8 Image Analysis

"Reduce it to binary, Siobhan," she told herself.
—**Ian Rankin**, ***Resurrection Men***.—

Roadmap

This final chapter covers the analysis of pixelized images through Markov random field models, towards pattern detection and image correction. We start with the statistical analysis of Markov random fields, which are extensions of Markov chains to the spatial domain, as they are instrumental in this chapter. This is also the perfect opportunity to cover the ABC method, as these models do not allow for a closed form likelihood. Image analysis has been a very active area for both Bayesian statistics and computational methods in the past 30 years, so we feel it well deserves a chapter of its own for its specific features.

8.1 Image Analysis as a Statistical Problem

If we think of a computer image as a (large) collection of colored pixels disposed on a grid, there does not seem to be any randomness involved nor any need for statistical analysis! Nonetheless, image analysis seen as a statistical analysis is a thriving field that saw the emergence of several major statistical advances, including, for instance, the Gibbs sampler. (Moreover, this field has predominantly adopted a Bayesian perspective both because this was a natural thing to do and because the analytical power of this approach was higher than with other methods.) The reason for this apparent paradox is that, while pixels usually are deterministic objects, the complexity and size of images require one to represent those pixels as the random output of a distribution governed by an object of much smaller dimension. For instance, this is the case in computer vision, where specific objects need to be extracted out of a much richer (or noisier) background.

In this spirit of extracting information from huge dimensional structure, we thus build in Sect. 8.2 a specific family of distributions inspired from particle physics, the Potts model, in order to structure images and other spatial structures in terms of local homogeneity. Unfortunately, this is a mostly theoretical section with very few illustrations. In Sect. 8.3, we address the fundamental issue of handling the missing normalizing constant in these models by introducing a new computational technique called ABC that operates on intractable likelihoods (with the penalty of producing an approximative answer). In Sect. 8.4, we impose a strong spatial dimension on the prior associated with an image in order to gather homogeneous structures out of a complex or blurry image.

8.2 Spatial Dependence

8.2.1 Grids and Lattices

An image (in the sense of a computer generated image) is a special case of a *lattice*, in the sense that it is a random object whose elements are indexed by the location of the pixels and are therefore related by the geographical proximity of those locations. In full generality, a *lattice* is a mathematical multidimensional object on which a neighbourhood relation can be defined. Even though the original analysis of lattice models by Besag (1974) focussed on plant ecology and agricultural experiments, the neighbourhood relation is only constrained to be a symmetric relation and it does not necessarily have a connection with a geographical proximity, nor with an image. For instance, the relation can describe social interactions between Amazon tribes or words in a manuscript sharing a linguistic root. (The neighbourhood relation between two points of the lattice is generally translated in statistical terms into a probabilistic dependence between those points.) The lattice associated with an image is a regular $n \times m$ array made of (i, j)'s $(1 \leq i \leq n, 1 \leq j \leq m)$, whose nearest (but not necessarily only) neighbors are made of the four entries $(i, j-1)$, $(i, j+1)$, $(i-1, j)$ and $(i+1, j)$. In order to properly describe a dependence structure in images or in other spatial objects indexed by a lattice, we need to expand the notion of Markov chain on those structures. Since a lattice is a multidimensional object—as opposed to the unidimensional line corresponding to the times of observation of the Markov chain—, a first requirement for the generalization is to define a proper neighbourhood structure.

In order to illustrate this notion, we consider a small dataset[1] depicting the presence of tufted sedges[2] in a part of a wetland. This dataset, called **Laichedata**, is simply a 25×25 matrix of zeroes and ones. The corresponding lattice is the 25×25 array (Fig. 8.1).

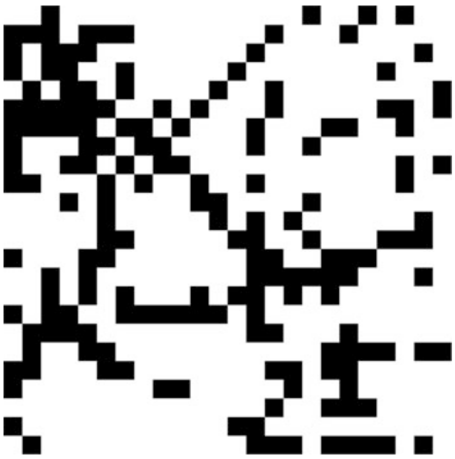

Fig. 8.1. Presence/absence of the tufted sedge plant (*Carex elata*) on a rectangular patch

Given a lattice $\mathcal{I}$ of sites $i \in \mathcal{I}$ on a map or of pixels in an image,[3] a neighbourhood relation on $\mathcal{I}$ is denoted by $\sim$, $i \sim j$ meaning that i and j are *neighbors*. If we associate a probability distribution on a vector $\mathbf{x}$ indexed by the lattice, $\mathbf{x} = (x_i)_{i \in \mathcal{I}}$, with this relation, meaning that two components x_i and x_j are correlated if the sites i and j are neighbors, a fundamental

[1] Taken from Gaetan and Guyon (2010), kindly provided by the authors.

[2] Wikipedia: "*Carex* is a genus of plants in the family *Cyperaceae*, commonly known as *sedges*. Most (but not all) sedges are found in wetlands, where they are often the dominant vegetation." Laîche is the French for sedge.

[3] We will indiscriminately use *site* and *pixel* in the remainder of the chapter.

requirement for the existence of this distribution is that the neighbourhood relation is symmetric (Cressie, 1993): if i is a neighbor of j (written as $i \sim j$), then j is a neighbor of i. (By convention, i is not a neighbor of itself.) Figure 8.2 illustrates this notion for three types of neighborhoods on a regular grid. For instance, **Laichedata** could be associated with a northwest-southeast neighbourhood to account for dominant winds: an entry (i, j) would have as neighbors $(i-1, j-1)$ and $(i+1, j+1)$.

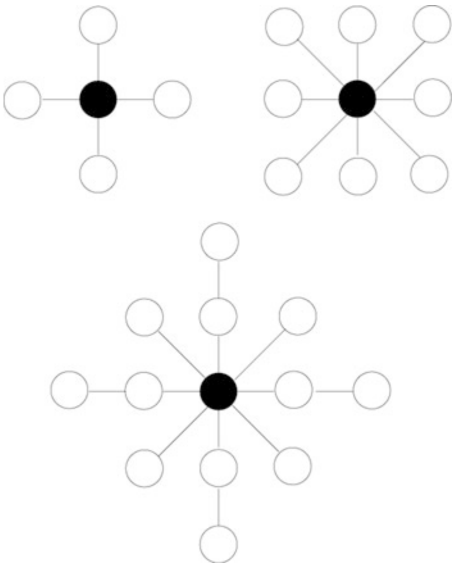

Fig. 8.2. Some common neighbourhood structures used in imaging, with four (*upper left*), eight (*upper right*), or twelve neighbors (*lower*)

8.2.2 Markov Random Fields

A *random field* on $\mathcal{I}$ is a random structure indexed by the lattice $\mathcal{I}$, a collection of random variables $\{x_i; i \in \mathcal{I}\}$ where each x_i takes values in a finite set χ. Obviously, the interesting case is when the x_i's are dependent random variables in relation with the neighbourhood structure on $\mathcal{I}$.

If $n(i)$ is the set of neighbors of $i \in \mathcal{I}$ and if $\mathbf{x}_A = \{x_i; i \in A\}$ denotes the subset of $\mathbf{x}$ for indices in a subset $A \subset \mathcal{I}$, then $\mathbf{x}_{n(i)}$ is the set of values taken by the neighbors of i. The extension from a Markov chain to a Markov random field then assumes only dependence on the neighbors.[4] More precisely, if, as before, we denote by $\mathbf{x}_{-A} = \{x_i; i \notin A\}$ the coordinates that are *not* in

[4]This dependence immediately forces the neighbourhood relation to be symmetric.

a given subset $A \subset \mathcal{I}$, a random field is a *Markov random field* (MRF) if the conditional distribution of any pixel given the other pixels only depends on the values of the neighbors of that pixel; i.e., for $i \in \mathcal{I}$,

$$\pi(x_i|\mathbf{x}_{-i}) = \pi(x_i|\mathbf{x}_{n(i)}) \,.$$

Markov random fields have been used for quite a while in imaging, not necessarily because images obey Markov laws but rather because these dependence structures offer highly stabilizing properties in modeling. Indeed, constructing the joint prior distribution of an image is a daunting task because there is no immediate way of describing the global properties of an image via a probability distribution. Just as for the directed acyclic graphs (DAG) models at the core of the BUGS software, using the full conditional distributions breaks the problem down to a sequence of *local* problems and this is therefore more manageable in the sense that we may be able to express more clearly how we think x_i behaves when the configuration of its neighbors is known.[5]

Before launching into the use of specific MRFs to describe prior assumptions on a given lattice, we need to worry[6] about the very existence of MRFs! Indeed, defining a set of full conditionals does not guarantee that there is a joint distribution behind them (Exercise 8.1). In our case, this means that general forms of neighborhoods and general types of dependences on the neighbors do not usually correspond to a joint distribution on $\mathbf{x}$.

We first obtain a representation that can be used for testing the existence of a joint distribution. Starting from a complete set of full conditionals on a lattice $\mathcal{I}$, if there indeed exists a corresponding joint distribution, $\pi(\mathbf{x})$, it is completely defined by the ratio $\pi(\mathbf{x})/\pi(\mathbf{x}^*)$ for a given fixed value $\mathbf{x}^*$ since the normalizing constant is automatically determined. Now, if $\mathcal{I} = \{1, \ldots, n\}$, it is simple to exhibit a full conditional density within the joint density by writing the natural decomposition

$$\pi(\mathbf{x}) = \pi(x_1|\mathbf{x}_{-1})\pi(\mathbf{x}_{-1})$$

and then to introduce $\mathbf{x}^*$ by the simple divide-and-multiply trick

$$\pi(\mathbf{x}) = \frac{\pi(x_1|\mathbf{x}_{-1})}{\pi(x_1^*|\mathbf{x}_{-1})}\,\pi(x_1^*, \mathbf{x}_{-1}) \,.$$

If we iterate this trick for all terms in the lattice (assuming we never divide by 0), we eventually get to the representation

[5]It is no surprise that computational techniques such as the Gibbs sampler stemmed from this area, as the use of conditional distributions is deeply ingrained in the imaging community.

