UNIVERSIDADE ESTADUAL DE CAMPINAS

Instituto de Matemática, Estatística e Computação Científica

VICTOR FREGUGLIA SOUZA

Bayesian selection of dependence structures for Markovian processes

Seleção bayesiana de estruturas de dependência para processos markovianos

Campinas

2022

Victor Freguglia Souza

Bayesian selection of dependence structures for Markovian processes

Seleção bayesiana de estruturas de dependência para processos markovianos

Tese apresentada ao Instituto de Matemática, Estatística e Computação Científica da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Doutor em Estatística.

Thesis presented to the Institute of Mathematics, Statistics and Scientific Computing of the University of Campinas in partial fulfillment of the requirements for the degree of Doctor in Statistics.

Supervisor: Nancy Lopes Garcia

Este trabalho corresponde à versão final da Tese defendida pelo aluno Victor Freguglia Souza e orientada pela Profa. Dra. Nancy Lopes Garcia.

Campinas

2022

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca do Instituto de Matemática, Estatística e Computação Científica
Ana Regina Machado - CRB 8/5467

So89b

Souza, Victor Freguglia, 1994-
Bayesian selection of dependence structures for Markovian processes / Victor Freguglia Souza. – Campinas, SP : [s.n.], 2022.

Orientador: Nancy Lopes Garcia.
Tese (doutorado) – Universidade Estadual de Campinas, Instituto de Matemática, Estatística e Computação Científica.

1. Teoria bayesiana de decisão estatística. 2. Campos aleatórios de Markov. 3. Cadeias de Markov. I. Garcia, Nancy Lopes, 1964-. II. Universidade Estadual de Campinas. Instituto de Matemática, Estatística e Computação Científica. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Seleção bayesiana de estruturas de dependência para processos markovianos
Palavras-chave em inglês:
Bayesian statistical decision theory
Markov random fields
Markov chains
Área de concentração: Estatística
Titulação: Doutor em Estatística
Banca examinadora:
Nancy Lopes Garcia [Orientador]
Helio dos Santos Migon
Florencia Graciela Leonardi
Jesus Enrique Garcia
Guilherme Vieira Nunes Ludwig
Data de defesa: 06-05-2022
Programa de Pós-Graduação: Estatística

Identificação e informações acadêmicas do(a) aluno(a)
- ORCID do autor: https://orcid.org/0000-0002-6189-4453
- Currículo Lattes do autor: http://lattes.cnpq.br/6306080581906246

Tese de Doutorado defendida em 06 de maio de 2022 e aprovada

pela banca examinadora composta pelos Profs. Drs.

Prof(a). Dr(a). NANCY LOPES GARCIA

Prof(a). Dr(a). HELIO DOS SANTOS MIGON

Prof(a). Dr(a). FLORENCIA GRACIELA LEONARDI

Prof(a). Dr(a). JESUS ENRIQUE GARCIA

Prof(a). Dr(a). GUILHERME VIEIRA NUNES LUDWIG

A Ata da Defesa, assinada pelos membros da Comissão Examinadora, consta no SIGA/Sistema de Fluxo de Dissertação/Tese e na Secretaria de Pós-Graduação do Instituto de Matemática, Estatística e Computação Científica.

Acknowledgements

This work was funded by Fundação de Amparo à Pesquisa do Estado de São Paulo - FAPESP grant 2017/25469-2.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Resumo

Este trabalho aborda o problema de seleção da estrutura de interação de uma classe de Campos Markovianos e de Cadeias de Markov de Alcance Variável sob uma perspectiva Bayesiana. A vizinhança de interação dos Campos Markovianos e a árvore de contexto das Cadeias de Markov de Alcance Variável são tratadas como objetos aleatórios não-observados de um sistema Bayesiano, e métodos de *Monte-Carlo Markov Chain* são propostos para gerar amostras desses objetos, construindo núcleos de proposta nesses espaços de objetos arbitrários que representam modelos, superando dificuldades computacionais instrínsicas desses modelos. Um ambiente completo para inferência em Campos Markovianos é proposto no pacote em R **mrf2d**. Os métodos propostos são aplicados à seleção da vizinhança de interaçao em um problema de análise de imagem de texture e na detecção de estados de renovação em uma Cadeia de Markov de textos escritos codificados.

Palavras-chave: Seleção Bayesiana de Modelos. Campos Markovianos. Cadeias de Markov de Alcance Variável.

Abstract

This work addresses the problem of selecting the interaction structure of a class of Markov Random Fields and Variable-Length Markov Chain models from a Bayesian perspective. The interaction neighborhood of the Markov Random Fields and the context tree of the Variable-Length Markov Chain model are treated as unobserved random objects of a Bayesian system, and Monte-Carlo Markov Chain methods are proposed to generate samples of these objects, constructing proposal kernels on these spaces of arbitrary objects that represent models, overcoming computational challenges intrinsic to these models. A complete framework for inference on Markov Random Fields is proposed in the R package **mrf2d**. The proposed methods are applied to selecting the interaction neighborhood in a texture image analysis problem and for detecting renewal states in a Markov Chain of encoded written texts.

Keywords: Bayesian Model Selection. Markov Random Fields. Variable-Length Markov Chains.

Contents

1 Introduction

Local dependence, referred as the Markov property, is a key assumption for modeling high-dimensional dependent data with a reduced mathematical complexity. This assumption allows us to describe the joint probability a high-dimensional random vector through a product of local interactions. Two popular probabilistic models with local dependence are Markov Chains, for sequence-indexed data, and Markov Random Fields for more general structures, for example, lattice data.

Throughout this work, we consider a Markov Random Field (MRF) model as a random vector indexed by a two-dimensional lattice, with local dependence based on a set of relative positions, as in Freguglia et al. (2020). This class of MRF models has many common definitions as particular cases, such as the texture models from Cross and Jain (1983) and Gimel'farb (1996), and the Potts model, used for example, in image segmentation problems (Zhang et al., 2001). Despite the decomposability of the probability distributions of MRFs, inference for such models is challenging due to the intractability of the normalizing constant that depends on parameters and is part of the Likelihood function. Typically, the Likelihood function must be approximated by using the Pseudolikelihood function (Besag, 1975) or using MCMC methods (Geyer and Thompson, 1992).

While MRFs have already proven useful for providing a probabilistic description of complex texture images, they make use of a complete neighborhood region, what can lead to an excessive number of unknown parameters to be estimated in inferential analyses. Standard hypothesis testing and model selection methods cannot be applied due to the unavailability of the Likelihood function and there are not many works dealing with model selection under Likelihood intractability, except for particular cases, for example, Ravikumar et al. (2010). Gimel'farb (1996) uses an heuristic method to search for smaller interaction structures to describe texture data, but the method may lead to poor results for some texture patterns.

Many strategies for Bayesian inference, in particular using Monte Carlo Markov Chain (MCMC) methods, have been developed for intractable Likelihood problems over the last few years. The main challenge for the use of MCMC methods is the intractability of the Likelihood function, that prevents the computation of posterior densities even up to a proportionality constant, because the intractable term of the Likelihood depends on the parameters. The three most common strategies for MCMC under intractability are substituting the Likelihood function for a tractable approximation, such as the Pseudolikelihood (Bouranis et al., 2017), using a Monte-Carlo approximation of the Likelihood function (Atchadé et al., 2013) and using additional random elements to the Markov Chain with analytical properties that remove the dependence on the intractable constant (Murray et al., 2012). In this work, we consider the Pseudolikelihood approach for MCMC methods because it can produce a direct approximation without producing better approximation in specific regions of the parameter space, which is an important factor for model selection since we would need better approximations on different regions for each model.

The Reversible Jump Monte Carlo Markov Chain (RJMCMC) introduced by Green (1995) is a powerful tool for turning MCMC into a variable selection procedure, by creating a Markov Chain on varying-dimensional spaces with a specified target invariant distribution over an arbitrary space that represents models and its parameters. This allows us to consider the interaction structure of a MRF as a random variable and generate a Markov Chain of interaction neighborhoods and parameter vectors associated with those neighborhoods jointly, then it is possible to make model-related decisions using the marginal posterior distribution of the interaction structures.

Even though RJMCMC and Likelihood approximations can be used for selection of the neighborhood structure in the MRF model, we can still think of this representation of models as an arbitrary random object in a Bayesian context as a more general framework, and apply the same concepts to other stochastic processes. On a Variable-Length Markov Chain (VLMC) model, the dependence structure of a Markov Chain with finite state space is represented by a *context tree* rather than a neighborhood and, due to the natural ordering of sequence data, the Likelihood function can be computed directly. Moreover, given a context tree, one can integrate the product of the Likelihood function and the transition probabilities prior distributions, especially when using conjugate priors, obtaining the marginal posterior distribution of a context tree up to a proportionality constant.

While obtaining a marginal measure for a context tree of a VLMC model in Bayesian context is useful for comparisons, the number of possible context trees grows at doubly exponential rate with their maximum depth considered, what causes the sum of marginal context tree measures to be intractable, preventing the direct computing the context tree posterior distribution. Kontoyiannis et al. (2020) proposes a Metropolis-Hastings algorithm to obtain a posterior sample of context trees under a specific choice of context tree and transition probabilities prior distributions. This strategy is aligned with the methodology proposed in Madigan et al. (1995), that constructs a Markov Chain based on random walks in a graph of models. The context tree posterior distribution sample allows us to perform all kinds of Bayesian inference decision regarding the model, based on Monte Carlo methods, like selecting the set of highest posterior probability context trees or computing measures such as the Intrinsic Bayes Factor (Berger and Pericchi, 1996) or the Posterior Bayes Factor (Aitkin, 1991) for hypothesis evaluation.

The variety of applications of MRFs, with particular properties used in each case, and the wide range of inferential methods resulted in the lack of proper software that can be used generally for implementing algorithms for the model. A common computational framework unifying all particular cases in a complete class of MRF models and the main tools required for creating new inference methods is highly useful. In this work we propose a software that can be used for implementing algorithms in a wide class of MRF models.

This thesis is composed by three self-contained research articles presented in the following three chapters. Chapter 2 (Freguglia and Garcia, 2022) introduces the **mrf2d** package, a complete inference framework for Markov Random Fields on two-dimensional lattices for the R programming language (R Core Team, 2020), with a detailed introduction to the MRF model considered and an in-depth description of the main functionalities available, with many practical examples such as image segmentation in neuroimaging data. Chapter 3 proposes a Bayesian model selection framework for MRF models, where the interaction structure, represented by a set of relative positions, is considered random with a given prior distribution. A RJMCMC algorithm is described and Pseudolikelihood is used as a proxy of the intractable Likelihood function to obtain Pseudoposteriors and the method is applied to a texture synthesis problem with textile image data. In Chapter 4 (Freguglia and Garcia, 2021) we use the same idea of treating models as random variables in the context of a VLMC model, being able to obtain posterior samples of random context trees given an observed sequence, with arbitrary prior distributions. These distributions are then used to estimate an Intrinsic Bayes Factor (Berger and Pericchi, 1996) and evaluate whether a specific state of the VLMC is a renewal state. The Intrinsic Bayes Factor method is applied to detecting renewal states in a dataset with sequences of encoded written texts in Portuguese. Final considerations are presented in Chapter 5 with a description of future works and extensions of the methodologies proposed in the articles.

2 Inference Tools for Markov Random Fields on Lattices: The R Package mrf2d

This chapter corresponds to the research article Freguglia and Garcia (2022) published on the Journal of Statistical Software on January of 2022.

Abstract: Markov random fields on two-dimensional lattices are behind many image analysis methodologies. mrf2d provides tools for statistical inference on a class of discrete stationary Markov random field models with pairwise interaction, which includes many of the popular models such as the Potts model and texture image models. The package introduces representations of dependence structures and parameters, visualization functions and efficient (C++-based) implementations of sampling algorithms, common estimation methods and other key features of the model, providing a useful framework to implement algorithms and working with the model in general. This paper presents a description and details of the package, as well as some reproducible examples of usage.

Journal of Statistical Software

January 2022, Volume 101, Issue 8. *doi: 10.18637/jss.v101.i08*

Inference Tools for Markov Random Fields on Lattices: The R Package mrf2d

Victor Freguglia
University of Campinas

Nancy Lopes Garcia
University of Campinas

Abstract

Markov random fields on two-dimensional lattices are behind many image analysis methodologies. **mrf2d** provides tools for statistical inference on a class of discrete stationary Markov random field models with pairwise interaction, which includes many of the popular models such as the Potts model and texture image models. The package introduces representations of dependence structures and parameters, visualization functions and efficient (C++-based) implementations of sampling algorithms, common estimation methods and other key features of the model, providing a useful framework to implement algorithms and working with the model in general. This paper presents a description and details of the package, as well as some reproducible examples of usage.

Keywords: Markov random fields, image analysis, R, Gibbs random fields, Potts model, texture.

1. Introduction

A Markov random field (MRF) is a generalization of the well-known concept of a Markov chain where variables are indexed by vertices of a graph instead of a sequence and the notion of memory is substituted by the neighborhood (edges) of that graph. Markov random fields on lattices, or more generally, Gibbs distributions, have been studied in statistical mechanics as models for interacting particle systems. They range from the basic Ising model (or its generalization Potts model) with pairwise nearest-neighbor interaction to models with more complex interaction types, presenting long-range and/or higher-order interaction. For an introduction to the subject we refer to Liggett (2012) and references therein.

A finite 2-dimensional lattice is a direct representation of pixel positions on a digital image. Geman and Geman (1984) make an analogy between image models and statistical mechanics systems, introducing probability-based computational methods for image restoration under a specific type of noise. Higher-order dependence structures are also described, for example, interactions with pixels more distant than nearest-neighbors. Cross and Jain (1983) use MRFs with special interaction structures to model texture images.

Many modern image analysis methodologies in statistics and machine learning are grounded on Markov random field theory and the local dependence characteristic of image data. Common tasks in image analysis involve image segmentation (Zhang, Brady, and Smith 2001; Kato and Pong 2006; Roche, Ribes, Bach-Cuadra, and Krüger 2011; Cao, Zhou, Xu, Meng, Xu, and Paisley 2018; Ghamisi *et al.* 2018), texture synthesis (Gimel'farb 1996; Freeman and Liu 2011; Versteegen, Gimel'farb, and Riddle 2016) and statistical modeling (Derin and Elliott 1987; Guillot, Rajaratnam, and Emile-Geay 2015; Freguglia, Garcia, and Bicas 2020) all of which can be achieved with the use of MRFs. Some basic references are Blake, Kohli, and Rother (2011) and Kato and Zerubia (2012).

In this paper, when we refer to a MRF, we consider the particular case where variables are indexed by points of a 2-dimensional lattice, not a general graph structure. The regular grid naturally creates a spatial structure and notions of distance and direction for the variables, allowing models to be specified based on this spatial structure (see Besag 1974, for examples).

Parametric inference based on maximum likelihood for such models is difficult, even for the simple models, because of the intractable constant that appears in the likelihood. Inference for the simplest non-trivial case of the Ising model was first studied by Pickard (1987) and continues to present challenges, see for example Bhattacharya and Mukherjee (2018). On the other hand, while there is a continuous development of methodologies used in MRFs in the theoretical field, implementing new algorithms is a challenge in practice, mostly due to the high-dimensionality of the problem and the complexity of the data structures required to represent the data in this type of problem. An overview of this topic, mainly from the Bayesian perspective, can be found in Winkler (2012).

Most methodologies developed are based on Monte Carlo Markov chain methods, thus simple tasks like evaluating pairs of pixels or sampling individual pixels need to be repeated millions or billions of times in iterative methods, depending on the image size, making an efficient implementation of such methods one of the main demands for researchers of the topic.

R (R Core Team 2021) is one of the most used programming languages among statistics researchers, what makes the existence of good packages important for any field of statistics. In a general context (considering the definition of MRFs with general undirected graphs), packages like **graph** (Gentleman, Whalen, Huber, and Falcon 2021) and **network** (Butts 2008) provides tools for representation and manipulation of graph structures which can be used for constructing and visualizing graph-based models. Different versions of graph-based MRFs appear in many packages. For example, the **CRF** package (Wu 2019) has inferential tools Markov random fields with pairwise and unary interactions and their hidden MRF version, **MRFcov** (Clark, Wells, and Lindberg 2018) allows inference for the interaction parameter of between nodes of a graph considering covariates and **gamlss.spatial** (De Bastiani, Rigby, Stasinopoulous, Cysneiros, and Uribe-Opazo 2018) allows fitting Gaussian Markov random fields in a spatial context, similar to **INLA** (Rue, Martino, and Chopin 2009) and **mgcv** (Wood 2017).

