4	.13	1.95	

E.2 Largest canonical variable for 6 teeth Andrews (1972, Table 2)

Tooth type	1	2	3	4	5	6
A	-5.35	-7.07	-9.37	-4.28	-2.15	-2.93
B	3.93	-6.04	-8.87	-2.16	-0.5	-1.09
C	3.12	6.66	6.28	4.96	4.13	4.60
D	1.45	1.73	4.82	3.96	3.35	3.63
E	2.83	5.10	5.11	2.72	1.21	1.49
F	1.49	1.63	3.61	1.29	-0.171	0.0503
G	0.38	3.82	3.46	-1.65	-2.32	-1.92
H	0.01	0.231	3.05	-2.25	-2.65	-2.15
I	-4.52	-6.49	-7.79	3.45	4.91	3.72
J	-1.81	-2.94	-6.63	-0.369	-1.32	1.09

Index