[6]For those that do not want nor do not need to worry, the end of this section can be skipped, it being of a more theoretical nature and not used in the rest of the chapter.

$$\frac{\pi(\mathbf{x})}{\pi(\mathbf{x}^*)} = \prod_{i=0}^{n-1} \frac{\pi(x_{i+1}|x_1^*, \ldots, x_i^*, x_{i+2}, \ldots, x_n)}{\pi(x_{i+1}^*|x_1^*, \ldots, x_i^*, x_{i+2}, \ldots, x_n)}\,. \tag{8.1}$$

Hence, we can truly write the joint density as a product of ratios of its full conditionals modulo one renormalization.[7]

This result can also be used toward our purpose of checking for compatibility of the full conditional distributions: if there exists a joint density such that the full conditionals never cancel, then (8.1) must hold for every representation of $\mathcal{I} = \{1, \ldots, n\}$; that is, for every ordering of the indices, and for every choice of reference value $\mathbf{x}^*$. Although we cannot provide here the reasoning behind the result, there exists a necessary and sufficient condition for the existence of an MRF. This condition relies on the notion of *clique*: Given a lattice $\mathcal{I}$ and a neighbourhood relation $\sim$, a clique is a maximal subset of $\mathcal{I}$ made of sites that are all neighbors. The corresponding existence result (Cressie, 1993) is that an MRF associated with $\mathcal{I}$ and the neighbourhood relation $\sim$ necessarily is of the form

$$\pi(\mathbf{x}) \propto \exp\left(-\sum_{C\in\mathscr{C}} \Phi_C(\mathbf{x}_C)\right)\,, \tag{8.2}$$

where $\mathscr{C}$ is the collection of all cliques. This result amounts to saying that the joint distribution must separate in terms of its system of cliques.

We now embark on the description of two specific MRFs that are appropriate for image analysis, namely the *Ising model* used for binary images and its extension, the *Potts model*, used for images with more than two colors.

8.2.3 The Ising Model

If pixels of the image $\mathbf{x}$ under study can only take two colors (black and white, say, as in Fig. 8.1), $\mathbf{x}$ is binary. We typically refer to each pixel x_i as being *foreground* if $x_i = 1$ (black) and *background* if $x_i = 0$ (white). The conditional distribution of a pixel is then Bernoulli, with the corresponding probability parameter depending on the other pixels. A simplification step is to assume that it is a function of the number of black neighboring pixels, using for instance a logit link as $(j = 0, 1)$

$$\pi(x_i = j|\mathbf{x}_{-i}) \propto \exp(\beta n_{i,j})\,, \qquad \beta > 0\,, \tag{8.3}$$

where $n_{i,j} = \sum_{\ell\in n(i)} \mathbb{I}_{x_\ell = j}$ is the number of neighbors of x_i with color j. The *Ising model* is then defined via these full conditionals

$$\pi(x_i = 1|\mathbf{x}_{-i}) = \frac{\exp(\beta n_{i,1})}{\exp(\beta n_{i,0}) + \exp(\beta n_{i,1})}\,,$$

[7] This representation is by no means limited to MRFs: it holds for every joint distribution such that the full conditionals never cancel. It is called the *Hammersley–Clifford theorem*, and a two-dimensional version of it was introduced in Exercise 3.10.

and the joint distribution therefore satisfies

$$\pi(\mathbf{x}) \propto \exp\left(\beta \sum_{j\sim i} \mathbb{I}_{x_j=x_i}\right) , \tag{8.4}$$

where the summation is taken over all pairs (i, j) of neighbors (Exercise 8.17).

When inferring on β and thus simulating the posterior distribution β, we will be faced with a major obstacle, namely that the normalizing constant of (8.4), $Z(\beta)$, is intractable except for very small lattices $\mathcal{I}$, while depending on β. Therefore the likelihood function cannot be computed. We will introduce in Sect. 8.3 a computational technique called ABC that is intended to fight this very problem. At this early stage, however, we consider β to be known and focus on the simulation of $\mathbf{x}$ in preparation for the inference on both β and $\mathbf{x}$ given a noisy version of the image, $\mathbf{y}$, as presented in Sect. 8.4.

The computational conundrum of Ising models goes deeper as, due to the convoluted correlation structure of the Ising model, a direct simulation of $\mathbf{x}$ is not possible, expect in very specific cases. Faced with this difficulty, the image community very early developed computational tools which eventually led in 1984 to the proposal of the Gibbs sampler (Sect. 3.5.1).[8] The specification of Markov random fields and in particular of the Ising model implies the full conditional distributions of those models are available in closed form. The local structure of Markov random fields thus provides an immediate site-by-site update for the Gibbs sampler:

Algorithm 8.16 Ising Gibbs Sampler

```
Initialization: For $i \in \mathcal{I}$, generate independently

    $$x_i^{(0)} \sim \mathscr{B}(1/2) .$$

Iteration $t$ $(t \geq 1)$:
  1. Generate $\mathbf{u} = (u_i)_{i\in\mathcal{I}}$, a random ordering of the elements of $\mathcal{I}$.
  2. For $1 \leq \ell \leq |\mathcal{I}|$, update $n_{u_\ell,0}^{(t)}$ and $n_{u_\ell,1}^{(t)}$, and generate

    $$x_{u_\ell}^{(t)} \sim \mathscr{B}\left(\frac{\exp(\beta n_{u_\ell,1}^{(t)})}{\exp(\beta n_{u_\ell,0}^{(t)}) + \exp(\beta n_{u_\ell,1}^{(t)})}\right) .$$
```

In this implementation, the order of the updates of the pixels of $\mathcal{I}$ is random in order to overcome possible bottlenecks in the exploration of the distribu-

[8] The very name "Gibbs sampling" was proposed in reference to Gibbs random fields, related to the physicist Willard Gibbs. Interestingly, both of the major MCMC algorithms are thus named after physicists and were originally developed for problems that were beyond the boundaries of (standard) statistical inference.

tion, although this is not a necessary condition for the algorithm to converge. In fact, when considering two pixels x_1 and x_2 that are m pixels apart, the influence of a change in x_1 is not felt in x_2 before at least m iterations of the basic Gibbs sampler. Of course, if m is large, the dependence between x_1 and x_2 is quite moderate, but this slow propagation of changes is indicative of slow mixing in the Markov chain. For instance, to see a change of color of a relatively large homogeneous region is an event of very low probability, even though the distribution of the colors is exchangeable (Exercise 8.18).

↯ If β is large, the Ising distribution (8.4) is very peaked around both single color configurations. In such settings, the Gibbs sampler will face enormous difficulties to simply change the value of a single pixel.

Running Algorithm 8.16 in R is straightforward: opting for a four-neighbor relation, if we use the following function for the number of neighbors at (a, b),

```
xneig4=function(x,a,b,col){
n=dim(x)[1];m=dim(x)[2]
nei=c(x[a-1,b]==col,x[a,b-1]==col)
if (a!=n)
  nei=c(nei,x[a+1,b]==col)
if (b!=m)
  nei=c(nei,x[a,b+1]==col)
sum(nei)
}
```

the above Gibbs sampler can be written as

```
isingibbs=function(niter,n,m=n,beta){
  # initialization
  x=sample(c(0,1),n*m,prob=c(0.5,0.5),rep=TRUE)
  x=matrix(x,n,m)
  for (i in 1:niter){
    sampl1=sample(1:n)
    sampl2=sample(1:m)
    for (k in 1:n){
    for (l in 1:m){
     n0=xneig4(x,sampl1[k],sampl2[l],0)
     n1=xneig4(x,sampl1[k],sampl2[l],1)
     x[sampl1[k],sampl2[l]]=sample(c(0,1),1,
                 prob=exp(beta*c(n0,n1)))
     }}}
  x
  }
```

where `niter` is the number of times the whole matrix `x` is modified. (It should therefore be scaled against `n*m`, the size of `x`.) Figure 8.3 presents the output of simulations from Algorithm 8.16

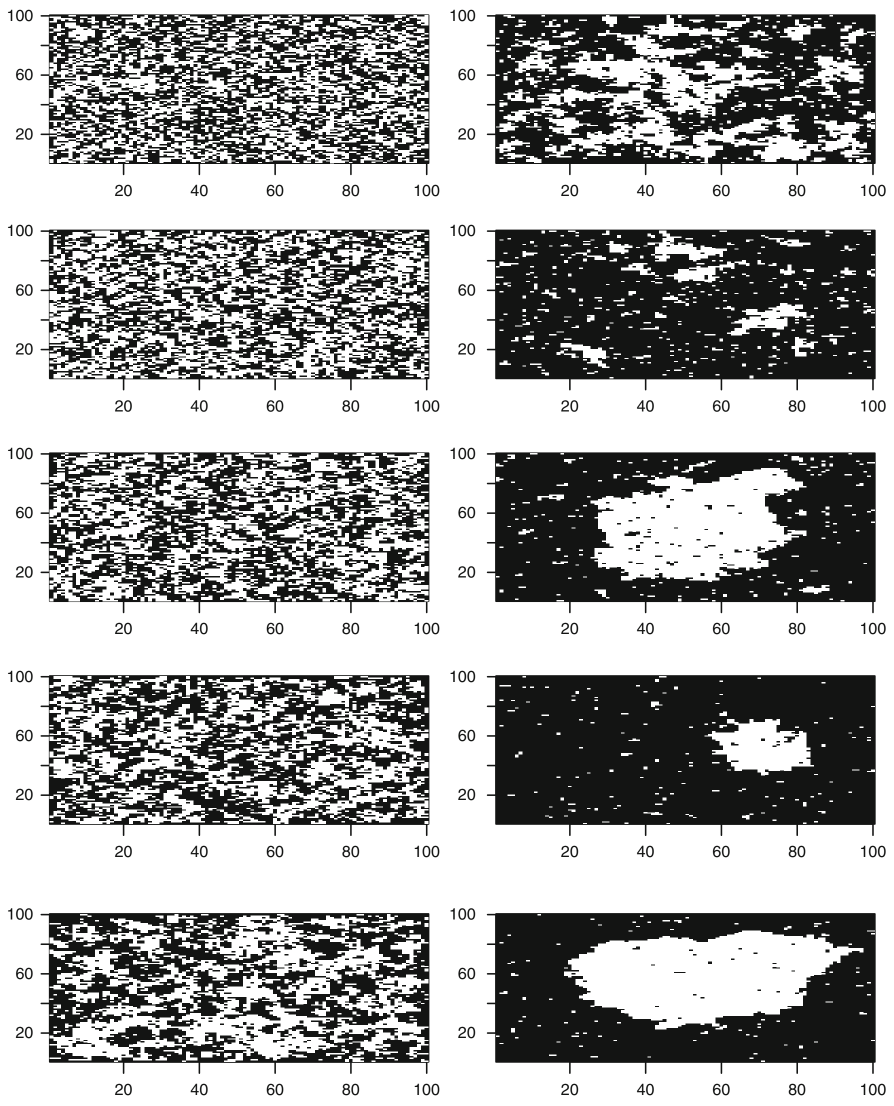

Fig. 8.3. Simulations from the Ising model with a four-neighbor neighbourhood structure on a 100×100 array after 1,000 iterations of the Gibbs sampler: β varies in steps of 0.1 from 0.3 to 1.2 (*first column, then second column*)

```
> image(1:100,1:100,isingibbs(10^3,100,100,beta))
```

for different values of β. Although we cannot discuss here convergence assessment for the Gibbs sampler (see Robert and Casella, 2009, Chap. 8), the images thus produced are representative of the Ising distributions: the larger

β, the more homogeneous the image (and also the slower the Gibbs sampler).[9] When looking at the result associated with the larger values of β, we can start to see the motivations for using such representations to model images like the **Menteith** dataset, discussed in Sect. 8.4.