Outside of the R ecosystem, there are powerful software in C++ used for image analysis related to Markov random fields, such as the **DGM** library (Kosov 2013) and **densecrf** (Krähenbühl and Koltun 2011), which also has a Python wrapper (Beyer 2015), and can be used for a variety of tasks and use extremely efficient computational methods.

For discrete MRFs on lattice data, closer to what is proposed in **mrf2d**, there are some R packages available. The **potts** package (Geyer and Johnson 2020), implements simulation algorithms and parameter estimation via composite-likelihood for a Potts model with nearest-neighbor interactions only. **PottsUtils** (Feng 2018) also implements simulation and tools for computing normalization constants in one, two and three-dimensional Potts model. The package **bayesImageS** (Moores, Nicholls, Pettitt, and Mengersen 2020) provides Bayesian image segmentation algorithms considering Gaussian mixtures driven by hidden Potts models with slightly more complex interaction neighborhood. **GiRaF** (Stoehr, Pudlo, and Friel 2020) allows calculation on, and sampling from general homogeneous Potts model. The **Pottslab** (Storath and Weinmann 2014) package for MATLAB also provides image segmentation algorithms using the Potts model, including for multivariate-valued data.

Although the available packages for discrete-valued MRFs offer efficient implementations of their methods, they do not provide an interface that allows simple extensions to different cases, for example, different interaction types for different positions and sparse long-range interaction neighborhoods. Some of the algorithms used also rely on specific characteristics of the specific setups they consider and cannot be applied more generally.

The **mrf2d** package (Freguglia 2022) provides a complete framework for statistical inference on discrete-valued MRF models on 2-dimensional lattice data, where all the elements used by algorithms (such as conditional probabilities, pseudo-likelihood function, simulation, sufficient statistics and more) are available for the user, as well as many built-in model fitting functions.

The package uses the model described in Freguglia *et al.* (2020) as a reference. Many other models, such as the Potts model and auto-models, are particular cases of our model obtained by including restrictions to the parameters or using specific interacting neighborhoods. These neighborhoods can be freely specified within the package and 5 families of parameter restrictions are available to cover the particular cases.

mrf2d (Freguglia 2022) is available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=mrf2d` and can be installed and loaded with

```
R> install.packages("mrf2d")
R> library("mrf2d")
```

The development version is in its GitHub repository and can be installed with

```
R> devtools::install_github("Freguglia/mrf2d")
```

This paper is organized as follows. Section 2 describes the model considered in **mrf2d**, Section 3 presents the main functionalities of the package and details of the implementation, which are illustrated by examples in Section 4. We finish with a discussion in Section 5. All the results of example code in this article were obtained using R version 4.0.2 and **mrf2d** version 1.0.

2. Model description

Let $\mathcal{L} \subset \{\mathbf{i} = (i_1, i_2) \in \mathbb{N}^2\}$ be a finite set of locations in a two-dimensional lattice region and $\mathbf{Z} = \{Z_\mathbf{i}\}_{\mathbf{i} \in \mathcal{L}}$ a field of random variables indexed by those locations.

The main purpose of **mrf2d** is to provide a general framework for Markov random field models which satisfy the following assumptions:

(a) Finite support Each $Z_{\mathbf{i}}$ can take values in $\mathcal{Z} = \{0, \dots, C\}$ for some finite $C > 0$.

(b) Pairwise interactions The probability of a complete configuration $\mathbb{P}(\mathbf{Z} = \mathbf{z})$ can be decomposed into a product of functions of the pairs $(z_{\mathbf{i}}, z_{\mathbf{j}}), \mathbf{i} \neq \mathbf{j} \in \mathcal{L}$.

(c) Homogeneous interactions For any two pixels $\mathbf{i}$ and $\mathbf{j}$ and any relative position $\mathbf{r}$, the interaction between pixels $\mathbf{i}$ and $\mathbf{i} + \mathbf{r}$ is the same as for pixels $\mathbf{j}$ and $\mathbf{j} + \mathbf{r}$, i.e., the interaction depends on the relative position of the pair of pixels, not on their position in the lattice.

These assumptions are satisfied by most commonly used models in image processing.

We use the representation in Freguglia *et al.* (2020) which expresses the probability distribution of the random field in the form of the exponential family and introduce additional constraints to parameter space and/or different dependence structures to include particular features of the model under study.

2.1. Homogeneous Markov random field with pairwise interactions

MRF models are characterized by their conditional independence property. Let $\mathcal{N}$ be a neighborhood system on $\mathcal{L}$, then $\mathbf{Z}$ is a *Markov random field* with respect to $\mathcal{N}$ if $Z_{\mathbf{i}}$ given its neighbors $\mathbf{Z}_{\mathcal{N}_{\mathbf{i}}}$ is conditionally independent from all other variables

$$\mathbb{P}(Z_{\mathbf{i}} = z_{\mathbf{i}} \mid \mathbf{Z}_{-\mathbf{i}}) = \mathbb{P}(Z_{\mathbf{i}} = z_{\mathbf{i}} \mid \mathbf{Z}_{\mathcal{N}_{\mathbf{i}}}), \qquad \mathbf{i} \in \mathcal{L}, \tag{1}$$

where $\mathbf{Z}_{-\mathbf{i}}$ denotes the set of variables $\{Z_{\mathbf{j}}, \mathbf{j} \neq \mathbf{i}\}$.

To start defining MRFs in an image processing context, a location of the lattice $\mathbf{i} \in \mathcal{L}$ will be referred as a *pixel* $\mathbf{i}$ and an observed value of the variable $z_{\mathbf{i}} \in \mathcal{Z}$ as *pixel value* or *color*.

We denote by $\mathcal{R} \subset \mathbb{Z}^2$ a set of interacting relative positions such that, for no pair of elements $\mathbf{r}, \mathbf{r}' \in \mathcal{R}$ we have $\mathbf{r}' = -\mathbf{r}$ (no position in $\mathcal{R}$ is a reflection of another). Based on $\mathcal{R}$, we can construct a neighborhood system (interaction structure) $\mathcal{N}$ in such way that the set of neighbors of site $\mathbf{i}$, $\mathcal{N}_i$ can be represented by a graph with vertices $\mathcal{L}$ where there is an edge connecting $\mathbf{i}$ and $\mathbf{j}$ if, and only if, $\mathbf{j} = \mathbf{i} \pm \mathbf{r}$. For example, a nearest-neighbor structure corresponds to $\mathcal{R} = \{(1,0), (0,1)\}$.

Given an interaction structure $\mathcal{R}$, for any relative position $\mathbf{r} \in \mathcal{R}$ the interactions associated to that relative position are characterized by a map $\theta_{\mathbf{r}}(\cdot,\cdot)$, $\theta_{\mathbf{r}} : \mathcal{Z}^2 \to \mathbb{R}$. For $a, b \in \mathcal{Z}$, the value $\theta_{\mathbf{r}}(a,b)$ is called a potential.

The model in **mrf2d** considers a neighborhood system $\mathcal{N}$ that connects pairs of pixel positions $\mathbf{i}, \mathbf{j}$ such that $\mathbf{i} - \mathbf{j} \in \mathcal{R}$. Under assumptions (a), (b) and (c), the Hammersley-Clifford theorem (Hammersley and Clifford 1971) implies that the probability function for $\mathbf{Z}$ belongs to the exponential family and can be described by a set of natural parameters $\boldsymbol{\theta} = \{\theta_{\mathbf{r}}(a,b), \mathbf{r} \in \mathcal{R}, a, b, \in \mathcal{Z}\}$,

$$\mathbb{P}(\mathbf{Z} = \mathbf{z}) = \frac{1}{\zeta_\theta} e^{H(\mathbf{z}, \boldsymbol{\theta})}, \tag{2}$$

where

$$H(\mathbf{z}, \boldsymbol{\theta}) = \sum_{\mathbf{r} \in \mathcal{R}} \sum_{\mathbf{i}, \mathbf{j} \in \mathcal{L}} \theta_r(z_{\mathbf{i}}, z_{\mathbf{j}}) \mathbb{1}_{(\mathbf{j} = \mathbf{i} + \mathbf{r})} \text{ and } \zeta_\theta = \sum_{\mathbf{z}'} e^{H(\mathbf{z}', \boldsymbol{\theta})}. \tag{3}$$

Figure 1 illustrates how the function $H(\mathbf{z}, \boldsymbol{\theta})$ is computed for an example interaction structure $\mathcal{R}$ and a field $\mathbf{z}$.

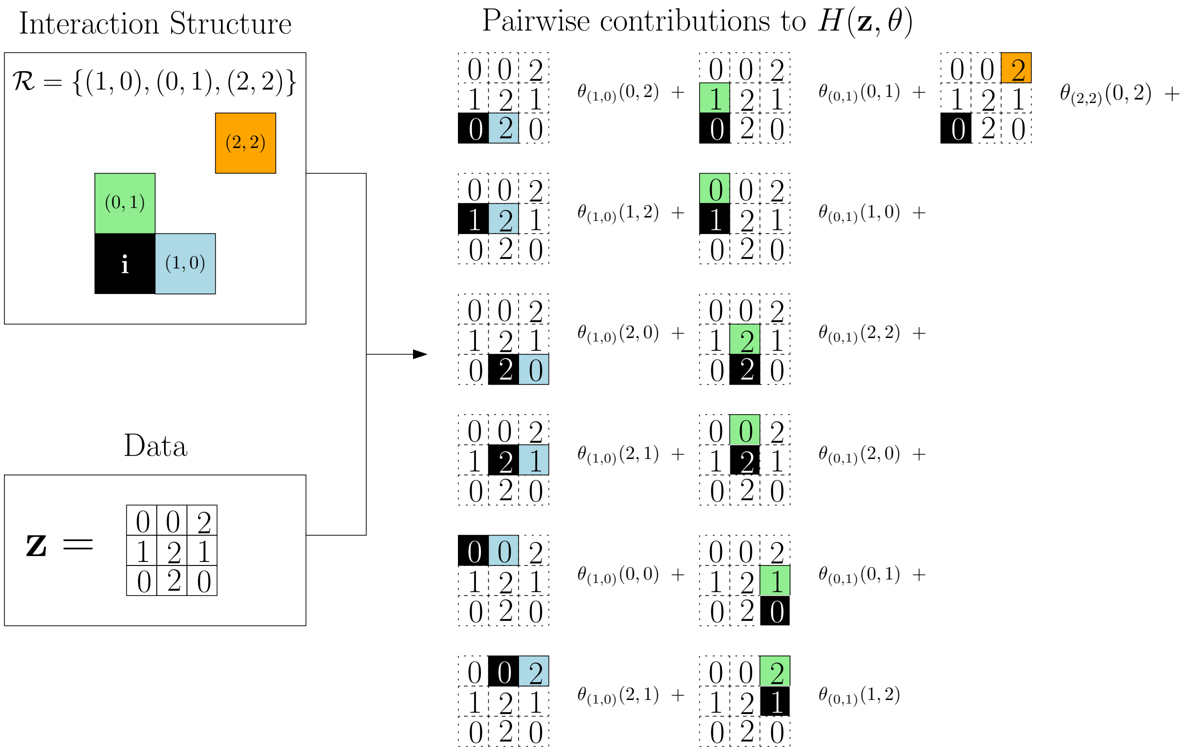


Figure 1: Example of interaction structure with three relative positions and example field on a 3 by 3 lattice (left) and contributions of each interacting pair to $H(\mathbf{z},\boldsymbol{\theta})$(right).

Note that adding a constant a constant $c_{\mathbf{r}}$ to the potentials associated with a relative position $\mathbf{r} \in \mathcal{R}$ results in the same probability because the constant cancels when dividing by $\zeta_{\boldsymbol{\theta}}$. Thus, constraints for the potentials $\theta_{\mathbf{r}}(a,b)$ are necessary to obtain identifiability in the model. We consider $\theta_{\mathbf{r}}(0,0) = 0$ for all relative positions $\mathbf{r}$, which ensures identifiability and also gives an interpretation for interactions in terms of the pair $(0,0)$: $\theta_{\mathbf{r}}(a,b) < 0$ (resp. > 0) means that the pair (a,b) is *less* (resp. *more*) likely to appear in a pair with relative position $\mathbf{r}$ than $(0,0)$.

Potts model as a particular case

The Potts model (Potts 1952) is one of the most important MRF model used in image segmentation because it can assign higher probability for equal-valued pairs of nearest-neighbors, creating large regions of pixels with the same values. The model has a single parameter ϕ that is interpreted as the inverse temperature in a mechanical statistics context.

A standard Potts model can be expressed as Equation 2 with the function $H(\mathbf{z},\boldsymbol{\theta})$ taking the form

$$\sum_{(\mathbf{i},\mathbf{j}):||\mathbf{i}-\mathbf{j}||=1} \phi \mathbb{1}_{(z_{\mathbf{i}} \neq z_{\mathbf{j}})}. \tag{4}$$

Assumptions (a), (b) and (c) are satisfied, thus, we can rewrite Equation 4 in terms of an interaction structure $\mathcal{R}$ and potentials $\boldsymbol{\theta}$ by noticing

- The set $\mathbf{i},\mathbf{j} : ||\mathbf{i}-\mathbf{j}|| = 1$ are vertical and horizontal pairs of neighbors, therefore, the interaction structure $\mathcal{R}$ is the set $\{(1,0),(0,1)\}$.

- The potential $\theta_{\mathbf{r}}(a, b)$ is equal to ϕ if $a \neq b$ and 0 otherwise, regardless of $\mathbf{r}$. The constraint $\theta_{\mathbf{r}}(0, 0) = 0$ is satisfied in this definition. Therefore, we have the parameter restriction
$$\theta_{\mathbf{r}}(a, b) = \phi \mathbb{1}_{(a \neq b)}$$
for all $\mathbf{r} \in \mathcal{R}$.

This parameter restriction corresponds to the `"onepar"` family described in Section 3.2.

2.2. Important elements of the model

The main inference challenge for MRFs lies in the normalizing constant $\zeta_{\boldsymbol{\theta}}$ appearing in Equation 2. It cannot be evaluated in practice as it requires summing over $\mathcal{Z}^{|\mathcal{L}|}$ possible field configurations and there is no analytical expression for it, except for trivial cases, leading to an intractable likelihood.

Being unable to evaluate the likelihood function hinders the use of most statistical methods. Inference under intractable likelihoods have been developed over the years. The main studies involve using conditional probability-based functions, like pseudo-likelihood (Jensen and Künsch 1994, for example) and Monte Carlo methods (Geyer and Thompson 1992; Møller, Pettitt, Reeves, and Berthelsen 2006, for example).

Although there is a wide variety of inferential methods available, most of them are built using the same pieces of the model. Thus, having access to each of these pieces is necessary to implement algorithms. We highlight important characteristics of the model available in **mrf2d** that are used by inference methods.