Along with the slow dynamic induced by the single-site updating, we can point out another inefficiency of this algorithm, namely that many updates will not modify the current value of $\mathbf{x}$ simply because the new value of x_l is equal to its previous value! It is, however, straightforward to modify the algorithm so that it only proposes changes of values. The update of each pixel l is then a Metropolis–Hastings step with acceptance probability

$$\rho = \exp(\beta n_{l,1-x_l}) / \exp(\beta n_{l,x_l}) \wedge 1\,,$$

with the corresponding R function

```
isinghm=function(niter,n,m=n,beta){
  x=sample(c(0,1),n*m,prob=c(0.5,0.5),rep=TRUE)
  x=matrix(x,n,m)
  for (i in 1:niter){
   sampl1=sample(1:n)
   sampl2=sample(1:m)
   for (k in 1:n){
   for (l in 1:m){
    n0=xneig4(x,sampl1[k],sampl2[l],x[sampl1[k],sampl2[l]])
    n1=xneig4(x,sampl1[k],sampl2[l],1-x[sampl1[k],sampl2[l]])
    if (runif(1)<exp(beta*(n1-n0)))
      x[sampl1[k],sampl2[l]]=1-x[sampl1[k],sampl2[l]]
   }}}
  x
  }
```

Although the details are too involved to be included here, Liu (1996) has shown that this alternative is faster (to converge) than the original Gibbs sampler.

8.2.4 The Potts Model

The generalization of the Ising model to cases when the image has more than two colors, G say, is straightforward. If $n_{i,g}$ denotes the number of neighbors of $i \in \mathcal{I}$ with color g $(1 \le g \le G)$, that is,

[9] In fact, there exists a critical value of β, $\beta_c = 2.269185$ in the case of the four neighbor relation, such that, when $\beta > \beta_c$, the Markov chain converges to one of two different stationary distributions, depending on the starting point. In other words, the chain is no longer irreducible. In particle physics, this phenomenon is called *phase transition*.

$$n_{i,g} = \sum_{j\sim i} \mathbb{I}_{x_j=g}\,,$$

the full conditional distribution of x_i is chosen as

$$\pi(x_i = g|\mathbf{x}_{-i}) \propto \exp(\beta n_{i,g})\,.$$

This choice corresponds to a (true) joint probability model, the *Potts model*, whose density is given by (Exercise 8.6)

$$\pi(\mathbf{x}) \propto \exp\left(\beta \sum_{j\sim i} \mathbb{I}_{x_j=x_i}\right)\,. \tag{8.5}$$

This model is a clear generalization of the Ising model and it suffers from the same drawback, namely that the normalizing constant of this density—which is a function of β—is not available in closed form and thus hinders inference and the computation of the likelihood function.

Once again, we face the hindrance that, when simulating $\mathbf{x}$ from a Potts model with a large β, the single-site Gibbs sampler may be quite slow. More efficient alternatives are available, including the Swendsen–Wang algorithm (Exercise 8.7). For instance, Algorithm 8.17 below is again a Metropolis–Hastings algorithm that forces moves on the current values. Note the special feature that, while this Metropolis–Hastings proposal is *not* a random walk, using instead a uniform proposal on the $G-1$ other possible values still leads to an acceptance probability that is equal to the ratio of the target densities.

```
Algorithm 8.17 Potts Metropolis–Hastings Sampler
    Initialization: For $i \in \mathcal{I}$, generate independently
        $x_i^{(0)} \sim \mathscr{U}(\{1,\ldots,G\})$.
    Iteration $t$ ($t \geq 1$):
     1. Generate $\mathbf{u} = (u_i)_{i\in\mathcal{I}}$ a random ordering of the elements of $\mathcal{I}$.
     2. For $1 \leq \ell \leq |\mathcal{I}|$,
            generate
                $\tilde{x}_{u_\ell} \sim \mathscr{U}(\{1,2,\ldots,x_{u_\ell}^{(t-1)}-1,x_{u_\ell}^{(t-1)}+1,\ldots,G\})$,
            compute the $n_{u_l,g}^{(t)}$ and
                $\rho_l = \left\{\exp(\beta n_{u_\ell,\tilde{x}_{u_\ell}})/\exp(\beta n_{u_\ell,x_{u_\ell}}^{(t)})\right\} \wedge 1$,
            and set $x_{u_\ell}^{(t)}$ equal to $\tilde{x}_{u_\ell}$ with probability $\rho_l$.
```

Figure 8.4 illustrates the result of a simulation using Algorithm 8.17 in a situation where there are $G = 4$ colors, using the following R function

```
pottshm=function(ncol=2,niter=10^4,n,m=n,beta=0){
x=matrix(sample(1:ncol,n*m,rep=TRUE),n,m)
for (i in 1:niter){
  sampl=sample(1:(n*m))
  for (k in 1:(n*m)){
    xcur=x[sampl[k]]
    a=(sampl[k]-1)%%n+1
    b=(sampl[k]-1)%/%n+1
    xtilde=sample((1:ncol)[-xcur],1)
    acpt=beta*(xneig4(x,a,b,xtilde)-xneig4(x,a,b,xcur))
    if (log(runif(1))<acpt) x[sampl[k]]=xtilde
    }}
return(x)
}
```

for the simulation. (The use of a single vector of indices for rows and columns is a programming trick that removes a loop in the code and thus saves a considerable amount of computing time. This also allows a true uniform distribution in `sampl`. Note the call to the congruential operators `%%` for modulo and `%/%` for integer division) We point out the reinforced influence of large β's on Fig. 8.4: not only is the homogeneity higher, but there is also a larger differentiation in the colors.[10] We stress that, while β in Fig. 8.4 ranges over the same values as in Fig. 8.3, the β's are not directly comparable since the larger number of classes in the Potts model induces a smaller value of the $n_{i,g}$'s for the neighbourhood structure.

8.3 Handling the Normalizing Constant

While simulating random variables distributed *from* a Potts model is required in several settings, one of which we will cover in the next section, a more common *statistical* setting is observing $\mathbf{x}$ distributed as

$$f(\mathbf{x}|\beta) = \frac{1}{Z(\beta)} \exp\left(\beta \sum_{j\sim i} \mathbb{I}_{x_j=x_i}\right) , \tag{8.6}$$

where $Z(\beta)$ is the normalizing constant of the density in $\mathbf{x}$, and inferring upon the parameter β, using for instance a uniform prior $\beta \sim \mathscr{U}(0,2)$.[11]

[10] Similar to the Ising model mentioned in Footnote 9, there also exist a phase transition phenomenon and a critical value for β in this model.

[11] The upper bound on β in the above prior is chosen for a very precise reason: As mentioned in the previous footnotes, when $\beta \geq 2$, the Potts model associated with a four-neighbor relation is almost surely concentrated on single-color images. It is thus pointless to consider larger values of β.

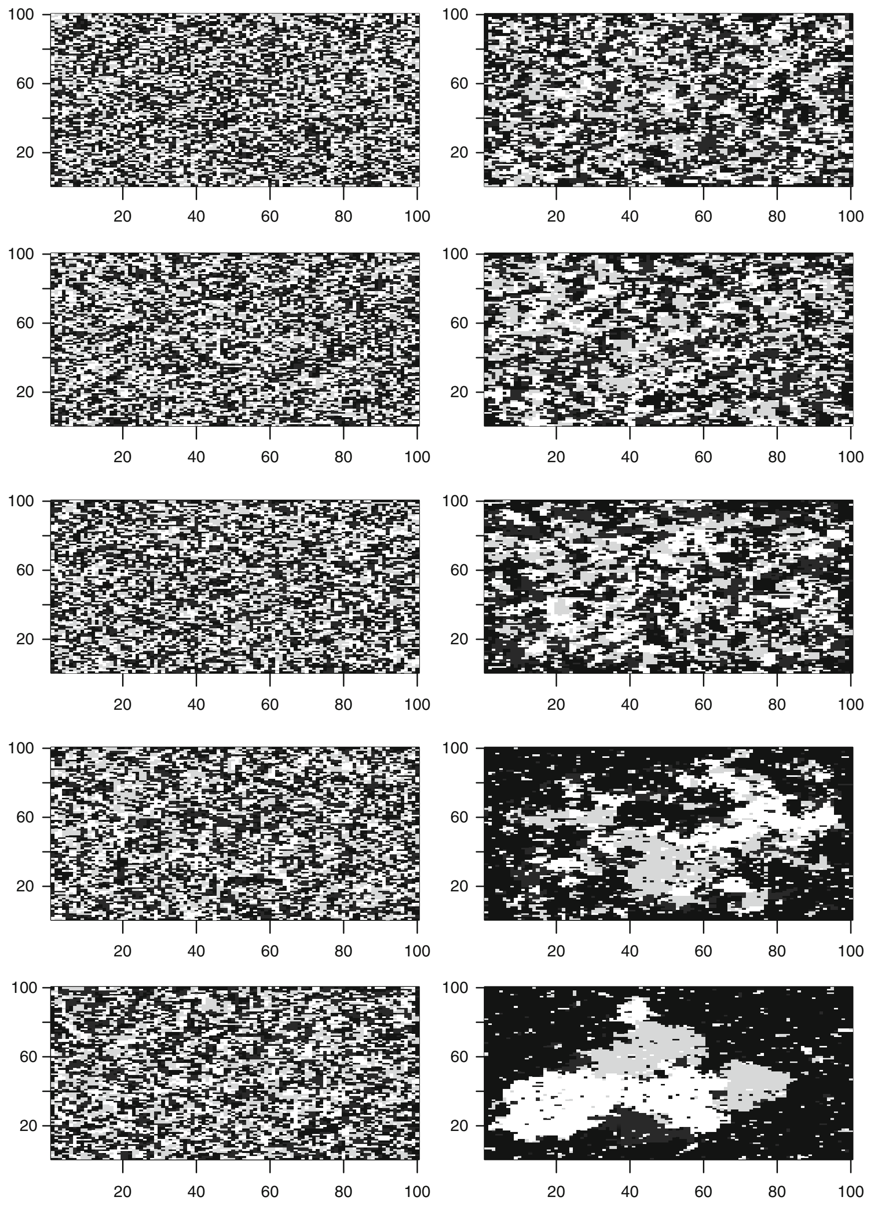

Fig. 8.4. Simulations from the Potts model with four grey levels and a four-neighbor neighbourhood structure based on 1,000 iterations of the Metropolis–Hastings sampler. The parameter β varies in steps of 0.1 from 0.3 to 1.2 (*first column, then second column*)

The primary computational difficulty with this inference is the unavailability of the normalizing constant

$$Z(\beta) = \sum_{\mathbf{x}} \exp\{\beta S(\mathbf{x})\} ,$$

where $S(\mathbf{x}) = \sum_{i\in\mathcal{I}} \sum_{j\sim i} \mathbb{I}_{x_j=x_i}$. The above summation operates over the $G^{|\mathcal{I}|}$ possible values of $\mathbf{x}$, where $|\mathcal{I}|$ denotes the size of $\mathcal{I}$. It involves too many terms to be manageable. In the case of the Ising model, the number of terms in the sum is for instance 2 to the power the number of points in the lattice. For a small 256×256 black-and-white image, there are therefore 2^{65536} terms in the sum! Furthermore, this is not a setting where a standard MCMC solution would apply because of the same difficulty: a Metropolis–Hastings algorithm also requires the evaluation of the ratio $Z(\tilde{\beta})/Z(\beta)$ in the acceptance probability. Unsurprisingly, addressing the approximation of $Z(\beta)$ has given rise to a huge literature, as shown by Ripley (1988) and Rue and Held (2005), but the solutions are mostly too convoluted for this book (see, e.g., the auxiliary variable method of Møller et al., 2006). We first describe a semi-practical resolution of this difficulty, called *path sampling*, which is costly in computing time for large images, before moving to a more generic if less precise solution.

8.3.1 Path Sampling

The path sampling technique is based on a derivative representation of the normalizing constant. Since

$$\frac{\mathrm{d}Z(\beta)}{\mathrm{d}\beta} = \sum_{\mathbf{x}} S(\mathbf{x}) \exp(\beta S(\mathbf{x})) ,$$

we can express this derivative as an expectation under $\pi(\mathbf{x}|\beta)$,

$$\frac{\mathrm{d}Z(\beta)}{\mathrm{d}\beta} = Z(\beta) \sum_{\mathbf{x}} S(\mathbf{x}) \frac{\exp(\beta S(\mathbf{x}))}{Z(\beta)} = Z(\beta)\, \mathbb{E}_\beta[S(\mathbf{X})] ,$$

that is,

$$\frac{\mathrm{d}\log Z(\beta)}{\mathrm{d}\beta} = \mathbb{E}_\beta[S(\mathbf{X})] .$$

Therefore, the ratio $Z(\beta_1)/Z(\beta_0)$ can be represented as an integral,

$$\log\{Z(\beta_1)/Z(\beta_0)\} = \int_{\beta_0}^{\beta_1} \mathbb{E}_\beta[S(\mathbf{x})]\mathrm{d}\beta , \tag{8.7}$$

leading to the *path sampling identity* (see Chen et al., 2000, for many more details about this technique.)

Although (8.7) may not look like a considerable improvement, since we now have to compute an expectation in $\mathbf{x}$ plus an integral over β, the representation (8.7) is appealing because we can use standard simulation procedures for its approximation. First, for a given value of β, $\mathbb{E}_\beta[S(\mathbf{X})]$ can be approximated from an MCMC sequence simulated by Algorithm 8.17. Obviously, changing the value of β should involve a new simulation run, however the cost can be attenuated by using instead importance sampling for similar values of β. Second, the integral itself can be approximated by *numerical quadrature,* namely by computing the value of $f(\beta) = \mathbb{E}_\beta[S(\mathbf{X})]$ for a finite number of values of β and approximating $f(\beta)$ by a piecewise-linear function $\hat{f}(\beta)$ for the intermediate values of β. Indeed, for arbitrary β_0 and β_1,

$$\int_{\beta_0}^{\beta_1} f(\beta)\,\mathrm{d}\beta \approx \hat{f}(\beta_0) + \left\{\hat{f}(\beta_1) - \hat{f}(\beta_0)\right\} \frac{(\beta_1-\beta_0)^2}{2}\,,$$

where $\hat{f}(\beta)$ is approximated by the above Monte Carlo method.

The rendering of the above in R for **Laichedata** is as follows for a four-neighbor relation: the expectation $\mathbb{E}_\beta[S(\mathbf{X})]$ is approximated via the following R function

```
sumising=function(niter=10^3,numb,beta){
S=0
x=matrix(sample(c(0,1),numb^2,rep=TRUE),ncol=numb)
for (i in 1:niter){
  s=0
  sampl1=sample(1:numb)
  sampl2=sample(1:numb)
  for (k in 1:numb){
  for (l in 1:numb){
   n0=xneig4(x,sampl1[k],sampl2[l],x[sampl1[k],sampl2[l]])
   n1=xneig4(x,sampl1[k],sampl2[l],1-x[sampl1[k],sampl2[l]])
   if (log(runif(1))<(beta*(n1-n0))){
     x[sampl1[k],sampl2[l]]=1-x[sampl1[k],sampl2[l]]
     n0=n1}
   s=s+n0
   }}
  if (2*i>niter)
    S=S+s
  }
return(2*S/niter)
}
```

for a few selected values of β, while the whole function $f(\beta)$ is then approximated using the R procedure `approxfun` as

```
Z=seq(0,2,by=.1)
for (i in 1:21)
```

```
    Z[i]=sumising(numb=24,beta=Z[i])
  lrcst=approxfun(seq(0,2,0.1),Z)
```

This approximation is illustrated by Fig. 8.5. The ratio of the constants, $Z(\tilde{\beta})/Z(\beta)$ is provided by the R numerical integration function, `integrate`, as

```
  Zratio=integrate(lrcst,betatilde,beta)$value
```

and can be easily inserted within a random walk Metropolis–Hasting algorithm. Indeed, now that we have painstakingly constructed a satisfactory

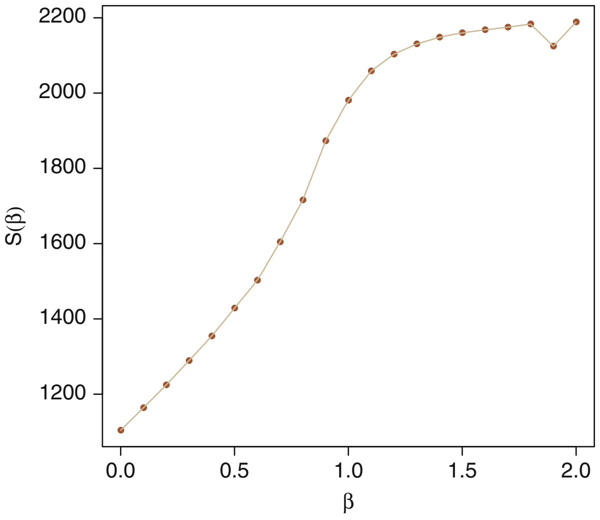

Fig. 8.5. Monte Carlo approximation of $\mathbb{E}_\beta[S(\mathbf{X})]$ for a 24×24 Ising model, based on 10^3 iterations. The irregularity at the penultimate value of β can be attributed to a failed convergence of the Gibbs sampler

approximation of $Z(\beta_1)/Z(\beta_0)$ for any arbitrary pair (β_0, β_1), we can run an MCMC sampler targeting the posterior distribution $\pi(\beta|\mathbf{x})$, where simulation at iteration t is based on the proposal

$$\tilde{\beta} \sim \mathscr{U}([\beta^{(t-1)} - h, \beta^{(t-1)} + h]) ;$$

that is, a uniform move with range $2h$. The acceptance ratio associated with the pair $(\beta^{(t-1)}, \tilde{\beta})$ is thus given by

$$1 \wedge \left(\hat{Z}(\beta^{(t-1)})/\hat{Z}(\tilde{\beta})\right) \exp\left\{(\tilde{\beta} - \beta^{(t-1)})S(\mathbf{x})\right\} ,$$

which translates into the R code

```
betatilde=beta[t-1]+runif(1,-0.05,0.05)
laccept=lvr*(betatilde-beta[t-1])+integrate(lrcst,
        betatilde,beta[t-1])$value
if (runif(1)<exp(laccept)){
  beta[t]=betatilde}else{
  beta[t]=beta[t-1]}
```

The outcome of this MCMC algorithm is represented by the histogram of Fig. 8.6, which exhibits a very regular posterior distribution for β, which is symmetric around 0.47. Thanks to the path sampling approximation to $Z(\beta)$, running 10^5 iterations is almost instantaneous.

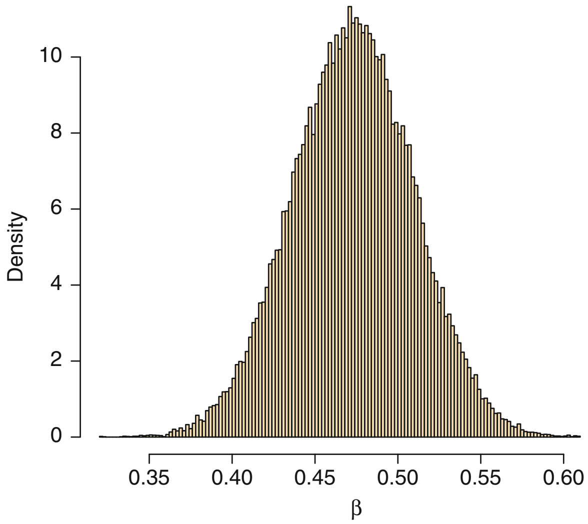

Fig. 8.6. Dataset **Laichedata**: Histogram of the MCMC sample of β's produced using the path sampling approximation to the ratio $Z(\tilde{\beta})/Z(\beta)$, when based on 10^5 iterations

8.3.2 The ABC Method

In a general setting where the likelihood function is not available in a closed form, the trick at the core of the path sampling technique is not always available. (Consider for instance the case of a multivariate β.) We thus need to turn towards faster if more rudimentary approximations and a method of choice is the ABC (approximate Bayesian computation) technique, introduced by Pritchard et al. (1999) in population genetic settings.

The method starts from a valid rejection technique bypassing the computation of the likelihood function. Namely, if we observe $x \sim f(x|\theta)$

and if $\pi(\theta)$ is the prior distribution on the parameter θ, then an algorithm that jointly simulates

$$\theta' \sim \pi(\theta) \quad \text{and} \quad y \sim f(y|\theta')$$

and accepts the simulated θ' if, and only if, the auxiliary variable y is equal to the observed value,

$$x = y\,,$$

is exact in the sense that the accepted θ''s are distributed from the posterior. Obviously, the algorithm is not practical in cases when x is continuous or even takes a large enough number of values.[12] In most standard occurrences, the ABC algorithm starts with an approximation, in the sense that the equality constraint $x = y$ is replaced with a tolerance condition, $\varrho(x, y) \leq \epsilon$, where ϱ is a measure of discrepancy between x and y. We will call $\epsilon > 0$ the tolerance bound and ϱ will be chosen as a distance between summary statistics. The output of the ABC algorithm is then distributed from the distribution with density proportional to

$$\pi(\theta)\,\mathbb{P}_\theta(\varrho(x, y) \leq \epsilon|x)\,,$$

where the probability is associated with $y \sim f(y|\theta)$. This density is denoted by $\pi_\epsilon(\theta \mid x)$.