Conditional probabilities

A consequence of the Markov property (conditional independence) is a simple expression for conditional probabilities. $H(\mathbf{z}, \boldsymbol{\theta})$ is a sum of terms that only depends on pairs of pixel values, which implies that all terms not involving position $\mathbf{i}$ cancel out when evaluating $\mathbb{P}(Z_{\mathbf{i}} \mid \mathbf{Z}_{-\mathbf{i}})$. Define the part of the sum that involves the pixel in position $\mathbf{i}$ as

$$h_{\mathbf{i}}(k \mid \mathbf{z}) = \sum_{\mathbf{r} \in \mathcal{R}}^{(\mathbf{i}+\mathbf{r}) \in \mathcal{L}} \theta_{\mathbf{r}}(k, z_{(\mathbf{i}+\mathbf{r})}) + \sum_{\mathbf{r} \in \mathcal{R}}^{(\mathbf{i}-\mathbf{r}) \in \mathcal{L}} \theta_{\mathbf{r}}(z_{(\mathbf{i}-\mathbf{r})}, k). \tag{5}$$

The conditional probability of $Z_{\mathbf{i}} = k$ given all other locations, $\mathbb{P}(Z_{\mathbf{i}} = k \mid \mathbf{Z}_{\mathcal{N}_{\mathbf{i}}})$, is then given by the standard softmax of $h_{\mathbf{i}}(k \mid \mathbf{z})$,

$$\mathbb{P}(Z_{\mathbf{i}} = k \mid \mathbf{Z}_{-\mathbf{i}} = \mathbf{z}_{\mathcal{N}_{\mathbf{i}}}) = \frac{e^{h_{\mathbf{i}}(k|\mathbf{z})}}{\sum_{k'} e^{h_{\mathbf{i}}(k'|\mathbf{z})}}. \tag{6}$$

Pseudo-likelihood function

The pseudo-likelihood function (Besag 1974, 1975) is defined as the product of conditional probabilities of each variable given all other variables of a random field,

$$PL(\boldsymbol{\theta}; \mathbf{z}) = \prod_{i \in \mathcal{L}} \mathbb{P}(Z_{\mathbf{i}} = z_{\mathbf{i}} \mid \mathbf{Z}_{-\mathbf{i}} = \mathbf{z}_{-\mathbf{i}}) = \prod_{\mathbf{i} \in \mathcal{L}} \frac{e^{h_{\mathbf{i}}(z_{\mathbf{i}}|\mathbf{z})}}{\sum_{k'} e^{h_{\mathbf{i}}(k'|\mathbf{z})}}. \tag{7}$$

Algorithm 1: Approximate sampling algorithm for MRFs using T steps of Gibbs sampler.

```
Initialize z with a starting configuration $\mathbf{z} = \mathbf{z}^{(0)}$;
Initialize the iteration counter $t = 0$; while $t \leq T$ do
    Sample $\{\mathbf{i}^{(1)}, \mathbf{i}^{(2)}, \dots, \mathbf{i}^{(|\mathcal{L}|)}\}$ a random permutation of the pixel positions $\mathcal{L}$;
    for $\ell$ in $1, \dots, |\mathcal{L}|$ do
        Update $z_{i^{(\ell)}}$ conditional to the rest of the field $\mathbf{z}_{-\mathbf{i}^{(\ell)}}$ with probabilities from
         Equation 6;
    end
    t = t + 1;
    Result: output the final configuration z.
end
```

In the special case of an independent field, it is equivalent to the likelihood function. Notice that the pseudo-likelihood function does not depend on the intractable normalizing constant and Equation 7 is numerically equivalent to a logistic regression problem where each pixel values corresponds to independent observations and the interacting pixel values are covariates with coefficients corresponding to the associated potentials.

Generating MRFs via Gibbs sampler

While exact sampling from dependent and high-dimensional processes is a challenging task overall, the conditional independence of MRFs simplifies the implementation of the Gibbs sampler algorithm (Geman and Geman 1984). In the Gibbs sampler algorithm, each pixel value is updated conditionally to the current state of its neighbors and a Gibbs sampler *cycle* consists of updating each pixel exactly one time.

To avoid introducing any kind of bias due to updates order, a random permutation of $\mathcal{L}$ is drawn to define the order in which pixels are updated at each cycle. After running a suitable number of cycles in Algorithm 1, the distribution of the resulting field sampled in the process is approximately the joint distribution of the MRF.

Sampling a field conditional to a subset of pixel values can be achieved with the same algorithm by skipping the updates for those pixels which are being conditioned on.

There exist faster mixing algorithms for particular cases such as Swendsen-Wang algorithm (Wang and Swendsen 1990), but they require specific conditions from the model and/or particular implementations to be efficient. Therefore, despite its slower mixing times in some scenarios, we keep the Gibbs sampler as the method of choice in this work due to its generalization ability as it only requires computing conditional distributions.

Sufficient statistics

An important computational consequence of the model assumptions is the fact that, in order to evaluate the probability (or likelihood) function for a particular observed field $\mathbf{z}$, it is not necessary to determine the values of each pixel individually, but only the co-occurrence counts for each relative position $\mathbf{r} \in \mathcal{R}$.

The function $H(\mathbf{z}, \theta)$ can be rewritten as

$$H(\mathbf{z}, \theta) = \sum_{\mathbf{r} \in \mathcal{R}} \sum_{a=0}^{C} \sum_{b=0}^{C} \theta_{\mathbf{r}}(a, b) n_{a,b,\mathbf{r}}(\mathbf{z}), \tag{8}$$

where $n_{a,b,r}(\mathbf{z}) = \sum_{\mathbf{i} \in \mathcal{L}} \mathbb{1}_{(z_{\mathbf{i}}=a, z_{(\mathbf{i}+\mathbf{r})}=b)}$ is the count of occurrences of the pair $(a, b) \in \mathcal{Z}^2$ in pairs of pixels with relative position $\mathbf{r}$. Therefore,

$$S_{\mathcal{R}}(\mathbf{z}) = \{n_{a,b,\mathbf{r}}(\mathbf{z}), a, b \in \mathcal{Z}, \mathbf{r} \in \mathcal{R}\}$$

is a vector of sufficient statistics, where each component $n_{a,b,\mathbf{r}}(\mathbf{z})$ is associated with a corresponding potential $\theta_{\mathbf{r}}(a, b)$. Gimel'farb (1996) calls this sufficient statistic the *co-occurrence histogram.*

Parameter constraints reduce the dimension of the sufficient statistic. Our identifiability constraint $\theta_{\mathbf{r}}(0, 0) = 0$ implies that all $n_{0,0,\mathbf{r}}(\mathbf{z})$ are excluded from $S_{\mathcal{R}}(\mathbf{z})$ and equality constraints require aggregating (sum) co-occurrence counts to match the parameter dimension. We shall keep the same notation for the constrained version of the sufficient statistics $S_{\mathcal{N}}(\mathbf{z})$ and potentials $\boldsymbol{\theta}$.

The main advantages of the representation with sufficient statistics are the reduced memory usage in Monte Carlo methods and a convenient representation of $H(\mathbf{z}, \boldsymbol{\theta})$ as an inner product that simplifies dealing with likelihood ratios as it is done in Geyer and Thompson (1992),

$$H(\mathbf{z}, \boldsymbol{\theta}) = \langle S_{\mathcal{N}}(\mathbf{z}), \boldsymbol{\theta} \rangle. \tag{9}$$

2.3. Gaussian mixtures driven by hidden MRFs

Another class of models present in the image processing field are hidden Markov random field models (HMRFs). The hidden version considers a latent (unobserved) process, denoted $\mathbf{Z}$ and an observed field, denoted $\mathbf{Y}$, where $\mathbf{Z}$ is distributed as a MRF and the distribution of $\mathbf{Y} \mid \mathbf{Z}$ is reasonably simple.

In this type of modeling, $\mathbf{Z}$ is often considered the "true" image and $\mathbf{Y}$ is a noisy image. Usually the goal of the analysis in this context is to recover the underlying field. Note that for the models considered in this work, where $\mathbf{Z}$ has finite support, the hidden field defines a segmentation of the image, making it a suitable approach for image segmentation.

In **mrf2d**, we provide built-in tools for the case where $\mathbf{Y} \mid \mathbf{Z}$ is a finite Gaussian mixture with mixture components driven by the hidden field. Additional covariates can also be included as fixed effects for the mean,

$$Y_{\mathbf{i}} \mid Z_{\mathbf{i}} = a \sim N(\mu_a + \mathbf{x_i}\top\boldsymbol{\beta}, \sigma_a^2), \qquad a = 0, 1, \dots, C. \tag{10}$$

Given the latent field, observed values $\mathbf{Y}$ are assumed to be independent leading to the conditional density

$$f(\mathbf{y} \mid \mathbf{Z} = \mathbf{z}) = \prod_{\mathbf{i} \in \mathcal{L}} \frac{1}{\sqrt{2\pi\sigma_{z_{\mathbf{i}}}^2}} \exp\left(\frac{(y_{\mathbf{i}} - \mu_{z_{\mathbf{i}}} - \mathbf{x}_{\mathbf{i}}^\top \boldsymbol{\beta})}{2\sigma_{z_{\mathbf{i}}}^2} \right) \tag{11}$$

and the complete likelihood function

$$L_{\boldsymbol{\theta}}\left(\boldsymbol{\beta}, \{(\mu_a, \sigma_a), a = 0, \ldots, C\}; \mathbf{y}, \mathbf{z}\right) = \frac{1}{\zeta_\theta} e^{(H(\mathbf{z}, \theta))} \prod_{\mathbf{i} \in \mathcal{L}} \frac{1}{\sqrt{2\pi\sigma^2_{z_\mathbf{i}}}} \exp\left(\frac{(y_\mathbf{i} - \mu_{z_\mathbf{i}} - \mathbf{x}_\mathbf{i}^\top \boldsymbol{\beta})}{2\sigma^2_{z_\mathbf{i}}}\right) \quad (12)$$

Inference for this models involves estimating the parameters $(\mu_k, \sigma_k)_{k=0,1,\ldots,C}$ and $\boldsymbol{\beta}$ associated with the Gaussian Mixture and predicting the labels of the latent field $\mathbf{z}$ simultaneously. Bayesian methods and the EM algorithm are the most common approaches. The parameters of the latent field distribution $\boldsymbol{\theta}$ are fixed a priori and considered tuning hyper-parameters of the algorithm.

3. Using the package

3.1. Model representation

The model described in Section 2 can be completely characterized by three components: the random field $\mathbf{z}$, the interaction structure $\mathcal{R}$ and the potentials $\boldsymbol{\theta}$. Additionally, $\mathbf{y}$ and the mixture parameters $(\mu_k, \sigma_k^2)_{k=0,\ldots,C}$ are included in hidden MRFs.

A consistent representation of each component is provided in the package so that inputs and outputs of built-in functions, as well as methods the user may implement, are compatible and usable in the analysis pipeline. Representations are described in Table 1.

Random fields $\mathbf{z}$ *and* $\mathbf{y}$.

Realizations of a random fields $\mathbf{z}$ and $\mathbf{y}$ are represented by simple `matrix` objects with dimension $N \times M$, where $N \geq \max_{i_1}(i_1, i_2) \in \mathcal{L}$ and $M \geq \max_{i_2}(i_1, i_2) \in \mathcal{L}$, i.e., the maximal

Model component	Function argument	Representation in **mrf2d**
$\mathbf{z}$: Discrete-valued field	`Z`	A `matrix` object with values in $\{0, \ldots, C\}$, where `Z[w,q]` represents the pixel value in position (w, q) of the lattice. `NA` values are used for positions that do not belong to $\mathcal{L}$ when it is not a rectangular region.
$\mathbf{y}$: Continuous-valued field	`Y`	A `matrix` object with real values, where `Y[u,v]` represents the pixel value in position (u, v) of the lattice. `NA` values are used for positions that do not belong to $\mathcal{L}$ when it is not a rectangular region.
$\mathcal{R}$: Interaction structure	`mrfi`	An object of the S4 class '`mrfi`'. It can be created with the `mrfi()` and `rpositions()` functions.
$\theta_\mathbf{r}(a, b)$: Array of potentials	`theta`	A three-dimensional `array` object with dimensions $(C+1) \times (C+1) \times \lvert\mathcal{R}\rvert$. For a pair of values (a, b) and the s-th interacting relative position $\mathbf{r}_s$ of $\mathcal{R}$, the corresponding potential is mapped at `theta[a+1, b+1, s]`.

Table 1: Model representation summary.

coordinates. This matrix represents a rectangular set of pixels that contains $\mathcal{L}$. The value in row i_1 and column i_2 represents the observed value of the random field in position (i_1, i_2): an integer in $\{0, 1, \ldots, C\}$ for $\mathbf{z}$ or a real number for $\mathbf{y}$.

We do not require $\mathcal{L}$ to be a complete rectangular region. Pixels which position does not belong to $\mathcal{L}$ are assigned the `NA` value.

Two functions are available for visualizing random fields: `dplot()` and `cplot()`. `dplot()` should be used for **d**iscrete-valued fields $\mathbf{z}$ while `cplot()` is used for **c**ontinuous-valued matrices $\mathbf{y}$. These functions provide an alternative to base R `image()` function, producing elegant images in the form of `ggplot` objects. The main advantage is that they allow the use of the **ggplot2** package (Wickham 2016) to customize the image using the grammar of graphics. Details and examples of customization of the images produced using **ggplot2** can be found in Appendix A.

Interaction structures $\mathcal{R}$

Interaction structures are represented by objects of the S4 class '`mrfi`' implemented in **mrf2d**. These objects can be created with the `mrfi()` or `rpositions()` functions, which have arguments `max_norm`, `norm_type` and `positions`. In `mrfi()`, the interaction structure created will include all relative positions which satisfy $||(i_1, i_2)|| \leq$ `max_norm` for the specified norm type. `positions` can be passed as a list containing length 2 integer vectors with relative positions to include when using `rpositions()`. The function automatically checks for repeated and opposite relative positions to ensure the structure is valid.

`norm_type` options are the same as R built-in `norm()` function, mainly, `"1"`, `"2"` and `"m"` are used for ℓ_1, ℓ_2 and the maximum norm, respectively. The default is ℓ_1 norm.

An algebra of `mrfi` objects is implemented for manipulating these objects. `+` is used to perform union of two `mrfi` objects or a `mrfi` object and a `numeric` vector with 2 integers can be used to add a single interacting position to an existing `mrfi` object. Similarly, the `-` operator can be used to perform set difference between two `mrfi` objects or to remove a single position if a vector with 2 integers is used in the right-hand-side.

Some examples for creating different $\mathcal{R}$ are detailed below.

- `mrfi(max_norm = 1)` creates an interaction structure with all positions with $||(i_1, i_2)||_1 \leq 1$, which corresponds to a nearest-neighbor structure $\mathcal{R} = \{(1, 0), (0, 1)\}$.
- `rpositions(positions = list(c(1,0), c(0,1)))` is an alternative way of specifying the same structure of the previous example.
- `mrfi(max_norm = 1) + rpositions(positions = list(c(2,0)))` results in the interaction structure $\mathcal{R} = \{(1, 0), (0, 1), (2, 0)\}$.
- `mrfi(max_norm = 1) + rpositions(positions = list(c(-1,0)))` results in $\mathcal{R} = \{(0, 1), (-1, 0)\}$. The norm-based and position-based positions had an intersection (reflected position at $(-1, 0)$), so the redundant position $(1, 0)$ was removed. In case of opposite directions being added together, right-hand size argument is prioritized.

Additionally, conversion of `mrfi` objects to `list` is implemented in the `as.list()` method. Subsetting methods are also available with the `"[]"` and `"[[]]"` operators. These methods

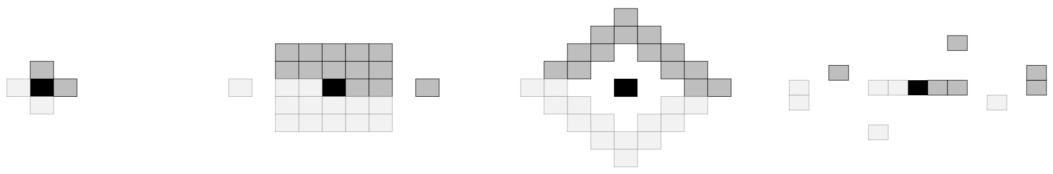


Figure 2: Examples of interaction structures $\mathcal{R}$ created and their visualization.

are particularly important for model selection algorithms, as many distinct sparse interaction structures can be obtained by using different subsets of a large reference base structure.

A `plot` method is available for `mrfi` objects. The code chunk below exemplifies the usage of plotting functions and manipulation of `mrfi` objects. The resulting plots are presented in Figure 2. The black square represents the origin position $(0, 0)$, positions included in the interaction structure $\mathcal{R}$ are represented by the dark-gray squares with black borders, while their opposite directions are the light-gray squares.

```
R> plot(mrfi(max_norm = 1))
R> plot(mrfi(max_norm = 2, norm_type = "m") + c(4, 0))
R> plot(mrfi(4) - mrfi(2))
R> plot(mrfi(6, norm_type = "m")[c(1, 2, 6, 9, 19, 41)])
```

Potentials array $\boldsymbol{\theta}$

The collection of potentials, $\theta_{\mathbf{r}}(a, b)$, is represented by an `array` object with dimensions $(C + 1) \times (C + 1) \times |\mathcal{R}|$. Rows and columns are used to map a and b, respectively, while slices are used to map relative positions $\mathbf{r}$. A set of potentials $\{\theta_{\mathbf{r}}(a, b), a, b \in \mathcal{Z}, \mathbf{r} \in \mathcal{R}\}$ is always related to an interaction structure $\mathcal{R} = \{\mathbf{r}_1, \mathbf{r}_2, \ldots, \mathbf{r}_{|\mathcal{R}|}\}$. Since the i-th slice maps the i-th relative position of $\mathcal{R}$, $\mathbf{r}_i$, this is the minimal representation required to store all parameters required by the model.

An important detail is that array indices in R start at `1`, while we consider our set of possible values $\mathcal{Z} = \{0, 1, \ldots, C\}$, therefore we need to shift a and b by one position when accessing their value in the R array. Figure 3 illustrates how potentials can be represented as an array in R in the $C = 2$ case. Two elements are highlighted and the associated indices used to access them are shown as examples.