If the tolerance ϵ is "too large", the approximation is poor; to understand why, consider that, when ϵ goes to ∞, the ABC algorithm amounts to simulating from the prior since all simulations are accepted. If ϵ is sufficiently small, $\pi_\epsilon(\theta|x)$ is a good approximation of $\pi(\theta|x)$, but the acceptance probability may be too low for this value to be practical. Selecting the "right" ϵ is thus crucial. It is customary to pick ϵ as an empirical quantile of $\varrho(x, y)$ when y is simulated from the marginal

$$\int \pi(\theta) f(y|\theta)\mathrm{d}\theta$$

and the choice is often the corresponding 1 % quantile. This quantile is easily approximated by simulation.

In settings when the data x has a large dimension, the ABC algorithm uses instead a distance between summary statistics $\varrho(S(x), S(y))$ rather than a distance between x and y. This choice throws away some information contained in the data about θ, but it also allows to concentrate on important features of the data in order to bring a maximal discrimination between the observed and the simulated statistics. It is thus rarely the case that S is a

[12]Note that, for **Laichedata**, it is possible to wait for the equality $S(x) = S(y)$ with a sufficiently high probability. In that case, since S is a sufficient statistic, we are simulating from the *exact* posterior.

sufficient statistic.[13] In the general case, the output of the ABC algorithm is therefore a simulation from the distribution $\pi_\epsilon(\theta \mid x)$. The ABC algorithm thus reads as follows:

Algorithm 8.18 ABC algorithm For $i = 1, \ldots, N$,

```
1. Generate $\theta_i$ from the prior $\pi$.
2. Generate $y_i$ from the model distribution $f(x|\theta_i)$.
3. Compute the distance $\varrho(S(y_i), S(x))$.
Deduce $\epsilon$ as the $1\%$ quantile of the distances. Accept the $\theta_i$'s such that $\varrho(S(x), S(y_i)) \le \epsilon$.
```

To illustrate the ABC method in a simple environment, consider the problem already processed in Chap. 2 about assessing whether a normal $\mathscr{N}(\mu, \sigma^2)$ distribution has a zero mean, $\mu = 0$. As explained in Sect. 2.3.1, the natural Bayesian approach is to include the model index $\mathfrak{M}$ as an extra parameter taking only the values 1 (when $\mu = 0$) and 2 (when $\mu \neq 0$). In other words, Bayesian inference covers the pair $(\mathfrak{M}, \theta)$, conditional on the data $\mathscr{D}_n$. Simulating by ABC from the posterior on $(\mathfrak{M}, \theta)$ given $\mathscr{D}_n$ then follows from Algorithm 8.18:

1. Generate $\mathfrak{M}^i$ uniformly at random on $\{1, 2\}$ $(i = 1, \ldots, n)$.
2. Generate θ_i from the prior $\pi(\theta|\mathfrak{M}^i)$ $(i = 1, \ldots, n)$.
3. Generate $\mathscr{D}_n^i$ from the normal model indexed by $(\mathfrak{M}^i, \theta_i)$ $(i = 1, \ldots, n)$.
4. Compute the distances between the statistics $(\bar{x}(\mathscr{D}_n), s^2(\mathscr{D}_n))$ and $(\bar{x}(\mathscr{D}_n^i), s^2(\mathscr{D}_n^i))$ $(i = 1, \ldots, n)$.
5. Deduce ϵ as the $1\,\%$ quantile of the distances.
6. Accept the $\mathfrak{M}^i$'s for which the distances are less than ϵ.

The distance we pick is inspired from the likelihood function, namely

$$\begin{aligned}\varrho\{(\bar{x}(\mathscr{D}_n), s^2(\mathscr{D}_n)), (\bar{x}(\mathscr{D}_n^*), s^2(\mathscr{D}_n^*))\} &= n\{\bar{x}(\mathscr{D}_n) - \bar{x}(\mathscr{D}_n^*)\}^2 \\ &\quad + \{s^2(\mathscr{D}_n)/s^2(\mathscr{D}_n^*)\} - 1 - \log\{s^2(\mathscr{D}_n)/s^2(\mathscr{D}_n^*)\}\,.\end{aligned}$$

The implementation is then straightforward: we select one of the models at random, simulate from the corresponding (necessarily proper) prior on the parameter(s) and create a normal sample $\mathscr{D}_n^*$. The posterior probability of the model associated with $\mu = 0$ is then estimated by the proportion of accepted simulations from the simpler model. Under an $\mathscr{E}(1)$ prior on σ^2 in both models and a $\mathscr{N}(0, \sigma^2)$ on μ under the larger model, with the **normaldata** benchmark, the R code goes as follow:

[13]The setting of Markov random fields like the Ising and the Potts models is an exception in that it allows for a sufficient statistic, while being intractable via classical approaches.

```
> xbar=mean(normaldata)
> s2=(n-1)*var(normaldata)
> Nsim=10^6                #simulations from the prior
> indem=sample(c(0,1),Nsim,rep=TRUE)
> ssigma=1/rexp(Nsim)
> smu=rnorm(Nsim)*sqrt(ssigma)*(indem==1)
> ss2=s2/(ssigma*rchisq(Nsim,n-1))
> sobs=n*(rnorm(Nsim,smu,sqrt(ssigma/n))-xbar)^2+
+ ss2-1-log(ss2)
> epsi=quantile(sobs,.001)  #bound and selection
> prob=sum(indem[sobs<=epsi]==0)/(0.001*Nsim)
> (1-prob)/prob
[1] 0.1574074
```

producing a numerical value to be compared with the exact Bayes factor

$$(n+1)^{-1/2}\left[\frac{n\bar{x}^2+s^2+2}{n\bar{x}^2/(n+1)+s^2+2}\right]^{n+2/2}$$

(deduced from the derivation on page 45 by modifying for the exponential prior), which is equal to 0.1369 for **normaldata**. Figure 8.7 represents the variability of the ABC approximation compared with the true value.

8.3.3 Inference on Potts Models

If we consider the specific case of the posterior distribution associated with (8.6) and a uniform prior, Algorithm 8.18 simulates values of β uniformly over

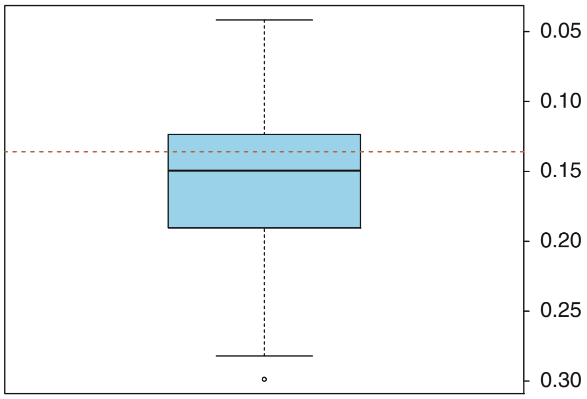

Fig. 8.7. Dataset **normaldata**: Boxplot representation of the ABC approximation to the Bayes factor, which true value is represented by an *horizontal line*, based on 10^5 proposals, a 1 % acceptance rate, and 500 replications

$(0, 2)$ and then values $\mathbf{x}$ from the Potts model (8.6). Simulating a data set $\mathbf{x}$ is unfortunately non-trivial for Markov random fields and in particular for Potts models, as we already discussed. While there exist developments towards this goal in the special case of the Ising model—in the sense that they produce exact simulations, at a high computing cost—, we settle for using a certain number of steps of an MCMC sampler (for instance, Algorithm 8.17) updating one clique at a time conditional on the others. Obviously, this solution brings a further degree of approximation into the picture in that running a fixed number of iterations of the MCMC sampler does not produce an exact simulation from (8.6). There is however little we can do about this if we want to use ABC. (And we can further argue that ABC involves such a significant departure from the exact posterior that an imperfect MCMC simulation does not matter so much!)

Since, for every new value of β, the algorithm runs a full MCMC simulation, we need to discuss the choice of the starting value as well. There are (at least) three natural solutions:

– start completely at random;
– start from the previously simulated $\mathbf{x}$.
– always start from the observed value $\mathbf{x}_0$;

The first one is the closest to the MCMC idea and it produces independent outcomes. The second solution is less compelling as the continuity it creates between draws is not statistically meaningful, given that the simulated β's change (independently or not) from one step to the other. The third solution offers the appealing feature of connecting with the observed value $\mathbf{x}_0$, thus favoring proximity between the simulated and the observed values, but this feature could confuse the issues in that this proximity may be due to a poor mixing of the chain rather than to a proper choice for β. (For instance, in the extreme case the MCMC chain does not move from $\mathbf{x}_0$, $\mathbf{x} = \mathbf{x}_0$ does not mean that the simulated β is at all interesting for $\pi(\beta|\mathbf{x}_0)\ldots$) The distance used in step 3 of Algorithm 8.18 is the (natural) absolute difference between the sufficient statistics $S(\mathbf{x})$ and $S(\mathbf{x}_0)$, with

$$S(\mathbf{x}) = \sum_{i\sim j} \mathbb{I}_{x_i=x_j} \,.$$

For the four-neighbour relation, the statistic can be computed directly without loops as

```
sum(x[-1,]==x[-n,])+sum(x[,-1]==x[,-m])
```

and the whole R code corresponding to a random start of the Metropolis–Hastings algorithm is as follows:

```
> ncol=4; nrow=10; Nsim=2*10^4; Nmc=10^2
> suf0=sum(x0[-1,]==x0[-nrow,])+sum(x0[,-1]==x0[,-nrow])
> outa=dista=rep(0,Nsim)
> for (tt in 1:Nsim){
+   beta=runif(1,max=2)
+   xprop=pottshm(ncol,nit=Nmc,n=nrow,beta=beta)
+   dista[tt]=abs(suf0-(sum(xprop[-1,]==xprop[-nrow,])+
+    sum(xprop[,-1]==xprop[,-ncol])))
+  outa[tt]=beta
+  }
betas=outa[order(dista)<=.01*Nsim]
```

Note the inequality sign `<=` and the use of `jitter` to get exactly `0.01*Nsim` values in the vector `beta`. This is due to the fact that the statistic S takes integer values.

When applying the above to the **Laichedata** dataset, we obtain the outcome represented in Fig. 8.8. When comparing with Fig. 8.6, we can check that ABC produces an almost exact representation, even though ϵ is not equal to zero. As mentioned above, it would actually be feasible to achieve $\epsilon = 0$ with a larger number of simulations.