3.2. Parameter restriction families

Parameter restrictions play an important role in the inference process of our Markov random field models. **mrf2d** functions support 5 families of parameter restrictions for the array of potentials to be considered in inference algorithms. They are specified by the `family` argument of functions to ensure the resulting output array (`theta`) respects those constraints. A brief description of each interaction structure is given next. Table 2 presents the mathematical definitions, number of free parameters and an example of a slice of the array of potentials for the case with $C = 2$ in each family.

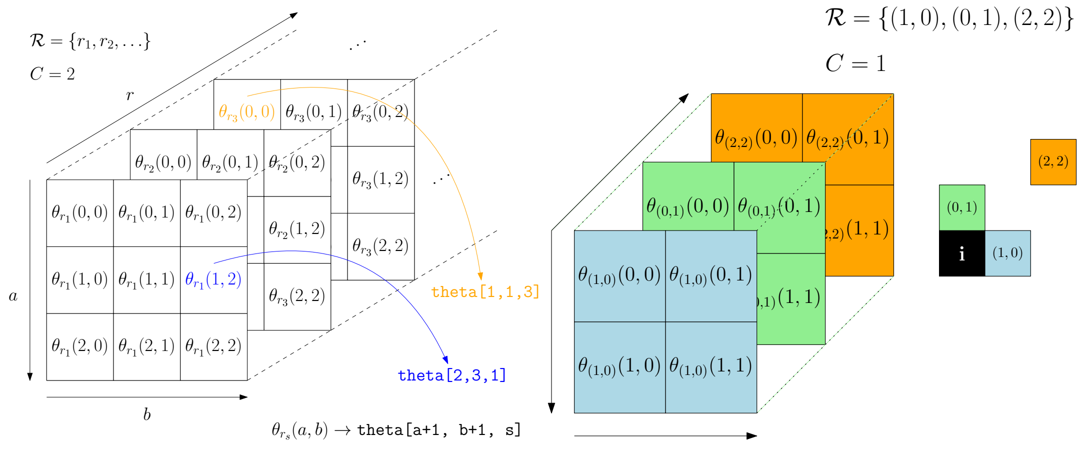


Figure 3: Left: Example of array representation of potentials with $C = 2$ for a generic interaction structure $\mathcal{R}$. Right: Array representation of potentials for the example from Figure 1.

Family	Restriction	Free parameters	Example slice
`"onepar"`	$\theta_{\mathbf{r}}(a,b) = \phi \mathbb{1}_{(a \neq b)}$	1	$\begin{bmatrix} 0 & \phi & \phi \\ \phi & 0 & \phi \\ \phi & \phi & 0 \end{bmatrix}$
`"oneeach"`	$\theta_{\mathbf{r}}(a,b) = \phi_{\mathbf{r}} \mathbb{1}_{(a \neq b)}$	$\lvert\mathcal{R}\rvert$	$\begin{bmatrix} 0 & \phi_{\mathbf{r}} & \phi_{\mathbf{r}} \\ \phi_{\mathbf{r}} & 0 & \phi_{\mathbf{r}} \\ \phi_{\mathbf{r}} & \phi_{\mathbf{r}} & 0 \end{bmatrix}$
`"absdif"`	$\theta_{\mathbf{r}}(a,b) = \sum_{d=1}^{C} \phi_{\mathbf{r},d} \mathbb{1}_{(\lvert b-a\rvert = d)}$	$\lvert\mathcal{R}\rvert C$	$\begin{bmatrix} 0 & \phi_{\mathbf{r},1} & \phi_{\mathbf{r},2} \\ \phi_{\mathbf{r},1} & 0 & \phi_{\mathbf{r},1} \\ \phi_{\mathbf{r},2} & \phi_{\mathbf{r},1} & 0 \end{bmatrix}$
`"dif"`	$\theta_{\mathbf{r}}(a,b) = \sum_{d=-C, d\neq 0}^{C} \phi_{\mathbf{r},d} \mathbb{1}_{(b-a=d)}$	$\lvert\mathcal{R}\rvert 2C$	$\begin{bmatrix} 0 & \phi_{\mathbf{r},1} & \phi_{\mathbf{r},2} \\ \phi_{\mathbf{r},-1} & 0 & \phi_{\mathbf{r},1} \\ \phi_{\mathbf{r},-2} & \phi_{\mathbf{r},-1} & 0 \end{bmatrix}$
`"free"`	$\theta_{\mathbf{r}}(0,0) = 0$	$\lvert\mathcal{R}\rvert(C^2 - 1)$	$\begin{bmatrix} 0 & \phi_{\mathbf{r},0,1} & \phi_{\mathbf{r},0,2} \\ \phi_{\mathbf{r},1,0} & \phi_{\mathbf{r},1,1} & \phi_{\mathbf{r},1,2} \\ \phi_{\mathbf{r},2,0} & \phi_{\mathbf{r},2,1} & \phi_{\mathbf{r},2,2} \end{bmatrix}$

Table 2: Description of parameter restriction families.

`"onepar"` A single-parameter (ϕ) model, where interactions depend only on the fact that values are equal or different, regardless of their relative position. This restriction corresponds to the classical Ising and Potts model.

`"oneeach"` The same interaction type as `"onepar"`, but allowing different values $\phi_{\mathbf{r}}$ for different interacting positions $\mathbf{r} \in \mathcal{R}$.

`"absdif"` For each $\mathbf{r} \in \mathcal{R}$, the potentials $\theta_{\mathbf{r}}(a,b)$ depend only on the absolute differences of their pixel values $d = \lvert b - a\rvert$. Note that `"absdif"` is equivalent to `"oneeach"` when $C = 1$.

`"dif"` Generalizes the `"absdif"` family allowing opposite signal differences to have different interactions.

`"free"` No restrictions, except for the identifiability constraint $\theta_{\mathbf{r}}(0,0) = 0$.

Families `"dif"` and `"absdif"` should only be used when pixel values represent quantities and their differences are well-defined, for instance, in grayscale images with few levels, as in the example analysed in Gimel'farb (1996). If relabeling the values does not change the interpretation of the problem, then these restrictions are probably not suitable.

The function `smr_array(theta, family)` can be used to transform a parameter array into a vector of appropriate length containing only the free parameters corresponding to the provided array (`theta`) and the restriction `family`. The opposite operation is also available as the `expand_array(theta_vec, family, mrfi, C)` function. These transformations use a simpler and less memory consuming structure so they are particularly useful for optimization problems as most functions, for example R built-in `optim` function, require a vector of parameters. Also, they are convenient for storing multiple vectors, for example in Monte Carlo methods.

3.3. Random field sampler

Being able to sample observations of Markov random fields is a key component of many inference methods that aim to avoid the intractable normalizing constant. In **mrf2d**, a complete and efficient routine to sample fields using the Gibbs sampler algorithm described in Algorithm 1 is provided by the `rmrf2d()` function. Its arguments are:

- `init_Z`: The initial field configuration, or a length-2 vector with the dimensions of the field to be sampled. If the dimensions are provided, the initial configuration is randomly sampled from independent discrete uniform distributions.
- `mrfi`: A `mrfi` object representing the interaction structure $\mathcal{R}$.
- `theta`: An `array` of potentials.
- `cycles`: The number of Gibbs sampler cycles.
- `sub_region`: Optional argument used for non-rectangular images when `init_Z` is a vector holding the dimensions. A `logical` matrix with the same dimensions as specified in `init_Z`. Pixels with `FALSE` value are not included in the image.
- `fixed_region`: Optional. A matrix with `logical` values. Pixel positions with `TRUE` value are conditioned on their initial configuration (`init_Z`) value and are not updated.

We illustrate below the use of the sampling function on two fields: a 200×200 field (`z_sample`) without conditioning on any pixel (nothing specified in `fixed_region`) and a 100×100 field (`z_border`) conditioning on the boundary values being fixed as 0, sampled from a random initial configuration. The resulting images are presented in Figure 4.

```
R> th <- expand_array(-1, family = "onepar", mrfi(1), C = 1)
R> z_sample <- rmrf2d(init_Z = c(200,200), mrfi = mrfi(1), theta = th)
```

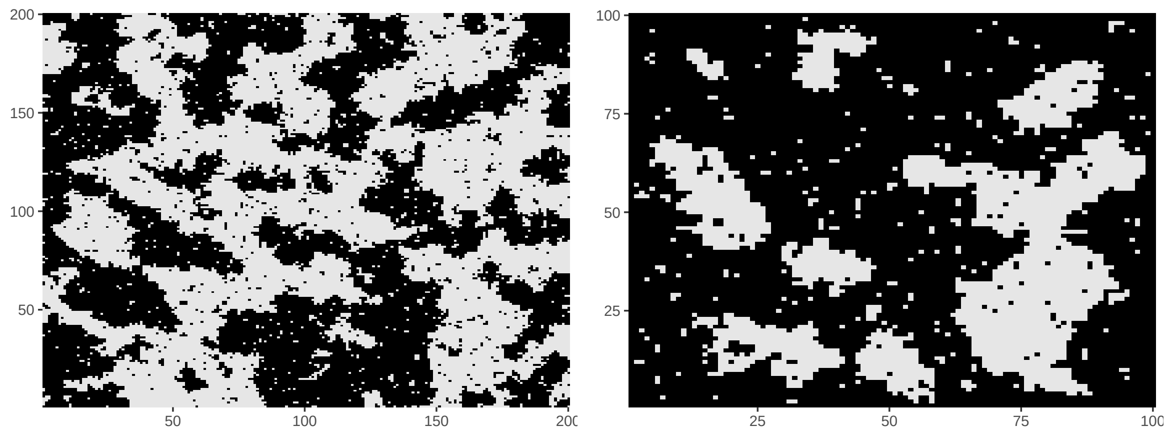


Figure 4: Simulated random fields from a nearest-neighbor structure. Left: no boundary conditions. Right: conditional to all border values being 0 (black).

```
R> border <- matrix(FALSE, nrow = 100, ncol = 100)
R> border[1, ] <- border[100, ] <- border[, 1] <- border[, 100] <- TRUE
R> initial <- matrix(sample(0:1, 100*100, replace = TRUE),
+    nrow = 100, ncol = 100)
R> initial[border] <- 0
R> z_border <- rmrf2d(initial, mrfi = mrfi(1), theta = th,
+    fixed_region = border)
```

Non-rectangular fields can be sampled either by passing a non-rectangular field as the `init_Z` argument or by using the `sub_region` argument and specifying the dimensions of the sampled field.

Another important feature is conditioning on a subset of pixel values. There are many situations where keeping a subset of pixels fixed during the sampling process can be useful, for example, filling a region of missing pixel values via simulation, defining boundary conditions (our model corresponds to a free boundary condition, but other types such as fixed or periodic boundary can be sampled with proper manipulation of the initial configuration and conditioning region) or performing block-wise updates of the data using conditionally independent blocks (for parallelization of algorithms).

Since the Gibbs sampler algorithm updates each pixel value multiple times, performance is one of our main implementation concerns. To improve the performance and speed up computations considerably, the internals of the sampling function, as well as most other computationally intensive functions are written in C++ with the use of **Rcpp** (Eddelbuettel and François 2011) and **RcppArmadillo** (Eddelbuettel and Sanderson 2014) packages.

3.4. Statistical inference in mrf2d

Inference methods for MRF models are diverse and their suitability highly depend on the type of data being analyzed. The framework provided by **mrf2d** can be used to implement all sorts of algorithms that are built from a common stack of components: simulation, conditional probabilities and sufficient statistics. It also provides complete built-in routines for some estimation algorithms.

Function	Use
Miscellaneous	
`rmrf2d`	Generates samples of a MRF via Gibbs sampler. Used for Monte Carlo based methods.
`cp_mrf2d`	Computes the conditional probabilities for a pixel position given its neighbors.
`pl_mrf2d`	Computes pseudo-likelihood value for an observed field considering interaction structure `mrfi` and array of potentials `theta`.
`cohist`	Creates the co-occurrence histogram of an observed field given an interaction structure. Can be converted to a vector of sufficient statistics given a restriction family with the `smr_stat` function.
`smr_array` and `expand_array`	Conversions between array and vector representation of potentials given a parameter restriction family.
Built-in inference algorithms	
`fit_pl`	Estimates the parameter array given an observed field via pseudo-likelihood optimization. Returns a `mrfout` object.
`fit_sa`	Estimates the parameter array given an observed field via stochastic approximation algorithm. Returns a `mrfout` object.
`fit_ghm`	Fits a Gaussian mixture driven by a given hidden MRF model using the EM algorithm from Zhang *et al.* (2001). `polynomial_2d` and `fourier_2d` can be used to create polynomial and 2-dimensional Fourier basis functions, respectively, to be used as a fixed effect. Returns a `hmrfout` object.

Table 3: List of available functions used for inference in **mrf2d** with a brief description of each one.

Table 3.4 presents a list of functions available in the package that can be used to construct inference algorithms, as well as built-in functions for parameter estimation for MRF and for the hidden MRF models defined in Section 2.3 that we describe next. The built-in inference functions return objects of class '`mrfout`' (MRF data) or '`hmrfout`' (hidden MRF models), which contains the information about the fitted model, as well as `summary` and `plot` methods associated for interpretation of the results.

Maximum pseudo-likelihood estimation

The pseudo-likelihood function in Equation 7 can be evaluated efficiently because it does not depend on the intractable normalizing constant. A common estimation procedure for intractable likelihood problems is optimizing the pseudo-likelihood with respect to the parameters. The pseudo-likelihood estimator is given by

$$\hat{\theta}_{PL} = \arg\max_{\theta} PL(\theta; \mathbf{z}). \tag{13}$$

The function `fit_pl()` from **mrf2d** implements an optimization of the pseudo-likelihood function using R built-in `optim()` function. It handles the conversions between array and vector representation of potentials automatically, respecting the restriction family selected and returns the estimated array of potentials and maximum value of the pseudo-likelihood in logarithmic scale. The arguments of `fit_pl` are:

- `Z`: The observed random field $\mathbf{z}$.
- `mrfi`: A `mrfi` representing an interaction structure $\mathcal{R}$.
- `family`: A parameter restriction family.
- `init`: An array with the initial configuration used in the optimization. `0` can be used to start from the independent model.
- `optim_args`: A named `list` with additional arguments passed to the `optim()` function call.

Stochastic approximation algorithm

Given an observed field $\mathbf{z}^{(0)}$, the stochastic approximation algorithm (Robbins and Monro 1951) seeks to create a Markov chain of parameter vectors $\{\boldsymbol{\theta}^{(t)}\}_{t\geq 1}$ that converges to the maximum likelihood estimate of $\boldsymbol{\theta}$, which is the solution of the zero gradient condition $\mathbb{E}_{\boldsymbol{\theta}}(S_{\mathcal{R}}(\mathbf{Z})) = S_{\mathcal{R}}(\mathbf{z}^{(0)})$, derived from Equation 9.
The algorithm is defined by the recurrence

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \gamma^{(t)}(S_{\mathcal{R}}(\mathbf{z}^{(0)}) - S_{\mathcal{R}}(\mathbf{z}^{(t)})), \tag{14}$$

where $\mathbf{z}^{(t)}$ is a field sampled using $\boldsymbol{\theta}^{(t)}$ and $\gamma^{(t)}$ is a sequence of positive constants that satisfies $\sum_{t=1}^{\infty} \gamma^{(t)} = \infty$ and $\sum_{t=1}^{\infty} \left(\gamma^{(t)}\right)^2 < \infty$.
Stochastic approximation is implemented in **mrf2d** as the `fit_sa()` function. It samples $\mathbf{z}^{(\mathbf{t})}$ via Gibbs sampler considering the previous field $\mathbf{z}^{(t-1)}$ as the initial configuration. Periodically, the field samples are refreshed, starting from an independent discrete uniform distribution and running a greater number of Gibbs sampler cycles, this procedure prevents the algorithms from getting stuck in problematic field samples. Its arguments are:

- `Z`: The observed field $\mathbf{z}^{(0)}$.
- `mrfi`: The interaction structure $\mathcal{R}$.
- `family`: The family of parameter restrictions considered when converting the potentials array to a vector.
- `gamma_seq`: A sequence of step size values to be used as $\gamma^{(t)}$. These values are divided by the number of pixels $|\mathcal{L}|$ internally to be invariant with respect to the image size. The typical sequence recommended is `seq(from = M, to = 0, length.out = B)`, with `M` ranging from 0.5 to 2 and large number of iterations (`B`).
- `init`: The initial array of parameters or the value `0` to start from the independent model.
- `cycles`: Number of Gibbs sampler ran between iterations.