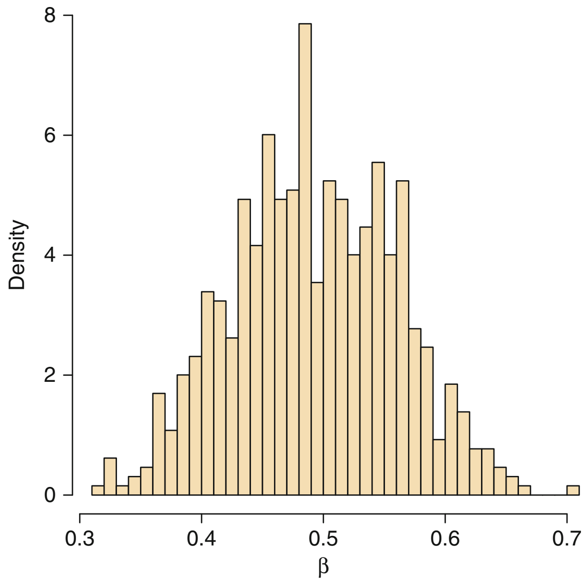

Fig. 8.8. Dataset **Laichedata**: Histogram of the sample of β's produced using an ABC algorithm with 10^4 iterations and a 1 % quantile on the difference between the sufficient statistics as its tolerance bound ϵ

8.4 Image Segmentation

In this section, we still consider images as statistical objects, but they are now "noisy" in the sense that the color or the grey level of a pixel is not observed exactly but with some perturbation (sometimes called *blurring* as in satellite imaging). The purpose of image segmentation is to cluster pixels into homogeneous classes without supervision or preliminary definition of those classes, based only on the spatial coherence of the structure.

This underlying structure of the "true" pixels is denoted by **x**, while the observed image is denoted by **y**. Both objects **x** and **y** are arrays, with each entry of **x** taking a finite number of values and each entry of **y** taking real values (for modeling convenience rather than reality constraints). We are thus interested in the posterior distribution of **x** given **y** provided by Bayes' theorem, $\pi(\mathbf{x}|\mathbf{y}) \propto f(\mathbf{y}|\mathbf{x})\pi(\mathbf{x})$. In this posterior distribution, the likelihood, $f(\mathbf{y}|\mathbf{x})$, describes the link between the observed image and the underlying classification; that is, it gives the distribution of the noise, while the prior $\pi(\mathbf{x})$ encodes beliefs about the (possible or desired) properties of the underlying image. Although, as in other chapters, we cannot provide the full story of Bayesian image segmentation, an excellent tutorial on Bayesian image processing based on a summer school course can be found in Hurn et al. (2003).

As indicated above, a proper motivation for image segmentation is satellite processing since images caught by satellites are often blurred, either because of inaccuracies in the instruments or transmission or because of clouds or vegetation cover between the satellite and the area of interest.

The **Menteith** dataset that motivates this section is a 100×100 pixel satellite image of the lake of Menteith, as represented in Fig. 8.9. The lake of Menteith is located in Scotland, near Stirling, and offers the peculiarity of being called "lake" rather than the traditional Scottish "loch." As shown by the image, there are several islands on this lake, one of which houses an ancient abbey. The purpose of analyzing this satellite dataset is to classify all pixels into one of six states in order to detect some homogeneous regions.

The model being introduced, we turn to the central issue, namely how to draw inference on the "true" image, **x**, given an observed noisy image, **y**. The prior on **x** is a Potts model with G categories,

$$\pi(\mathbf{x}|\beta) = \frac{1}{Z(\beta)} \exp\left(\beta \sum_{j\sim i} \mathbb{I}_{x_j=x_i}\right) ,$$

where $Z(\beta)$ is the (intractable, see Sect. 8.3) normalizing constant of the Potts model. Given **x**, we assume that the observations in **y** are independent normal random variables,

$$f(\mathbf{y}|\mathbf{x},\sigma^2,\mu_1,\ldots,\mu_G) = \prod_{i\in\mathcal{I}} \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(y_i-\mu_{x_i})^2\right\} .$$

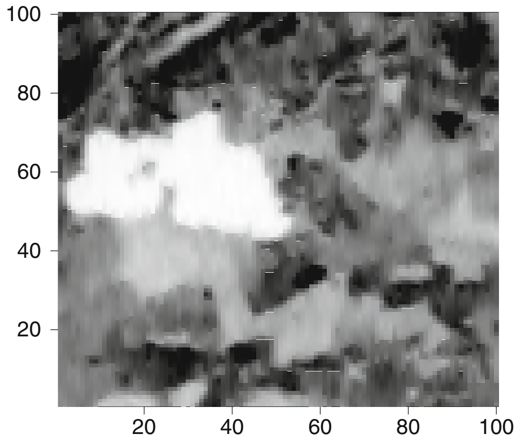

Fig. 8.9. Dataset **Menteith**: Satellite image of the lake of Menteith

This model is not exact in that the y_i's are integer grey levels that vary between 0 and 255, but it is easier to handle than a parameterized distribution on $\{0, \ldots, 255\}$. This setting is clearly reminiscent[14] of the mixture and hidden Markov models of Chaps. 6 and 7 in that a Markov structure, the Markov random field, is only observed through random variables indexed by the states.

In this problem, the parameters $\beta, \sigma^2, \mu_1, \ldots, \mu_G$ are usually considered to be *nuisance* parameters, a point of view that justifies the use of uniform priors like

$$\begin{aligned} \beta &\sim \mathscr{U}([0,2])\,, \\ \boldsymbol{\mu} = (\mu_1, \ldots, \mu_G) &\sim \mathscr{U}(\{\boldsymbol{\mu}\,;\, 0 \le \mu_1 \le \ldots \le \mu_G \le 255\})\,, \\ \pi(\sigma^2) &\propto \sigma^{-2} \mathbb{I}_{]0,\infty[}(\sigma^2)\,, \end{aligned}$$

the last prior corresponding to a uniform prior on $\log \sigma$.

The upper bound on β has been discussed in the previous section. The ordering of the μ_g's is not necessary, strictly speaking, but it avoids the label switching phenomenon discussed in Sect. 6.5. (The alternative is to use the same uniform prior on all μ_g's and then reorder them once the MCMC simulation is done. While this may avoid slow convergence behaviors in some cases, this strategy also implies more involved bookkeeping and higher storage requirements. In the case of large images, it simply cannot be considered.)

[14]Besides image segmentation, another typical illustration of such structures is *character recognition* where a machine scans handwritten documents, e.g., envelopes, and must infer a sequence of symbols (i.e., numbers or letters) from digitized pictures. Hastie et al. (2001) provide an illustration of this problem.

The corresponding posterior distribution is thus

$$\pi(\mathbf{x},\beta,\sigma^2,\boldsymbol{\mu}|\mathbf{y}) \propto \pi(\beta,\sigma^2,\boldsymbol{\mu}) \times \frac{1}{Z(\beta)} \exp\left(\beta \sum_{j\sim i} \mathbb{I}_{x_j=x_i}\right)$$
$$\times \prod_{i\in\mathcal{I}} \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{\frac{-1}{2\sigma^2}(y_i-\mu_{x_i})^2\right\} .$$

We can therefore construct the various full conditionals of this joint distribution with a view to the derivation of a hybrid Gibbs sampler for this model. First, the full conditional distribution of x_i $(i \in \mathcal{I})$ is $(1 \le g \le G)$

$$\mathbb{P}(x_i = g|\mathbf{y},\beta,\sigma^2,\boldsymbol{\mu}) \propto \exp\left\{\beta \sum_{j\sim i} \mathbb{I}_{x_j=g} - \frac{1}{2\sigma^2}(y_i-\mu_g)^2\right\} ,$$

which can be simulated directly, even though this is no longer a Potts model. As in the mixture and hidden Markov cases, once $\mathbf{x}$ is known, the groups associated with each category g separate and therefore the μ_g's can be simulated independently conditional on $\mathbf{x}$, $\mathbf{y}$, and σ^2. More precisely, if we denote by

$$n_g = \sum_{i\in\mathcal{I}} \mathbb{I}_{x_i=g} \quad \text{and} \quad s_g = \sum_{i\in\mathcal{I}} \mathbb{I}_{x_i=g} y_i$$

the number of observations and the sum of the observations allocated to category g, respectively, the full conditional distribution of μ_g is a truncated normal distribution on $[\mu_{g-1},\mu_{g+1}]$ (setting $\mu_0 = 0$ and $\mu_{G+1} = 255$) with mean s_g/n_g and variance σ^2/n_g. (Obviously, if no observation is allocated to this group, the conditional distribution turns into a uniform distribution on $[\mu_{g-1},\mu_{g+1}]$.) The full conditional distribution of σ^2 is an inverse gamma distribution with parameters $|\mathcal{I}|^2/2$ and $\sum_{i\in\mathcal{I}}(y_i-\mu_{x_i})^2/2$. Finally, the full conditional distribution of β is such that

$$\pi(\beta|\mathbf{y}) \propto \frac{1}{Z(\beta)} \exp\left(\beta \sum_{j\sim i} \mathbb{I}_{x_j=x_i}\right) , \tag{8.8}$$

since β does not depend on σ^2, $\boldsymbol{\mu}$, and $\mathbf{y}$, given $\mathbf{x}$. As discussed in Sect. 8.3.1, a path sampler can provide an approximation for the ratio of normalizing constants.

In the case of the **Menteith** data, we use a four-neighbour neighbourhood and $G = 6$ on a 100×100 image. For β ranging from 0 to 2 by steps of 0.1, the approximation to $f(\beta)$ is based on 15,000 iterations of Algorithm 8.17 (after burn-in), following the same procedure as with Fig. 8.5. The resulting piecewise-linear function is given in Fig. 8.10 and is smooth enough for us to consider the approximation as acceptable. (We use these numerical values in the clustering function `reconstruct` as the vector `dali`.) Note that the increasing nature of the function f in β is intuitive: As β grows, the probability of having more neighbors of the same category increases and so does $S(\mathbf{x})$.

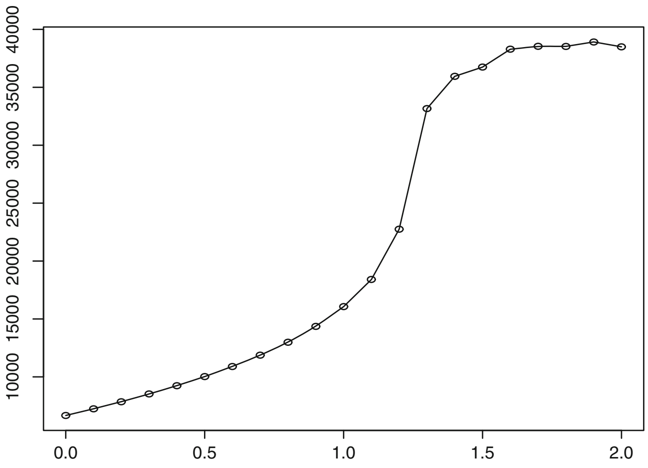

Fig. 8.10. Approximation of $f(\beta)$ for the Potts model on a 100×100 image, a four-neighbour neighbourhood, and $G = 6$, based on 1,500 MCMC iterations after burn-in