- `refresh_each`: Restarts the sample $\mathbf{z}^{(t)}$ from a random configuration each `refresh_each` iterations.
- `refresh_cycles`: When a refresh happens, how many Gibbs sampler cycles are ran in the current parameter configuration.

Among the elements of the returned `mrfout` object, `theta` contains the estimated potential array and `metrics` a data frame with the Euclidean distances between $S_{\mathcal{R}}(\mathbf{z}^{(0)})$ and $S_{\mathcal{R}}(\mathbf{z}^{(t)})$ for each iteration. This sequence of distances is used to monitor the convergence of the algorithm as a form of diagnostics analysis.

EM algorithm for HMRF models

Gaussian Mixtures driven by hidden MRFs can be fitted in **mrf2d** with an extension of the EM algorithm from Zhang *et al.* (2001) to include a fixed effect (see Freguglia *et al.* 2020, for details). The probabilities computed for the latent label of each pixel in the E-step are conditioned on the global maximum probability configuration of its neighbors, obtained via iterated conditional modes (ICM) algorithm at each iteration (Besag 1986).

The complete algorithm is available in the `fit_ghm` function. Its main arguments are

- `Y`: The observed continuous-valued field $\mathbf{y}$.
- `mrfi`: Interaction structure of the latent field $\mathcal{R}$.
- `theta`: The array of potentials that defines the latent field distribution.
- `fixed_fn`: A `list` of functions of pixel positions $f(i_1, i_2)$ to be used as fixed effect. Constructors for 2-dimensional polynomials and Fourier basis are available in the functions `polynomial_2d` and `fourier_2d`, respectively.
- `equal_vars`: A `logical` value indicating if mixture components are forced to have equal variances.
- `init_mus` and `init_sigmas`: Optional initial values of $(\mu_a, \sigma_a)_{a=0,\dots,C}$. If none is passed, an independent Gaussian mixture is fitted with initial values based on quantiles and the estimates of this fitted model are used as (often good) starting values in the main procedure.
- `maxiter`: Maximum number of iterations before stopping.
- `max_dist`: Defines a stopping condition for the EM algorithm. For consecutive iterations t and $t+1$, the absolute difference in each parameter, $|\mu_k^{(t)} - \mu_k^{(t+1)}|$ and $|\sigma_k^{(t)} - \sigma_k^{(t+1)}|$ are computed for $k = 0, 1, \dots, C$. The algorithms stops if all differences are less than `max_dist`.
- `icm_cycles`: Number of cycles of Iterated Conditional Modes algorithm executed in each iteration.

`fit_ghm()` returns a `hmrfout` object represented by a `data.frame` containing `par` – the estimates of the mixture parameters $\{(\hat{\mu}_a, \hat{\sigma}_a), a = 0, 1 \dots, C\}$, `Z_pred` – the highest probability

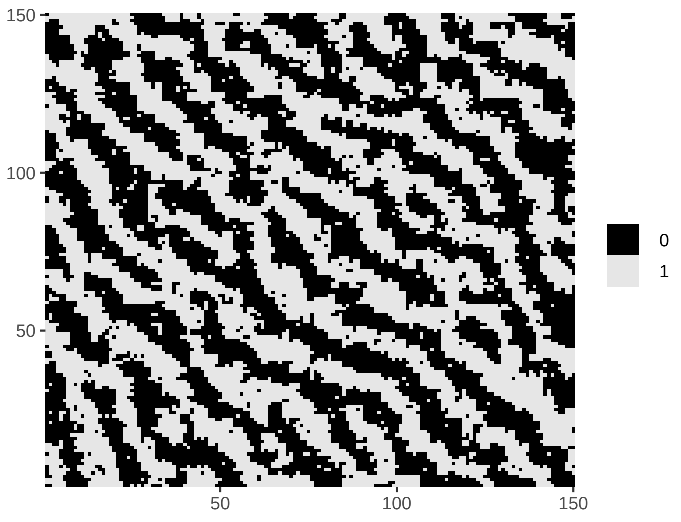


Figure 5: Visualization of `field1`.

configuration of the latent field computed via ICM algorithm $\hat{\mathbf{z}}$, `fixed` – a matrix with the estimated fixed effects $\left(\mathbf{x_i}^\top\hat{\beta}\right)$ for each pixel, and `predicted` – a matrix with the predicted mean for each pixel $\left(\mathbf{x_i}^\top\hat{\beta} + \hat{\mu}_{\hat{z}_\mathbf{i}}\right)$.

4. Data analysis using mrf2d

4.1. Example 1: A binary image with texture-like pattern

Description

To illustrate the usage of **mrf2d** for finite-valued images, we use the object `field1` available in the package, which contains a binary field with anisotropic pattern as seen in Figure 5. It is a synthetic texture image of the same type as the binary texture data presented in Cross and Jain (1983). The data can be loaded and viewed using the code chunk below.

```
R> data("field1", package = "mrf2d")
R> dplot(field1, legend = TRUE)
```

Our goal is to fit a MRF model to this data and sample images from the fitted model to evaluate if the patterns achieved in the generated data are similar to the original data. This is the typical setup of a texture synthesis problem with finite-valued images.

This analysis involves three main stages: Specifying the model (interaction structure and parameter restrictions), estimating the parameters and evaluating the fitted model. All of the run times described in this section were obtained using an Intel Core i7-7500U 2.70GHz CPU processor.

Specifying $\mathcal{R}$ and parameter family

Model selection under intractability is a challenging problem because most algorithms require comparing (maximum) likelihood functions for different models, what cannot be done exactly and/or have a high computational cost for MRF models.

The main routes in the model specification stage are: using prior information of the data or problem to select what type of restrictions and interaction structure are best suited, using the most general model (e.g., no restrictions and a complex interaction structure) as in Freguglia *et al.* (2020) or using some estimation technique.

This image presents a diagonal pattern what indicates a nearest-neighbor interaction structure may not be appropriate to capture all the dependence present in the field. We first choose what kind of parameter restriction family will be considered by checking that relabeling the values $\mathcal{Z}$ does not change the patterns in the image indicating symmetric potentials should be suited for this image, and there is a clear difference in the interactions when considering pixels in different directions, what indicates we need different types of interaction for different relative positions. These characteristics match the `"oneeach"` family that will be used in this example.

In order to estimate the set of interacting positions $\mathcal{R}$, we use use a naive algorithm which consists of performing 300 steps of stochastic approximation considering a large set of candidate interacting positions (all positions with maximum norm less or equal 6, 84 positions total) and then select the positions with absolute value of the associated potential higher than a threshold value. This is a strategy similar to the heuristic search algorithm from Gimel'farb (1996). Stochastic approximation was preferred over maximum pseudo-likelihood, for example, because it is computationally more suited for high-dimensional situations and we are not requiring a very accurate estimation at this point, so we can use a reasonably low number of iterations.

The code below implements this naive interaction selection algorithm in a few lines using the tools available in **mrf2d** considering a threshold value of 0.10. A large set of interaction position (`candidates`) is defined and the stochastic approximation algorithm is executed based on this complete interaction structure, obtaining `complete_sa`, then a selection based on thresholding absolute values of interactions is performed (`selected`).

```
R> candidates <- mrfi(6, norm_type = "m")
R> set.seed(1)
R> complete_sa <- fit_sa(field1, candidates, family = "oneeach",
+    gamma_seq = seq(from = 1, to = 0, length.out = 300),
+    cycles = 2, refresh_each = 301)
```

The complete stochastic approximation procedure in the `fit_sa()` call took 168 seconds to complete in total.

```
R> plot(complete_sa)
R> thr_value <- 0.1
R> theta_vec <- smr_array(complete_sa$theta, "oneeach")
R> selected <- which(abs(theta_vec) > thr_value)
R> R1 <- candidates[selected]
R> R1
```

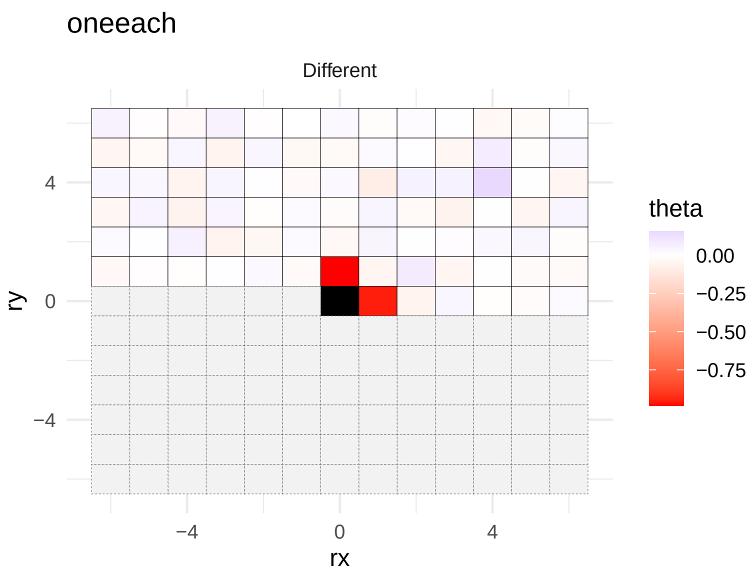


Figure 6: Plot of the `complete_sa` object. Nearest-neighbor positions have strong interactions (with negative potential) for different-valued pairs, while position $(4, 4)$ has a weaker positive potential. This can be interpreted as nearest-neighbors having more weight when they are equal, while pixel with relative position $(4, 4)$ have more weight when they are different.

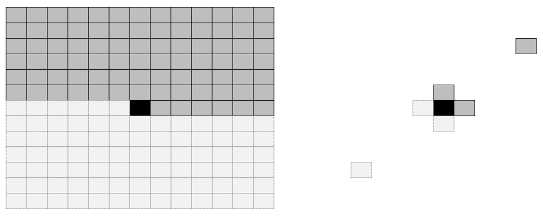


Figure 7: Candidate positions for the interaction structure (left) and selected positions (right).

```
3 interacting positions.
  rx     ry
   1      0
   0      1
   4      4
```

Estimating $\boldsymbol{\theta}$

Considering the parameter restriction family `"onepar"` and the selected interaction structure $\mathcal{R} = \{(1, 0), (0, 1), (4, 4)\}$, we have a model with 3 free parameters. A 3-dimensional optimization problem is simple enough to be solved using the built-in pseudo-likelihood optimization function. We also fit the model via stochastic approximation, now only considering the selected interaction structure, for comparison. The results are compared with the `summary()` method for the '`mrfout`' class and presented below.

```
R> pl <- fit_pl(field1, R1, family = "oneeach")
```

```
R> summary(pl)


Model adjusted via Pseudolikelihood
Image dimension: 150 150
2 colors, distributed as:
     0      1
 11083  11417

Interactions for different-valued pairs:
Position|  Value  Rel. Contribution
   (1,0)| -0.993  1.000 ***
   (0,1)| -1.021  0.995 ***
   (4,4)|  0.183  0.735 **


R> sa <- fit_sa(field1, R1, family = "oneeach",
+    gamma_seq = seq(from = 1, to = 0, length.out = 300))
R> summary(sa)


Model adjusted via Stochastic Approximation
Image dimension: 150 150
2 colors, distributed as:
     0      1
 11083  11417

Interactions for different-valued pairs:
Position|  Value  Rel. Contribution
   (1,0)| -0.964  1.000 ***
   (0,1)| -0.983  0.987 ***
   (4,4)|  0.183  0.756 ***
```

The resulting estimates were roughly the same using the two functions. The `fit_pl` call ran in 2.371 seconds while the `fit_sa` call took 56.627 seconds to complete. The optimization process from maximum pseudo-likelihood estimation was substantially faster, mainly due to the low-dimensionality of the problem, than running a satisfactory number of stochastic approximation steps to achieve reasonable precision.

Evaluating the fitted model

To evaluate how well the estimated parameters fit the data, we generate a new sample from the fitted model. Figure 8 shows the original image and the image simulated from the fitted model for comparison. The patterns created are visually very similar. Therefore the fitted MRF model successfully describes the characteristics of the data and is capable of synthesizing new images with the same texture pattern.

```
R> z_sim <- rmrf2d(dim(field1), R1, pl$theta)
```

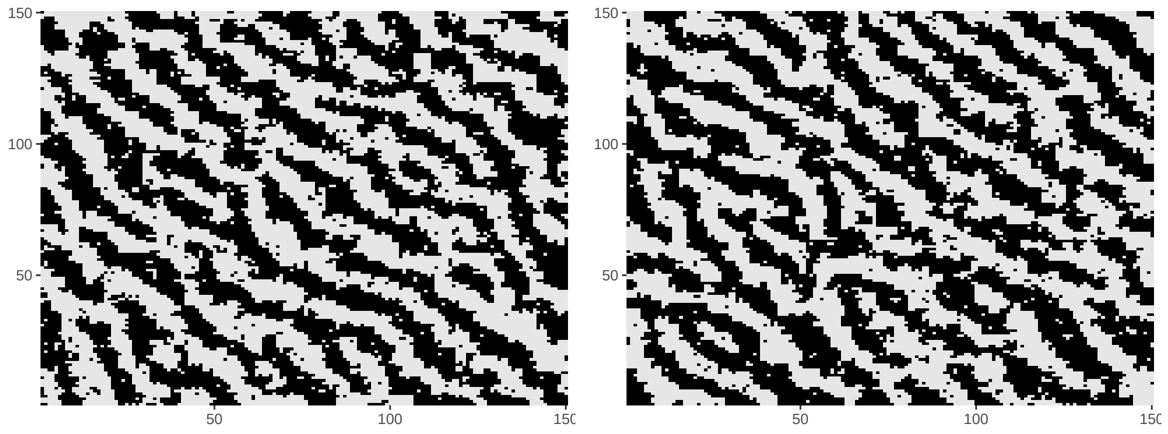


Figure 8: Original data (left) and random field simulated from the fitted model (right).

4.2. Example 2: Image segmentation of a hidden MRF

Description

The data available in the object `hfield1` in the package will be used to illustrate the use of **mrf2d** for Gaussian mixtures driven by hidden MRFs. It consists of an image with continuous-valued pixels ranging from 0.3 to 15.2. A pattern similar to the previous example can be observed with the addition of a continuous noise.

```
R> data("hfield1", package = "mrf2d")
R> cplot(hfield1)
```

We consider an image composed by a latent (hidden) MRF plus a random noise whose distribution for each pixel may depend on the pixel value in the latent field. The main goal for this type of data is to recover the segmentation of the underlying pixel labels, an image segmentation problem (Li, Wu, and Zhang 2009; Shah and Chauhan 2015, for example). This is a typical problem where Gaussian mixtures driven by hidden Markov random fields are well suited.

Fitting a hidden MRF with no fixed effect

The built-in function for fitting hidden MRFs (`fit_ghm()`), like most algorithms used for Gaussian mixtures driven by HMRFs, considers the distribution of the underlying field as a hyper-parameter specified a priori. In this example, since we observe an underlying pattern similar to one in Example 1, we will reuse the model estimated by penalized likelihood as the MRF distribution.

We fit a HMRF model to the data using the `fit_ghm` function. The mixture parameters estimates are shown below and the resulting segmentation is presented in Figure 10(b).

```
R> hmrf_nofixed <- fit_ghm(hfield1, mrfi = R1, theta = pl$theta)
R> summary(hmrf_nofixed)
```

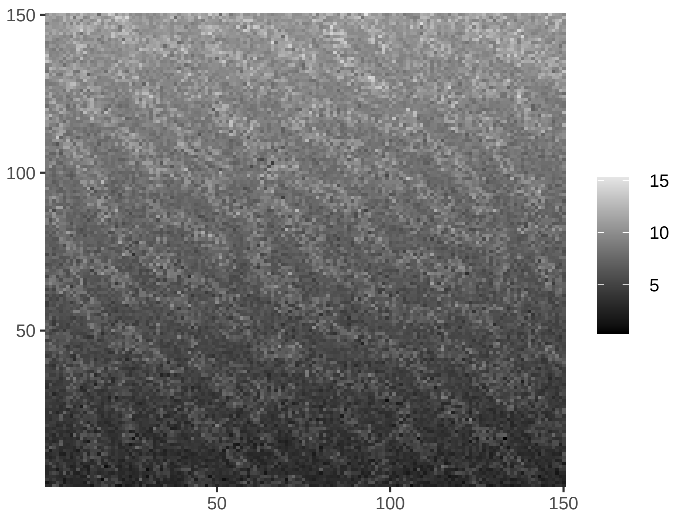


Figure 9: Image data for `hfield1`.

```
Gaussian mixture model driven by Hidden MRF fitted by EM-algorithm.
Image dimensions: 150 150
Predicted mixture component table:
     0      1
 10988  11512
Number of covariates (or basis functions): 0
Interaction structure considered: (1,0) (0,1) (4,4)

Mixture parameters:
 Component     mu  sigma
         0   5.29   1.39
         1   9.19   1.46

Model fitted in 4 iterations.
```

The labels in the segmentation follow the expected pattern only in the middle part of the image. Two large clusters without the pattern appear at the upper and lower parts of the image, what indicates there might be some missing spatial information not included in the model.