The corresponding R code for 6 colors and 4 neighbors (which are the specifications for the **Menteith** dataset) is as follows:

```
reconstruct=function(niter=10^3,y){
numb=dim(y)[1]
x=0*y
mu=matrix(0,niter,6)
sigma2=rep(0,niter)
#prior input
mu[1,]=c(35,50,65,84,92,120)
sigma2[1]=100
beta=rep(1,niter)
xcum=matrix(0,numb^2,6)
n=rep(0,6)
dali=c(6667.729,7245.159,7856.514,8523.00,9242.127,10025.211,
10896.380,11877.379,12985.344,14360.080,16062.470,18408.592,
22755.124,33163.207,35947.756,36745.675,38286.608,38534.912,
38531.211,38916.662,38495.781)
thefunc=approxfun(seq(0,2,length=21),dali)

for (i in 2:niter){
  lvr=0
  for (k in 1:numb){
    for (l in 1:numb){
      for (co in 1:6)
        n[co]=xneig4(x,k,l,co)
      x[k,l]=sample(1:6,1,prob=exp(beta[i-1]*n)*
             dnorm(y[k,l],mu[i-1,],sqrt(sigma2[i-1])))
      xcum[(k-1)*numb+l,x[k,l]]=xcum[(k-1)*numb+l,x[k,l]]+1
      lvr=lvr+n[x[k,l]]
      }}
  mu[i,1]=truncnorm(1,mean(y[x==1]),sqrt(sigma2[i-1]/
             sum(x==1)),0,mu[i-1,2])
  for (co in 2:5)
     mu[i,co]=truncnorm(1,mean(y[x==co]),sqrt(sigma2[i-1]/
             sum(x==co)),mu[i,co-1],mu[i-1,co+1])
  mu[i,6]=truncnorm(1,mean(y[x==6]),sqrt(sigma2[i-1]/
          sum(x==5)),mu[i,5],255)
  sese=sum((y-mu[i,1])^2*(x==1))
  for (co in 2:6)
    sese=sese+(y-mu[i,co])^2*(x==co))
  sigma2[i]=1/rgamma(1,numb^2/2,sese/2)
  betilde=beta[i-1]+runif(1,-0.05,0.05)
  laccept=vr*(betatilde-beta[i-1])+integrate(thefunc,
   betatilde,beta[i-1])$value
    integrate(lrcst,betilde,beta[i-1])$value
  if (log(runif(1))<laccept){
    beta[i]=betilde}else{beta[i]=beta[i-1]}
  }
list(beta=beta,mu=mu,sigma2=sigma2,xcum=xcum)
}
```

In the above, `truncnorm` is the standard simulator of a truncated normal variate based on the inverse cdf (see Robert and Casella, 2004, Chap. 2, for details).

In the case of the **Menteith** data, we use a four-neighbour neighbourhood and $G = 6$ on a 100×100 image. For β ranging from 0 to 2 by steps of 0.1, the approximation to $f(\beta)$ is based on 1,500 iterations of Algorithm 8.17 (after burn-in), following the same procedure as with Fig. 8.5. The resulting piecewise-linear function is given in Fig. 8.11 and is smooth enough for us to consider the approximation as acceptable. Note that the increasing nature of the function f in β is intuitive: As β grows, the probability of having more neighbors of the same category increases and so does $S(\mathbf{x})$.

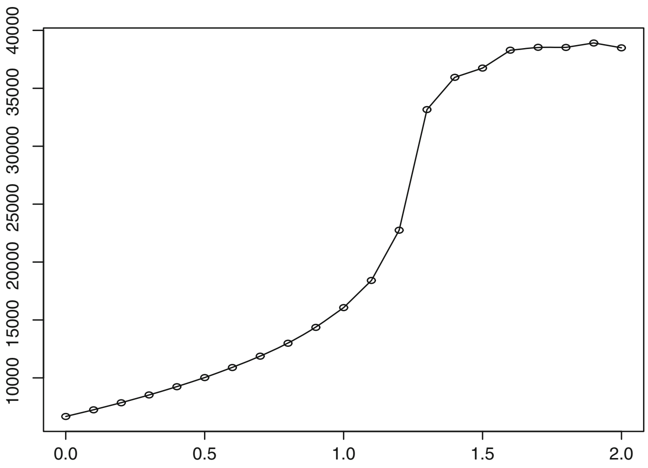

Fig. 8.11. Approximation of $f(\beta)$ for the Potts model on a 100×100 image, a four-neighbour neighbourhood, and $G = 6$, based on 1,500 MCMC iterations after burn-in

Figures 8.12–8.14 illustrate the convergence performances of the hybrid Gibbs sampler for **Menteith**. In that case, using $h = 0.05$ shows that 2,000 MCMC iterations are sufficient for convergence. (Recall, however, that $\mathbf{x}$ is a 100×100 image and thus that a single Gibbs step implies simulating the value of 10^4 pixels. This comes in addition to the cost of approximating the ratio of normalizing constants.)All histograms are smooth and unimodal, even though the moves on β are more difficult than for the other components. (Different values of h were tested for this dataset and none improved this behavior.) Note that large images like **Menteith** often lead to a very concentrated posterior on β. (Other starting values for β were also tested to check for the stability of the stationary region.)

We recall that the primary purpose of this image analysis is to clean (de-noise) and to classify into G categories the pixels of the image. Based on the MCMC output and in particular on the chain $(\mathbf{x}^{(t)})_{1\leq t\leq T}$ (where T is the number of MCMC iterations), an estimator of $\mathbf{x}$ needs to be derived through an evaluation of the consequences of wrong allocations. Two common ways of running this evaluation are either to count the number of (individual) pixel misclassification,

$$L_1(\mathbf{x},\hat{\mathbf{x}}) = \sum_{i\in\mathcal{I}} \mathbb{I}_{x_i\neq\hat{x}_i},$$

or to use the global "zero–one" loss function (see Sect. 2.3.1),

$$L_2(\mathbf{x},\hat{\mathbf{x}}) = \mathbb{I}_{\mathbf{x}\neq\hat{\mathbf{x}}}\,,$$

which amounts to saying that only a perfect reconstitution of the image is acceptable (and thus sounds rather extreme in its requirements). It is then easy to show that the estimators associated with these loss functions are the marginal posterior mode (MPM), $\hat{\mathbf{x}}^{MPM}$; that is, the image made of the pixels

$$\hat{x}_i^{MPM} = \arg\max_{1\leq g\leq G} \mathbb{P}^\pi(x_i = g|\mathbf{y})\,,\quad i\in\mathcal{I}\,,$$

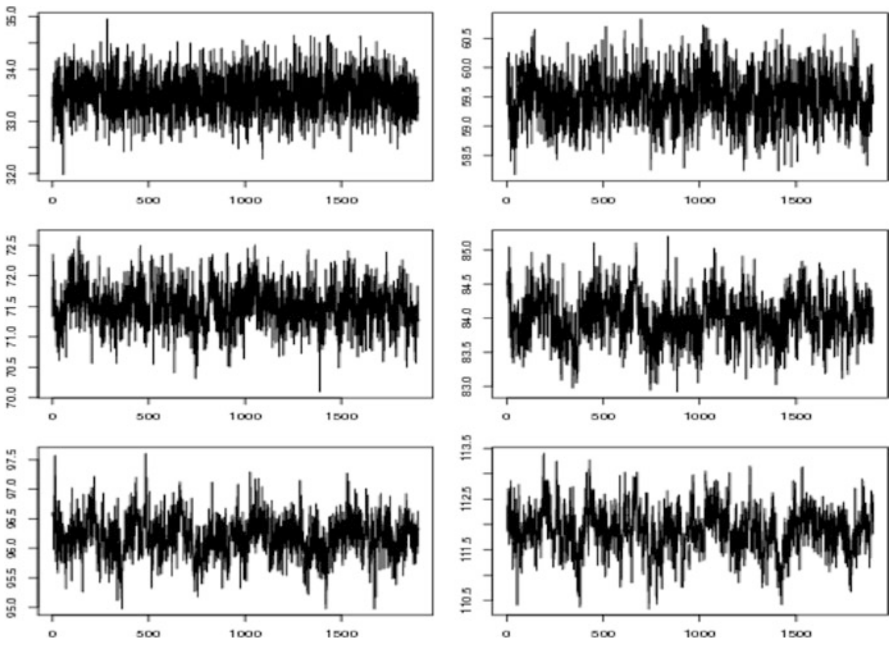

Fig. 8.12. Dataset **Menteith**: Sequence of μ_g's based on 2,000 iterations of the hybrid Gibbs sampler (*read row-wise from* μ_1 *to* μ_6)

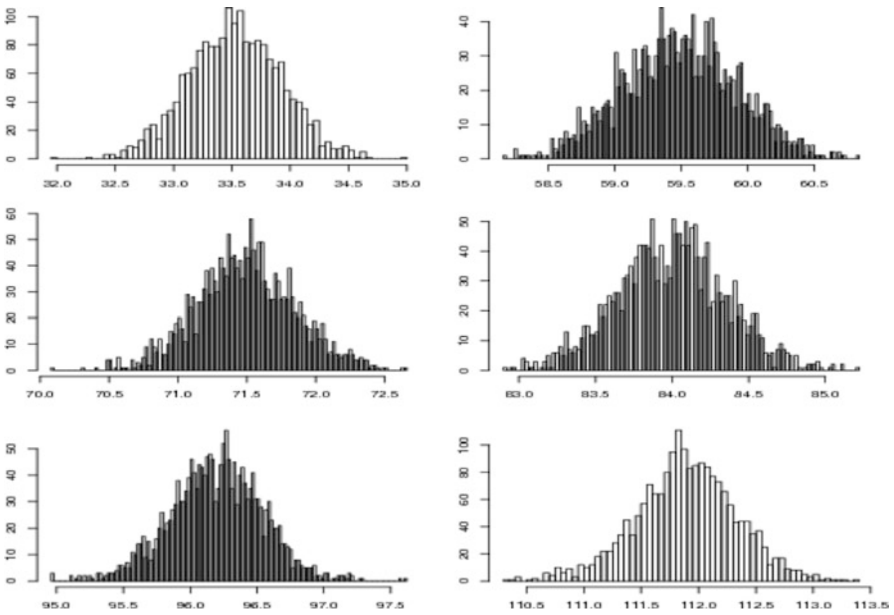

Fig. 8.13. Dataset **Menteith**: Histograms of the μ_g's represented in Fig. 8.12

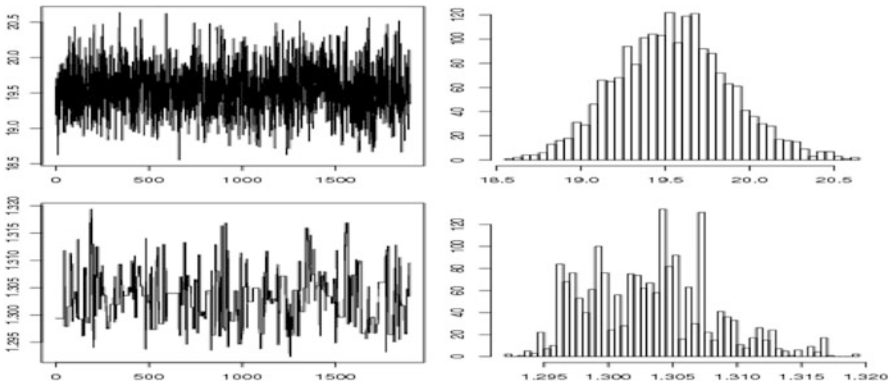

Fig. 8.14. Dataset **Menteith**: Raw plots and histograms of the σ^2's and β's based on 2,000 iterations of the hybrid Gibbs sampler (*the first row corresponds to* σ^2)

and the maximum a posteriori estimator (2.4),

$$\hat{\mathbf{x}}^{MAP} = \arg\max_{\mathbf{x}} \pi(\mathbf{x}|\mathbf{y}) \,,$$

respectively. Note that it makes sense that the $\hat{\mathbf{x}}^{MPM}$ estimator only depends on the marginal distribution of the pixels, given the linearity of the loss function. Both loss functions are nonetheless associated with image reconstruction rather than true classification (Exercise 8.14).