Adding a polynomial trend as fixed effect

A HMRF model without covariates has an intrinsic assumption that the mean values of pixel intensities, given their labels, are homogeneous along the image region. This is not the case for the data in Figure 9, as a vertical gradient effect can be observed.

In order to incorporate this spatial effect not captured in the model, we include spatial covariates, in the form of polynomial functions of pixel positions (i_1, i_2) as a fixed effect. These covariates can be specified in `fixed_fn` argument of the function and a (centered) polynomial can be created with the `polynomial_2d()` function from **mrf2d**.

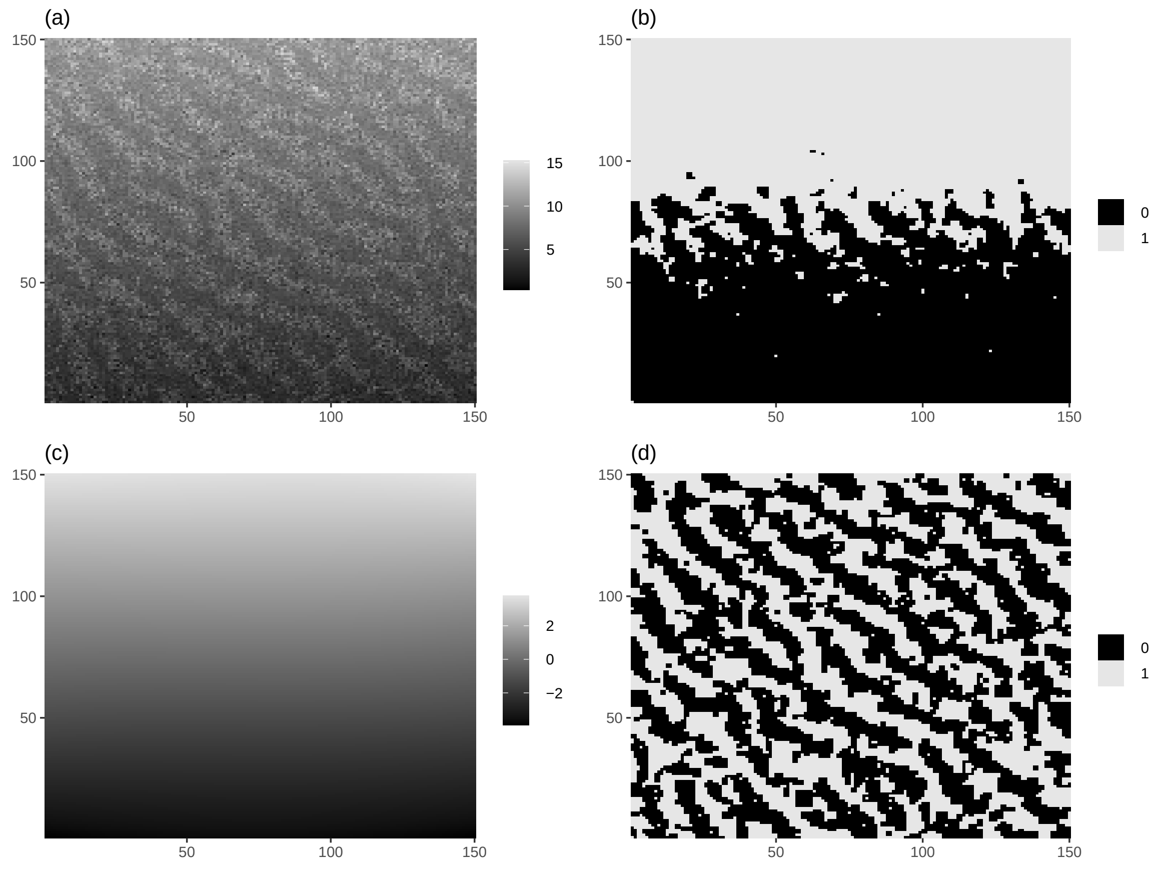


Figure 10: Results from the hidden MRF fits: (a) the original image data, (b) segmentation obtained without adding a polynomial effect, (c) polynomial fitted as a fixed effect, (d) image segmentation when the polynomial effect is included.

In this example, we include all terms of a two-dimensional centered cubic polynomial, that is,

$$p(i_1, i_2) = \sum_{d_1=0}^{3} \sum_{d_2=0}^{3} \beta_{i_1,i_2} (i_1 - c_1)^{d_1} (i_2 - c_2)^{d_2}, \tag{15}$$

where the centering position (c_1, c_2) is the middle pixel position of the image.

```
R> hmrf_poly <- fit_ghm(hfield1, mrfi = R1, theta = pl$theta,
+    fixed_fn = polynomial_2d(c(3, 3), dim(hfield1)))
R> summary(hmrf_poly)
```

```
Gaussian mixture model driven by Hidden MRF fitted by EM-algorithm.
Image dimensions: 150 150
Predicted mixture component table:
     0      1
 11718  10782
Number of covariates (or basis functions): 15
Interaction structure considered: (1,0) (0,1) (4,4)
```

```
Mixture parameters:
 Component     mu  sigma
         0   6.79   0.62
         1   7.82   1.20

Model fitted in 4 iterations.
```

The code chunk below illustrates how the resulting fields available in the function output can be visualized. The results are presented in Figure 10.

```
R> cplot(hfield1)
R> dplot(hmrf_nofixed$Z_pred, legend = TRUE)
R> cplot(hmrf_poly$fixed)
R> dplot(hmrf_poly$Z_pred, legend = TRUE)
```

This example highlights two features of **mrf2d** that are not available in other packages: The possibility to specify a distribution for the underlying field that is more flexible than a simple Potts model and the option to include covariates (in this example, the polynomial trend) that are estimated simultaneously to the mixture parameters, preventing undesired effects in the segmentation results.

4.3. Example 3: Neuroimaging segmentation with BOLD5000 data

Neuroimaging is one of the most frequent applications of HMRF models (Zhang *et al.* 2001; Shah and Chauhan 2015). We illustrate a brain magnetic resonance image segmentation using a sample of the BOLD5000 dataset (Chang, Pyles, Marcus, Gupta, Tarr, and Aminoff 2019) available in the `bold5000` object in the package.

```
R> data("bold5000", package = "mrf2d")
R> cplot(bold5000)
```

Our main goal in this problem is to segment the brain image into large regions corresponding to different elements, like a background, bones, fat, grey matter, white matter, etc.

The most common approach for the segmentation using HMRFs is to consider a simple Potts model (nearest-neighbor interaction structure $\mathcal{R} = \{(1,0),(0,1)\}$ and the `"onepar"` parameter restriction family. The potential associated with different-valued pairs controls, as well as the number of components are considered fixed a priori and will not be discussed in this paper. For the purpose of illustration, we use 4 components ($C = 3$) and the value -1 for the potentials of different-valued pairs.

```
R> Rnn <- mrfi(1)
R> theta_nn <- expand_array(-1, family = "onepar", C = 3, mrfi = Rnn)
```

We add a constraint that all variance parameters of the mixture components must be equal by setting the `equal_vars` parameter to `TRUE`. This improves the results in this problem by preventing some of the mixture components to be estimated with too high variance, what may causes pixels with large and small values to be predicted in the same class with high probability.

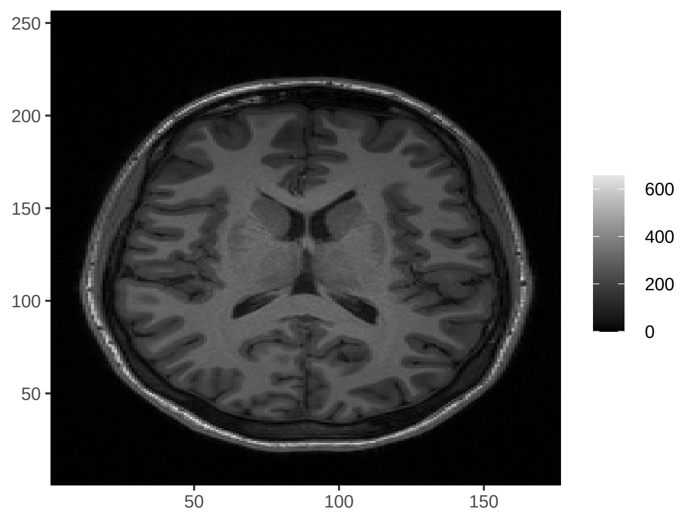


Figure 11: Brain magnetic resonance image in `bold5000` data.

We also fit an independent Gaussian mixture (by multiplying all potentials by zero) for a comparison with the HMRF model. Segmentation results are presented in Figure 12 and the parameter estimates are shown below.

```
R> set.seed(1)
R> fit_brain <- fit_ghm(bold5000, Rnn, theta_nn, equal_vars = TRUE)
R> fit_brain_ind <- fit_ghm(bold5000, Rnn, theta_nn*0, equal_vars = TRUE)
```

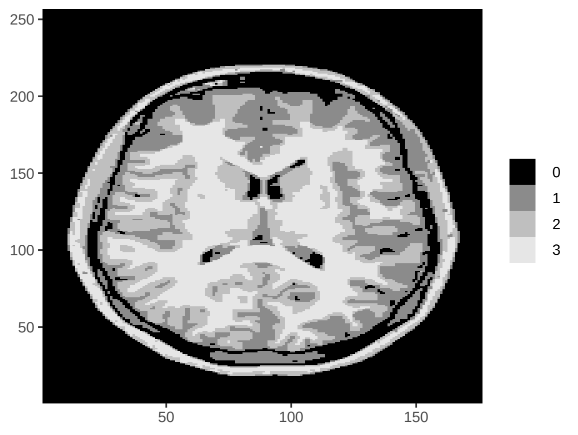
```
R> summary(fit_brain)
```

```
Gaussian mixture model driven by Hidden MRF fitted by EM-algorithm.
Image dimensions: 176 256
Predicted mixture component table:
     0      1      2      3
 22921   6829   7305   8001
Number of covariates (or basis functions): 0
Interaction structure considered: (1,0) (0,1)

Mixture parameters:
 Component     mu  sigma
         0   7.23  28.78
         1 128.90  28.78
         2 207.53  28.78
         3 294.71  28.78

Model fitted in 11 iterations.
```

```
R> summary(fit_brain_ind)
```



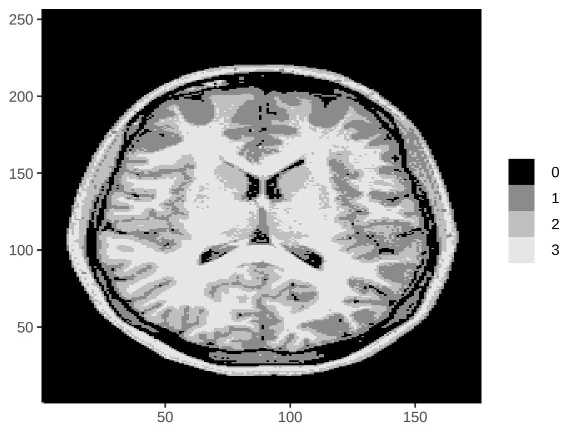


Figure 12: Image segmentation predicted by the hidden MRF fitted (left) and the independent mixture model (right).

```
Gaussian mixture model driven by Hidden MRF fitted by EM-algorithm.
Image dimensions: 176 256
Predicted mixture component table:
     0      1      2      3
 23013   7021   7065   7957
Number of covariates (or basis functions): 0
Interaction structure considered: (1,0) (0,1)

Mixture parameters:
 Component     mu  sigma
         0   7.51  30.23
         1 133.10  30.23
         2 211.72  30.23
         3 295.33  30.23

Model fitted in 4 iterations.
```

The resulting parameter estimates are not much different when comparing the independent mixture model and the HMRF and both ran in approximately 85 seconds, but the segmentation is cleaner when using the HMRF model, without sparse different-labeled pixels inside regions.

```
R> dplot(fit_brain$Z_pred, legend = TRUE)
R> dplot(fit_brain_ind$Z_pred, legend = TRUE)
```

5. Discussion

mrf2d provides a consistent programming interface for statistical inference in a large class of discrete Markov random field models defined on 2-dimensional lattices. It has an efficient and simple to use implementation of the main stack of computations used by most of inference algorithms, as well as complete routines for some commonly used and more complex estimation methods. The objects used for representing each model component have been carefully designed and tuned over several iterations to achieve a balance between performance and usability in the stable version.

The model featured in the package generalizes Potts model from other available packages in different ways, such as allowing a flexible definition of interacting pixel positions and interaction types, with the drawback that it cannot take advantage of algorithms that require the setup of a Potts model to improve their efficiency.

The versatility from the non-parametric model behind **mrf2d** and the flexible representation proposed in the package allows us to create special 2-dimensional structures that are equivalent to 3-dimensional representations of data. This can make **mrf2d** also an interesting option for applications with 3-dimensional data where the assumptions of the model also hold. A vignette is available with a deeper explanation on how to reshape 3-dimensional problems for the **mrf2d** framework and it can be viewed by using the `vignette()` function.

```
R> vignette("three-dimensions-on-mrf2d", package = "mrf2d")
```

We currently have over 160 unit tests supported by the **testthat** package (Wickham 2011) and more than 90% of the code covered in the tests. These tests were designed to verify mathematical correctness of functions, the behavior of functions with unexpected input and the consistency of error messages. The package is in constant development and new tests are added whenever new functionalities are implemented to ensure its reliability over time.

For these reasons, **mrf2d** is an important tool for making statistical inference in images using Markov random fields more accessible, allowing researchers to perform data analysis and implement new algorithms in R with a simple and consistent framework.

Acknowledgments

This work was funded by Fundação de Amparo à Pesquisa do Estado de São Paulo – FAPESP grant 2017/25469-2 and by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001.

We are thankful to the editors and reviewers for their valuable comments that were essential to improve the manuscript, especially the anonymous referee who commented on 3-dimensional applications that motivated us to add the new vignette.

References

Besag J (1974). "Spatial Interaction and the Statistical Analysis of Lattice Systems." *Journal of the Royal Statistical Society B*, **36**(2), 192–225. doi:10.1111/j.2517-6161.1974.tb00999.x.

Besag J (1975). "Statistical Analysis of Non-Lattice Data." *Journal of the Royal Statistical Society D*, **24**(3), 179–195. doi:10.2307/2987782.

Besag J (1986). "On the Statistical Analysis of Dirty Pictures." *Journal of the Royal Statistical Society B*, **48**(3), 259–279. doi:10.1111/j.2517-6161.1986.tb01412.x.

Beyer L (2015). "**PyDenseCRF**." URL https://github.com/uzeful/pydensecrf.

Bhattacharya BB, Mukherjee S (2018). "Inference in Ising Models." *Bernoulli*, **24**(1), 493–525. doi:10.3150/16-bej886.

Blake A, Kohli P, Rother C (2011). *Markov Random Fields for Vision and Image Processing.* MIT Press. doi:10.7551/mitpress/8579.001.0001.

Butts CT (2008). "**network**: A Package for Managing Relational Data in R." *Journal of Statistical Software*, **24**(2). doi:10.18637/jss.v024.i02.

Cao X, Zhou F, Xu L, Meng D, Xu Z, Paisley J (2018). "Hyperspectral Image Classification with Markov Random Fields and a Convolutional Neural Network." *IEEE Transactions on Image Processing*, **27**(5), 2354–2367. doi:10.1109/tip.2018.2799324.

Chang N, Pyles JA, Marcus A, Gupta A, Tarr MJ, Aminoff EM (2019). "BOLD5000, a Public fMRI Dataset While Viewing 5000 Visual Images." *Scientific Data*, **6**(1), 1–18. doi:10.1038/s41597-019-0052-3.

Clark NJ, Wells K, Lindberg O (2018). "Unravelling Changing Interspecific Interactions Across Environmental Gradients Using Markov Random Fields." *Ecology*, **99**(6), 1277–1283. doi:10.1002/ecy.2221.

Cross GR, Jain AK (1983). "Markov Random Field Texture Models." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **5**(1), 25–39. doi:10.1109/tpami.1983.4767341.

De Bastiani F, Rigby RA, Stasinopoulous DM, Cysneiros AH, Uribe-Opazo MA (2018). "Gaussian Markov Random Field Spatial Models in GAMLSS." *Journal of Applied Statistics*, **45**(1), 168–186. doi:10.1080/02664763.2016.1269728.

Derin H, Elliott H (1987). "Modeling and Segmentation of Noisy and Textured Images Using Gibbs Random Fields." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **9**(1), 39–55. doi:10.1109/tpami.1987.4767871.

Eddelbuettel D, François R (2011). "**Rcpp**: Seamless R and C++ Integration." *Journal of Statistical Software*, **40**(8), 1–18. doi:10.18637/jss.v040.i08.

Eddelbuettel D, Sanderson C (2014). "**RcppArmadillo**: Accelerating R with High-Performance C++ Linear Algebra." *Computational Statistics & Data Analysis*, **71**, 1054–1063. doi:10.1016/j.csda.2013.02.005.

Feng D (2018). ***PottsUtils**: Utility Functions of the Potts Models.* R package version 0.3-3, URL https://CRAN.R-project.org/package=PottsUtils.

Freeman WT, Liu C (2011). "Markov Random Fields for Super-Resolution and Texture Synthesis." In *Advances in Markov Random Fields for Vision and Image Processing*. MIT Press.

Freguglia V (2022). ***mrf2d**: Markov Random Field Models for Image Analysis.* R package version 1.0, URL https://CRAN.R-project.org/package=mrf2d.

Freguglia V, Garcia NL, Bicas JL (2020). "Hidden Markov Random Field Models Applied to Color Homogeneity Evaluation in Dyed Textile Images." *Environmetrics*, **31**(4), e2613. doi:10.1002/env.2613.

Geman S, Geman D (1984). "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **6**(6), 721–741. doi:10.1109/tpami.1984.4767596.

Gentleman R, Whalen E, Huber W, Falcon S (2021). ***graph**: A Package to Handle Graph Data Structures.* R package version 1.72.0, URL https://bioconductor.org/packages/graph/.

Geyer CJ, Johnson L (2020). ***potts**: Markov Chain Monte Carlo for Potts Models.* R package version 0.5-9, URL https://CRAN.R-project.org/package=potts.

Geyer CJ, Thompson EA (1992). "Constrained Monte Carlo Maximum Likelihood for Dependent Data." *Journal of the Royal Statistical Society B*, **54**(3), 657–683. doi:10.1111/j.2517-6161.1992.tb01443.x.

Ghamisi P, Maggiori E, Li S, Souza R, Tarablaka Y, Moser G, De Giorgi A, Fang L, Chen Y, Chi M, Serpico SB, Benediktsson JA (2018). "New Frontiers in Spectral-Spatial Hyperspectral Image Classification: The Latest Advances Based on Mathematical Morphology, Markov Random Fields, Segmentation, Sparse Representation, and Deep Learning." *IEEE Geoscience and Remote Sensing Magazine*, **6**(3), 10–43. doi:10.1109/mgrs.2018.2854840.

Gimel'farb GL (1996). "Texture Modeling by Multiple Pairwise Pixel Interactions." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **18**(11), 1110–1114. doi:10.1109/34.544081.

Guillot D, Rajaratnam B, Emile-Geay J (2015). "Statistical Paleoclimate Reconstructions via Markov Random Fields." *The Annals of Applied Statistics*, **9**(1), 324–352. doi:10.1214/14-aoas794.

Hammersley JM, Clifford P (1971). "Markov Fields on Finite Graphs and Lattices." Unpublished Manuscript.

Jensen JL, Künsch HR (1994). "On Asymptotic Normality of Pseudo Likelihood Estimates for Pairwise Interaction Processes." *Annals of the Institute of Statistical Mathematics*, **46**(3), 475–486. URL https://link.springer.com/article/10.1007/BF00773511.

Kato Z, Pong TC (2006). "A Markov Random Field Image Segmentation Model for Color Textured Images." *Image and Vision Computing*, **24**(10), 1103–1114. doi:10.1016/j.imavis.2006.03.005.

Kato Z, Zerubia J (2012). "Markov Random Fields in Image Segmentation." *Foundations and Trends in Signal Processing*, **5**(1–2), 1–155. doi:10.1561/2000000035.

Kosov S (2013). "Direct Graphical Models C++ Library." URL http://research.project-10.de/dgm/.

Krähenbühl P, Koltun V (2011). "Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials." In *Advances in Neural Information Processing Systems*, pp. 109–117.

Li M, Wu Y, Zhang Q (2009). "SAR Image Segmentation Based on Mixture Context and Wavelet Hidden-Class-Label Markov Random Field." *Computers & Mathematics with Applications*, **57**(6), 961–969. doi:10.1016/j.camwa.2008.10.042.

Liggett TM (2012). *Interacting Particle Systems*, volume 276. Springer-Verlag.

Møller J, Pettitt AN, Reeves R, Berthelsen KK (2006). "An Efficient Markov Chain Monte Carlo Method for Distributions with Intractable Normalising Constants." *Biometrika*, **93**(2), 451–458. doi:10.1093/biomet/93.2.451.

Moores M, Nicholls G, Pettitt A, Mengersen K (2020). "Scalable Bayesian Inference for the Inverse Temperature of a Hidden Potts Model." *Bayesian Analysis*, **15**(1), 1–27. doi:10.1214/18-ba1130.

Pickard DK (1987). "Inference for Discrete Markov Fields: The Simplest Nontrivial Case." *Journal of the American Statistical Association*, **82**(397), 90–96. doi:10.1080/01621459.1987.10478394.

Potts RB (1952). "Some Generalized Order-Disorder Transformations." In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 48, pp. 106–109. Cambridge University Press.

R Core Team (2021). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Robbins H, Monro S (1951). "A Stochastic Approximation Method." *The Annals of Mathematical Statistics*, pp. 400–407. doi:10.1214/aoms/1177729586.

Roche A, Ribes D, Bach-Cuadra M, Krüger G (2011). "On the Convergence of EM-Like Algorithms for Image Segmentation Using Markov Random Fields." *Medical Image Analysis*, **15**(6), 830–839. doi:10.1016/j.media.2011.05.002.

Rue H, Martino S, Chopin N (2009). "Approximate Bayesian Inference for Latent Gaussian Models by Using Integrated Nested Laplace Approximations." *Journal of the Royal Statistical Society B*, **71**(2), 319–392. doi:10.1111/j.1467-9868.2008.00700.x.

Shah SA, Chauhan NC (2015). "An Automated Approach for Segmentation of Brain MR Images Using Gaussian Mixture Model Based Hidden Markov Random Field with Expectation Maximization." *Journal of Biomedical Engineering and Medical Imaging*, **2**(4), 57–57. doi:10.14738/jbemi.24.1411.

Stoehr J, Pudlo P, Friel N (2020). ***GiRaF**: A Toolbox for Gibbs Random Fields Analysis*. R package version 1.0.1, URL https://CRAN.R-project.org/package=GiRaF.

Storath M, Weinmann A (2014). "Fast Partitioning of Vector-Valued Images." *SIAM Journal on Imaging Sciences*, **7**(3), 1826–1852. doi:10.1137/130950367.

Versteegen R, Gimel'farb G, Riddle P (2016). "Texture Modelling with Nested High-Order Markov-Gibbs Random Fields." *Computer Vision and Image Understanding*, **143**, 120–134. doi:10.1016/j.cviu.2015.11.003.

Wang JS, Swendsen RH (1990). "Cluster Monte Carlo Algorithms." *Physica A: Statistical Mechanics and Its Applications*, **167**(3), 565–579. doi:10.1016/0378-4371(90)90275-w.

Wickham H (2011). "**testthat**: Get Started with Testing." *The R Journal*, **3**, 5–10. doi:10.32614/rj-2011-002.

Wickham H (2016). ***ggplot2**: Elegant Graphics for Data Analysis.* Springer-Verlag, New York.

Winkler G (2012). *Image Analysis, Random Fields and Markov Chain Monte Carlo Methods: A Mathematical Introduction*, volume 27. Springer-Verlag.

Wood SN (2017). *Generalized Additive Models: An Introduction with R*. 2nd edition. Chapman & Hall/CRC.

Wu LY (2019). ***CRF**: Conditional Random Fields.* R package version 0.4-3, URL https://CRAN.R-project.org/package=CRF.

Zhang Y, Brady M, Smith S (2001). "Segmentation of Brain MR Images through a Hidden Markov Random Field Model and the Expectation-Maximization Algorithm." *IEEE Transactions on Medical Imaging*, **20**(1), 45–57. doi:10.1109/42.906424.

A. Customizing visualizations with ggplot2

A.1. Random field visualization

The two plotting functions `dplot()` and `cplot()` return an object of the '`ggplot`' class, what allows users to produce customized visualizations by changing scales, legend characteristics, titles, themes and much more by using the grammar of graphics from the `ggplot2` package (Wickham 2016).

Random fields (represented by `matrix` objects) are transformed into a `data.frame` structure with columns `x`, `y` and `value`. `x` and `y` are the indices of the matrix object while `value` maps the pixel-value in that position (`Z[x,y]`).

The plots are constructed using a tile plane with rectangles (`geom_tile()` from **ggplot2**) with the `value` column map to the `fill` aesthetics. In `dplot()`, which is used for finite-valued fields, `value` is treated as a factor, while in `cplot()` it is a continuous `numeric`. This is the only difference between the functions and it should be kept in mind when defining custom color scales.

Figure 13 created with the code chunk below shows examples of customized versions of a random field visualization built from the same base `dplot()` result. Modifications include adding a title, removing all scale-related information (keeping only the actual image), using a custom color-scale and changing the legend position, respectively.

```
R> library("ggplot2")
R> base_plot <- dplot(field1, legend = TRUE)
R> base_plot + ggtitle("This is a custom title")
R> base_plot + theme_void() + theme(legend.position = "none")
R> base_plot + scale_fill_manual(values = c("red", "blue"))
R> base_plot + theme(legend.position = "bottom")
```

A.2. Interaction structure visualization

Similarly to the functions used to visualize random fields, a `plot()` method is available for `mrfi` objects. A `data.frame` with columns named `rx` and `ry` is created internally with the coordinates of each interacting relative position. These columns are mapped to the `x` and `y` axis, respectively and a `geom_tile` is used to produce the plot. The reverse positions are included automatically with a light-gray color, but this can be prevented by setting `include_opposite = FALSE` in the `plot` call. It also returns a `ggplot` object that can be customized. The code chunk next presents examples of how plots can be customized by adding custom colors, text labels to the interacting positions and a custom title.

```
R> mrfi_plot <- plot(mrfi(3) + c(5,1))

R> mrfi_plot + geom_tile(fill = "orange", color = "blue")
R> mrfi_plot + geom_text(aes(label = paste0("(", rx, ",", ry, ")")))
R> mrfi_plot + ggtitle("Add a custom title") +
+    theme(plot.title = element_text(hjust = 0.5, size = 24))
```

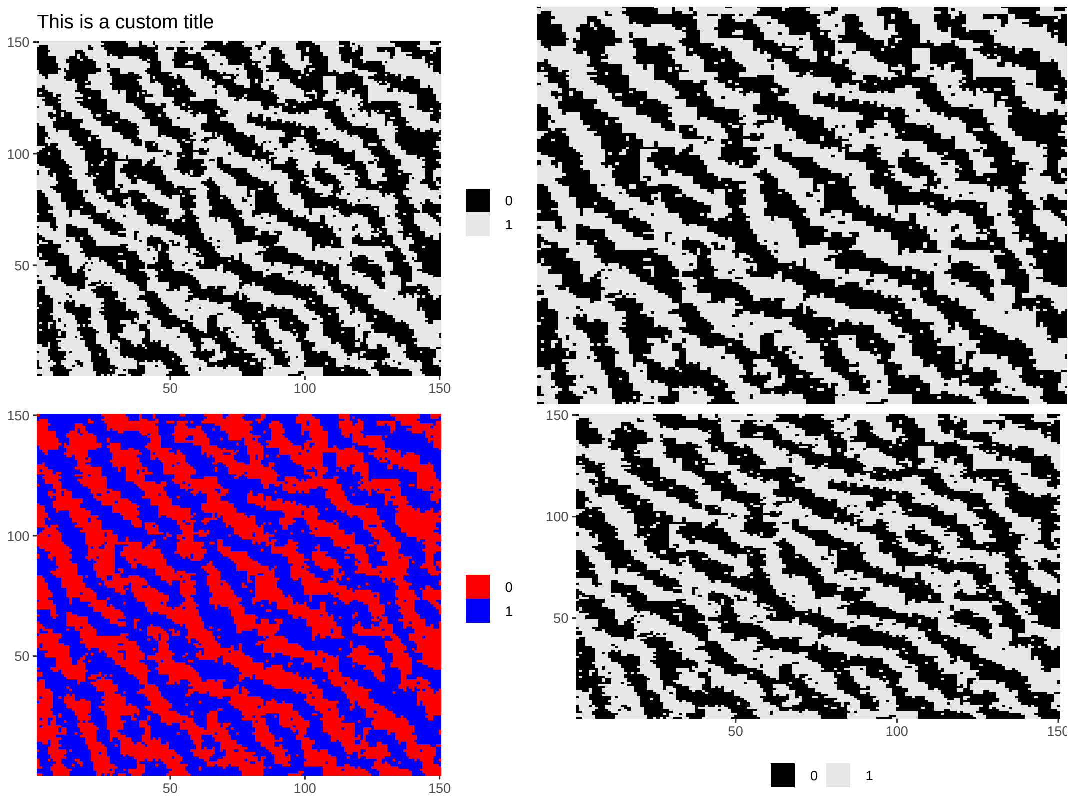


Figure 13: Four examples of field visualizations achieved by adding **ggplot2** layers to a base plot produced in **mrf2d**.

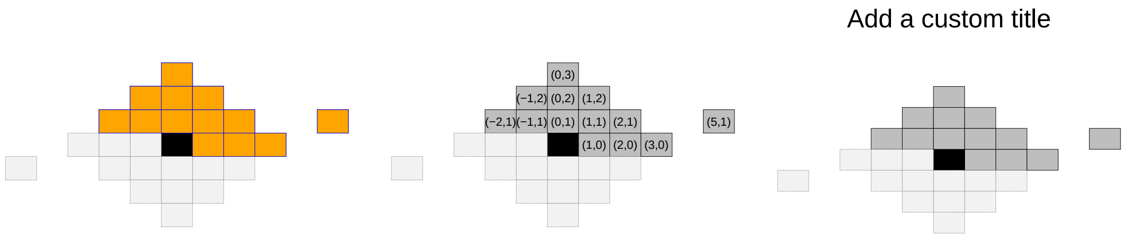


Figure 14: Examples of customized visualizations of `mrfi` objects.

B. Comparing methods between packages for a Potts model

One of the most important particular cases for the model presented in Section 2.1.1 is the Potts model, which is used in the theory supporting many available packages. In particular, the **potts** package (Geyer and Johnson 2020) provides functionalities similar to some of the ones implemented in **mrf2d**, mainly random sampling and computing the composite likelihood (comparable to the pseudo-likelihood function). The main difference between the packages is that **mrf2d** uses a Gibbs sampler scheme, with sequential updates of individual pixel while **potts** uses the Swendsen-Wang algorithm (Wang and Swendsen 1990), that updates blocks of pixels at each iteration.

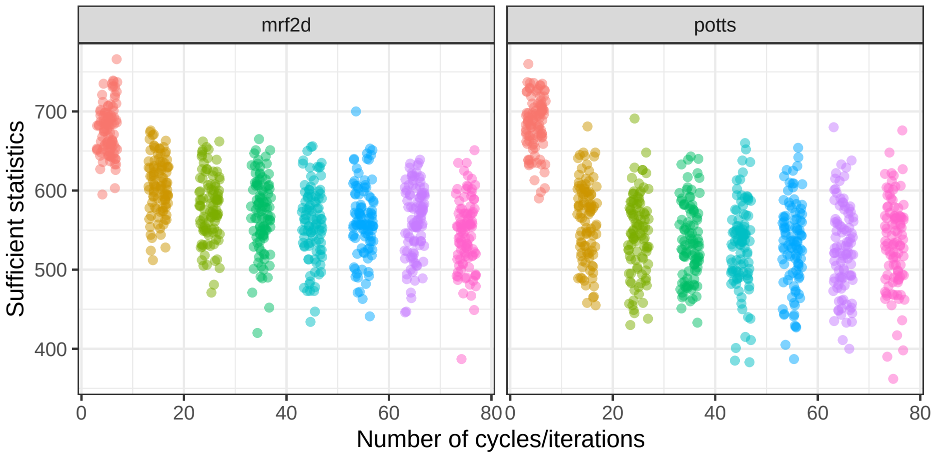


Figure 15: Sufficient statistics for 100 simulated realizations of the Potts Model with varying number of iterations using both **mrf2d** and **potts** packages.