The estimators $\hat{\mathbf{x}}^{MPM}$ and $\hat{\mathbf{x}}^{MAP}$ obviously have to be approximated since the marginal posterior distributions $\pi(x_i|\mathbf{y})$ $(i \in \mathcal{I})$ and $\pi(\mathbf{x}|\mathbf{y})$ are not available in closed form. The marginal distributions of the x_i's being by-products of the MCMC simulation of $\mathbf{x}$, we can use, for instance, as an approximation to $\hat{\mathbf{x}}^{MPM}$ the most frequent occurrence of each pixel $i \in \mathcal{I}$,

$$\hat{x}_i^{MPM} = \max_{g \in \{1,\ldots,G\}} \sum_{j=1}^{N} \mathbb{I}_{x_i^{(j)}=g}\,,$$

based on a simulated sequence, $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}$, from the posterior distribution of $\mathbf{x}$. (This is not the most efficient approximation to $\hat{\mathbf{x}}^{MPM}$, obviously, but it comes as a cheap by-product of the MCMC simulation and it does not require the use of more advanced simulated annealing tools, mentioned in Sect. 6.7.)

Unfortunately, the same remark cannot be made about $\hat{\mathbf{x}}^{MAP}$: the state space of the simulated chain $(\mathbf{x}^{(t)})_{1\le t\le T}$ is so huge, being of cardinality $G^{100\times 100}$, that it is completely unrealistic to look for a proper MAP estimate out of the sequence $(\mathbf{x}^{(t)})_{1\le t\le T}$. Since $\pi(\mathbf{x}|\mathbf{y})$ is not available in closed form, even though this density could be approximated by

$$\hat{\pi}(\mathbf{x}|\mathbf{y}) \propto \sum_{t=1}^{T} \pi(\mathbf{x}|\mathbf{y}, \beta^{(t)}, \boldsymbol{\mu}^{(t)}, \sigma^{(t)})\,,$$

thanks to a Rao–Blackwellization argument, it is rather difficult to propose a foolproof simulated annealing that converges to $\hat{\mathbf{x}}^{MAP}$ (although there exist cheap approximations; see Exercise 8.15).

The segmented image of Lake Menteith is given by the MPM estimate that was found after 2,000 iterations of the Gibbs sampler. We reproduce in Fig. 8.15 the original picture to give an impression of the considerable improvement brought by the algorithm.

8.5 Exercises

8.1 Find two conditional distributions $f(x|y)$ and $g(y|x)$ such that there is no joint distribution corresponding to both f and g. Find a necessary condition for f and g to be compatible in that respect; i.e., to correspond to a joint distribution on (x, y).

8.2 Using the Hammersley–Clifford theorem, show that the full conditional distributions given by (8.3) are compatible with a joint distribution. Deduce that the Ising model is a Markov random field.

8.3 If a joint density $\pi(y_1, \ldots, y_n)$ is such that the conditionals $\pi(y_{-i}|y_i)$ never cancel on the supports of the marginals $m_{-i}(y_{-i})$, show that the support of π is equal to the Cartesian product of the supports of the marginals.

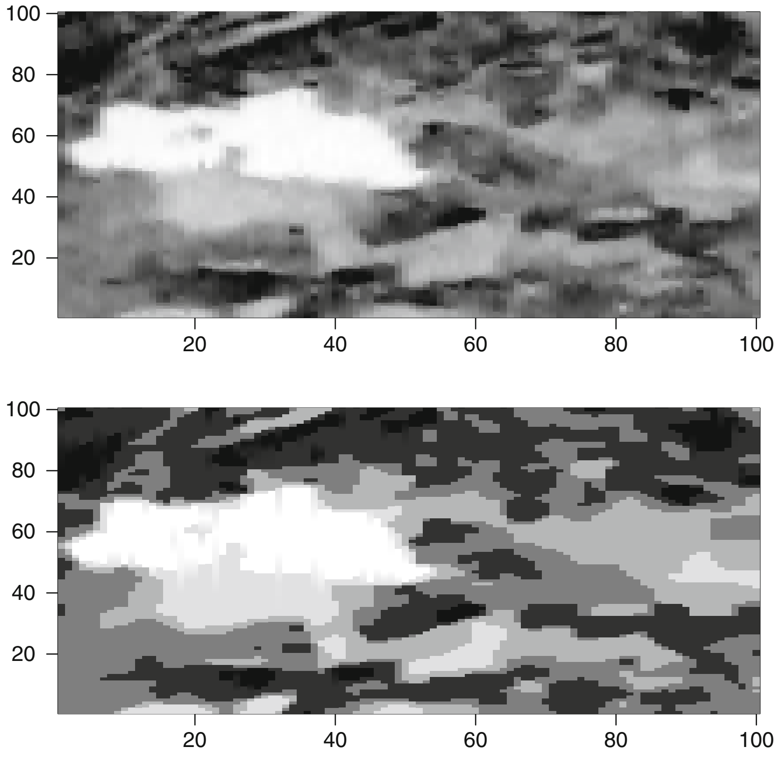

Fig. 8.15. Dataset **Menteith**: (*top*) Segmented image based on the MPM estimate produced after 2,000 iterations of the Gibbs sampler and (*bottom*) the observed image

8.4 Describe the collection of cliques $\mathcal{C}$ for an eight-neighbour neighbourhood structure such as in Fig. 8.2 on a regular $n \times m$ array. Compute the number of cliques.

8.5 Draw the function $Z(\beta)$ for a 3×5 array. Determine the computational cost of the derivation of the normalizing constant $Z(\beta)$ of (8.4) for an $m \times n$ array.

8.6 Show that the joint distribution (8.5) is indeed compatible with the full conditionals of the Potts model. Can you derive this joint distribution from the Hammersley–Clifford representation (8.1)?

8.7 For an $n \times m$ array $\mathcal{I}$, if the neighbourhood relation is based on the four nearest neighbors, show that the $x_{i,j}$'s for which $(i+j) \equiv 0 (\text{mod } 2)$ are independent conditional on the $x_{i,j}$'s for which $(i+j) \equiv 1 (\text{mod } 2)$ $(1 \le i \le n,\ 1 \le j \le m)$. Deduce that the update of the whole image can be done in two steps by simulating the pixels with even sums of indices and then the pixels with odd sums of indices. (This modification of Algorithm 8.16 is a version of *the Swendsen–Wang* algorithm.)

8.8 Determine the computational cost of the derivation of the normalizing constant of the distribution (8.5) for an $n \times m$ array and G different colors.

8.9 Use the Hammersley–Clifford theorem to establish that (8.5) is the joint distribution associated with the conditionals above. Deduce that the Potts model is an MRF.

8.10 Derive an alternative to Algorithm 8.17 where the probabilities in the multinomial proposal are proportional to the numbers of neighbors $n_{u_\ell,g}$ and compare its performance with that of Algorithm 8.17.

8.11 Show that the Swendsen–Wang improvement given in Exercise 8.7 also applies to the simulation of $\pi(\mathbf{x}|\mathbf{y},\beta,\sigma^2,\boldsymbol{\mu})$.

8.12 Using a piecewise-linear interpolation of $f(\beta)$ based on the values $f(\beta^1),\ldots,f(\beta^M)$, with $0<\beta_1<\ldots<\beta_M=2$, give the explicit value of the integral

$$\int_{\alpha_0}^{\alpha_1} \hat{f}(\beta)\,\mathrm{d}\beta$$

for any pair $0\le\alpha_0<\alpha_1\le 2$.

8.13 Show that the estimators $\hat{\mathbf{x}}$ that minimize the posterior expected losses $\mathbb{E}^\pi[L_1(\mathbf{x},\hat{\mathbf{x}})|\mathbf{y})]$ and $\mathbb{E}^\pi[L_2(\mathbf{x},\hat{\mathbf{x}})|\mathbf{y}]$ are $\hat{\mathbf{x}}^{MPM}$ and $\hat{\mathbf{x}}^{MAP}$, respectively.

8.14 Determine the estimators $\hat{\mathbf{x}}$ associated with two loss functions that penalize differently the classification errors,

$$L_3(\mathbf{x},\hat{\mathbf{x}})=\sum_{i,j\in\mathcal{I}}\mathbb{I}_{x_i=x_j}\,\mathbb{I}_{\hat{x}_i\neq\hat{x}_j}\quad\text{and}\quad L_4(\mathbf{x},\hat{\mathbf{x}})=\sum_{i,j\in\mathcal{I}}\mathbb{I}_{x_i\neq x_j}\,\mathbb{I}_{\hat{x}_i=\hat{x}_j}\,.$$

8.15 Since the maximum of $\pi(\mathbf{x}|\mathbf{y})$ is the same as that of $\pi(\mathbf{x}|\mathbf{y})^\kappa$ for every $\kappa\in\mathbb{N}$, show that

$$\pi(\mathbf{x}|\mathbf{y})^\kappa=\int\pi(\mathbf{x},\theta_1|\mathbf{y})\,\mathrm{d}\theta_1\times\cdots\times\int\pi(\mathbf{x},\theta_\kappa|\mathbf{y})\,\mathrm{d}\theta_\kappa\,, \tag{8.9}$$

where $\theta_i=(\beta_i,\boldsymbol{\mu}_i,\sigma_i^2)$ $(1\le i\le\kappa)$. Deduce from this representation an optimization scheme that slowly increases κ over iterations and that runs a Gibbs sampler for the integrand of (8.9) at each iteration.

8.16 For the Ising model, show that the distribution (8.4) can be also defined as

$$\pi(\mathbf{x})\propto\exp\left(2\beta\sum_{j\sim i}\mathbb{I}_{x_j=x_i=1}\right)$$

when the number of neighbors is constant.

8.17 Show that the joint distribution (8.4) can be obtained from the full conditionals (8.3) by virtue of the Hammersley–Clifford representation (8.1).

8.18 Show that the Ising distribution is symmetric in that inverting the color of all pixels does not change the probability (8.4).

8.19 For the Ising model, run a simulation experiment that should locate the limiting value of β above which almost all pixels are of the same color. Same question for the (negative) limiting value of β below which the image is a perfect checkerboard.

8.20 Show that the ABC algorithm implemented with $\epsilon=0$ and a distance between sufficient statistics is not approximate in that the output is truly simulated from the posterior distribution $\pi(\theta|\mathbf{x})\propto f(\mathbf{x}|\theta)\pi(\theta)$.