We conducted a simulation study in order to compare **mrf2d** and the **potts** package version 0.5-9 with respect to the Potts model simulation algorithms. Our goal was not to compare implementations as the algorithms are different, but to check whether the simulated fields are comparable between packages as a form of validation and understanding the behavior of our simulation method with respect to the number of Gibbs sampler cycles.

Considering the parametrization of the Potts model in Equation 4, 100 realizations of a three-color ($C = 2$) Potts model with parameter $\phi = -1$ (corresponds $+1$ in the parametrization used in the **potts** package) were simulated considering number of iterations equal to 5, 15, 25, 35, 45, 55, 65 and 75 in each package. We use the term iteration referring to the `cycles` parameter in **mrf2d** and `nspace` in **potts**. For each simulated field, we computed the sufficient statistics $T(\mathbf{z}) = \sum_{||\mathbf{i}-\mathbf{j}||=1} \mathbb{1}_{(z_\mathbf{i} \neq z_\mathbf{j})}$. The results are presented in Figure 15 and the code for the simulations is available in the replication script for this article.

We can consider the algorithms achieved equilibrium when executing more iterations do not change the distribution of the samples, which we summarize by the distribution of their sufficient statistics. In this case, the Gibbs sampler from **mrf2d** seems to achieve equilibrium within 35 to 45 cycles while roughly a value of 25 for the `nspace` parameter of **potts** seems to be enough to achieve equilibrium in the Swendsen-Wang algorithm. Finally, we verify that the distribution of the sufficient statistics after reaching an equilibrium state is approximately the same in both packages and we conclude that even though **potts** uses the more specialized and efficient Swendsen-Wang algorithm, **mrf2d** with a Gibbs sampler provides a valid alternative simulation tool for the model that can also be extended, even within the scope of the Potts model, to cases such as positive ϕ, not covered by the Swendsen-Wang algorithm.

We also compare estimation algorithms in both packages for the maximum pseudo-likelihood method, which is a particular case of composite likelihood, thus available in **potts**. We simulated 100 realizations of a 3 color Potts model in a 64×64 lattice again with the parameter $\phi = -1$. For each realization, we computed maximum pseudo-likelihood estimates using **mrf2d** and **potts** to compare the results. Results are presented in Figure 16. In our comparison, estimates obtained with **potts** were consistently greater (smaller absolute value) than

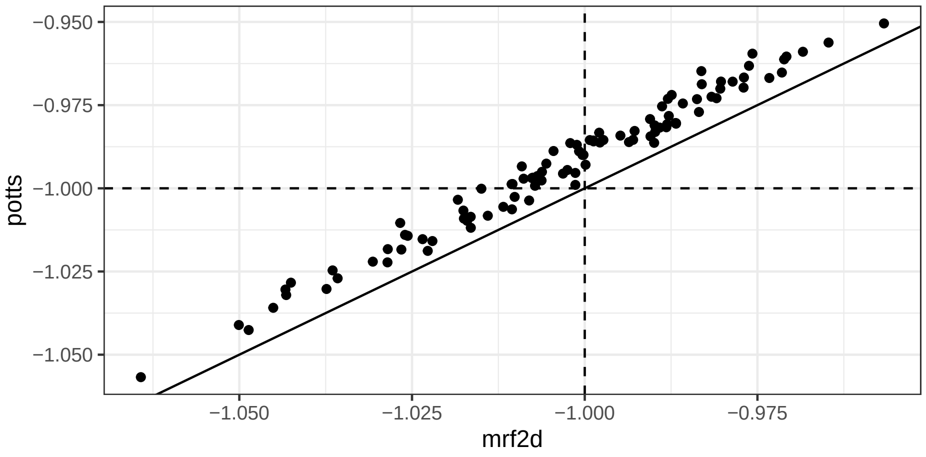


Figure 16: Estimated parameter via pseudo-likelihood optmization for 100 simulated realizations of the Potts model using the **mrf2d** and **potts** packages. The solid line represents the $y = x$ equation and dashed lines mark the parameter value used in the simulations.

the ones obtained via **mrf2d**. The average estimate for **mrf2d** was −1.0027996 while for **potts** the average was −0.9930211, with sample deviations equal to 0.0213258 and 0.0216528, respectively. We conclude that maximum pseudo-likelihood estimation methods are roughly equivalent in both packages.

Affiliation:

Victor Freguglia, Nancy Lopes Garcia
University of Campinas
Institute of Mathematics, Statistics and Scientific Computation
Campinas, Brazil
E-mail: victorfreguglia@gmail.com, nancyg@unicamp.br
URL: https://freguglia.github.io

Journal of Statistical Software https://www.jstatsoft.org/
published by the Foundation for Open Access Statistics https://www.foastat.org/
January 2022, Volume 101, Issue 8 *Submitted:* 2020-06-03
doi:10.18637/jss.v101.i08 *Accepted:* 2021-06-05

3 Sparse Interaction Neighborhood Selection for Markov Random Fields via Reversible Jump and Pseudoposteriors

3.1 Introduction

Markov Random Fields on two-dimensional lattices are popular probabilistic models for describing features of digital images in a wide range of applications. Classical problems like image segmentation rely on these models to describe unobserved variables used for pixel classification (Held et al., 1997; Zhang et al., 2001), while more general inference-oriented models describe pixel values directly as a Markov Random Field, for example, in texture modeling problems (Hassner and Sklansky, 1981; Cross and Jain, 1983).

One of the main inferential challenges in Markov Random Fields is the fact that these models have their dependence structure implicitly described by a graph which, in most cases, contains cycles that prevents expressing the likelihood function as a product of simpler conditional probabilities as in classical Markov Chain models. The impossibility of decomposing the joint probability of a high-dimensional random vector into simpler pieces makes a high-dimensional integral (or sum) required in order to compute the normalizing constant of those probability measures. In general, the normalizing constant directly depends on the parameters of the distribution, thus being an important part of likelihood-based analyses. Whenever this high-dimensional integral cannot be computed in reasonable time, often due to the exponential complexity of a non-independent high-dimensional space, the likelihood function becomes intractable, rendering most of the usual inference and model selection techniques unusable.

Inference under intractable likelihoods is a key topic for analyzing high-dimension data with local dependence. In particular, in a Bayesian context, Monte Carlo Markov Chain (MCMC) methods that generate samples from the posterior distributions under intractability have been developed using different strategies, such as including additional random elements with particular distributions that lead to convenient analytical properties that cancels out the intractable constant (Murray et al., 2012) or generating samples from model configurations that help producing approximations for the intractable likelihood function at each step of the MCMC algorithm (Atchadé et al., 2013) or prior to the Markov Chain iterations (Boland et al., 2018).

Another frequently used approach is to directly substitute the Likelihood function term that appears in the posterior distribution for the Pseudolikelihood function, introduced in Besag (1975), resulting in an analysis based not on the posterior distribution, but on a function that is referred as the Pseudoposterior distribution. While the Pseudolikelihood function may differ from the actual Likelihood function in important characteristics such as its mode, inference methods based on Pseudolikelihoods have theoretical results available and demonstrated practical usefulness, including in Bayesian contexts, often with adjustments such as in Bouranis et al. (2017). In the context of determining the basic neighborhood for MRF pseudolikelihoods have been used by several authors. Csiszár and Talata (2006) proved that a modification of the Bayesian Information Criteria replacing the likelihood by the pseudolikelihood provides strongly consistent estimators of the basic neighborhood from a single realization of the process observed at increasing regions. Under the Bayesian paradigm, marginal pseudoposterior likelihood has been used to find the dependence structure in Markov networks, see Pensar et al. (2017) and references therein.

Model selection is another highly important topic that has most of its common methodologies unusable in Markov Random Field models due to the intractability of likelihoods, and using an approximation of the likelihood function removes most of the probabilistic properties which model selection and hypothesis testing in general rely on. One advantage of studying complex models under a Bayesian framework is that the space of unobserved random quantities may be extended to include not only a vector of real-valued parameters, but also more general objects that can represent models, such as subsets of an arbitrary parameter space, and obtaining distributions from these general objects using MCMC methods tends to be more feasible than constructing efficient optimizations algorithm in such arbitrary spaces. The Reversible-Jump Monte Carlo Markov Chain (RJMCMC) methods, proposed in Green (1995), provide a framework for constructing a Markov Chain with an arbitrary invariant distribution on general spaces, including varying-dimensional parameter spaces that are highly useful for variable and model selection under a Bayesian philosophy, but these methods require a careful construction of a proposal kernel in order to be efficient.

Pseudolikelihood-based methods have been used for model selection purposes in Ji and Seymour (1996) and in recent works considering the Reversible Jump as a strategy for model selection with intractable likelihoods in a Bayesian context including Arnesen and Tjelmeland (2017), which aims to select the dependence structure of an Ising model among a small set of possible candidates with low-dimensional parameter spaces, and Bouranis et al. (2018) that proposes a RJMCMC model selection procedure for the exponential random graph model.

In this work, we propose a Pseudoposterior-based procedure for selecting the interaction structure, in a large set of candidate subsets of a maximal structure, on a general class of Markov Random Field models with pairwise interactions based on a set of relative positions as introduced in section 3.2, using RJMCMC with a kernel specially constructed for this problem defined in section 3.3. In section 3.4 we use a simulation study to evaluate the proposed method under different scenarios and apply the algorithm in a texture synthesis problem with a real dataset in section 3.5.

3.2 Markov Random Fields with Spatially Homogeneous Pairwise Interactions

3.2.1 Model Description and Definitions

In this manuscript we consider the Markov Random Field (MRF) model on two-dimensional lattices with finite support and non-parametric pairwise interactions as described in Freguglia et al. (2020). The probability function for this model is completely defined by two main elements: *a set of relative positions*, that described the interaction structure of the process, and *a vector of potentials* describing the weights of interactions for each of these relative positions.

We denote by $\mathcal{S}$ a set of sites (also referred as pixels) in a finite n_1 by n_2 two-dimensional lattice

$$\mathcal{S} = \{\mathbf{i} = (i_1, i_2) : 1 \leqslant i_1 \leqslant n_1, 1 \leqslant i_2 \leqslant n_2\},$$

and $\mathbf{Z} = (Z_\mathbf{i})_{\mathbf{i}\in\mathcal{S}}$ a random field indexed by $\mathcal{S}$, where each $Z_\mathbf{i}$ is a random variable assuming values in a finite alphabet denoted $\mathcal{Z}$. Without loss of generality, we consider that that $\mathcal{Z} = \{0, 1, \dots, C\}$.

We define a *Relative Position Set (RPS)*, denoted $\mathcal{R}$, as a finite set of integer vectors $\mathbf{r} \in \mathbb{Z}^2$ without pairs of vectors with opposing directions, i.e.,

$$\mathbf{r} \in \mathcal{R} \implies -\mathbf{r} \notin \mathcal{R},$$

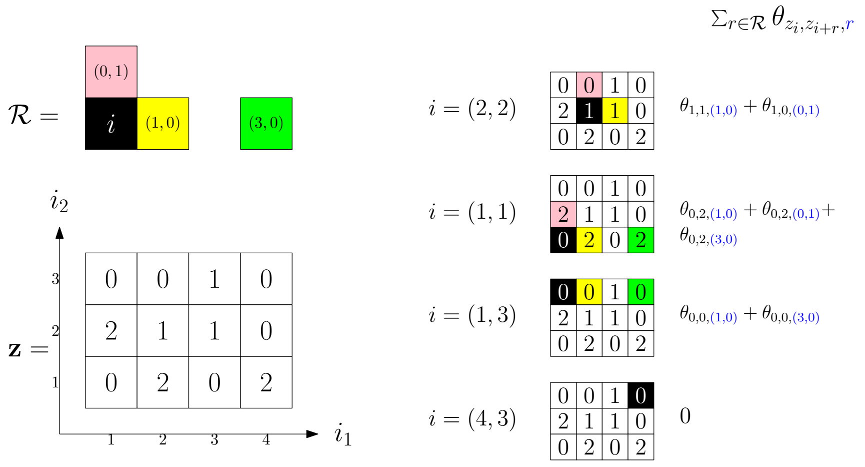
and, given a fixed RPS $\mathcal{R}$, we define a vector of potentials denoted $\boldsymbol{\theta}$, as a vector of real numbers indexed by $\mathcal{Z} \times \mathcal{Z} \times \mathcal{R}$,

$$\boldsymbol{\theta} = (\theta_{a,b,\mathbf{r}})_{a,b\in\mathcal{Z},\mathbf{r}\in\mathcal{R}}.$$

Given a RPS $\mathcal{R}$ and an associated vector $\boldsymbol{\theta}$, the Markov Random Field with homogeneous pairwise interaction considered in this work is characterized by the probability measure

$$f(\mathbf{z}|\mathcal{R}, \boldsymbol{\theta}) = \frac{1}{\zeta(\boldsymbol{\theta})} \exp\left(\sum_{\mathbf{i}\in\mathcal{S}} \sum_{\mathbf{r}\in\mathcal{R}} \sum_{a=0}^{C} \sum_{b=0}^{C} \theta_{a,b,\mathbf{r}} \mathbb{1}_{(z_\mathbf{i}=a)} \mathbb{1}_{(z_{\mathbf{i}+\mathbf{r}}=b)}\right), \tag{3.1}$$

where $\zeta(\boldsymbol{\theta}) = \sum_{\mathbf{z}'\in\mathcal{Z}^{|\mathcal{S}|}} \exp\left(\sum_{\mathbf{i}\in\mathcal{S}} \sum_{\mathbf{r}\in\mathcal{R}} \sum_{a=0}^{C} \sum_{b=0}^{C} \theta_{a,b,\mathbf{r}} \mathbb{1}_{(z'_\mathbf{i}=a)} \mathbb{1}_{(z'_{\mathbf{i}+\mathbf{r}}=b)}\right)$ is a normalizing constant, with the convention that the term $\mathbb{1}_{(z'_\mathbf{i}=a)}$ is treated as 0 if $\mathbf{i}' \notin \mathcal{S}$ for every $a \in \mathcal{Z}$. This ensures that the sum terms are consistently defined across every pair of positions in $\mathcal{S}$ that are within a relative position in $\mathcal{R}$. Figure 1 presents an illustration of how the terms $\sum_{\mathbf{r}\in\mathcal{R}} \theta_{z_{\mathbf{i}},z_{\mathbf{i}+\mathbf{r}},\mathbf{r}}$ are computed for some positions $\mathbf{i}$ of an example field $\mathbf{z}$.



$\sum_{r\in\mathcal{R}} \theta_{z_i,z_{i+r},r}$
$\mathcal{R} =$
$(0,1)$
i
$(1,0)$
$(3,0)$
i_2
$\mathbf{z} =$
i_1
$i = (2,2)$
$\theta_{1,1,(1,0)} + \theta_{1,0,(0,1)}$
$i = (1,1)$
$\theta_{0,2,(1,0)} + \theta_{0,2,(0,1)} + \theta_{0,2,(3,0)}$
$i = (1,3)$
$\theta_{0,0,(1,0)} + \theta_{0,0,(3,0)}$
$i = (4,3)$
0

Figure 1 – An example field $\mathbf{z}$ with dimensions $n_1 = 4$, $n_2 = 3$ and the computed sums $\sum_{\mathbf{r}\in\mathcal{R}} \theta_{z_{\mathbf{i}},z_{\mathbf{i}+\mathbf{r}},\mathbf{r}}$ for some positions $\mathbf{i}$ considering a RPS $\mathcal{R} = \{(1,0),(0,1),(3,0)\}$.

In (3.1), adding a constant value to the potentials $\theta_{a,b,\mathbf{r}}$ associated with every pair $a, b \in \mathcal{Z}$ and a fixed relative position $\mathbf{r}$, causes the value of $f(\mathbf{z},\mathcal{R},\boldsymbol{\theta})$ to be unchanged, as the resulting scale change is also reflected in the normalizing constant $\zeta(\boldsymbol{\theta})$. In other words, two different vector of potentials $\boldsymbol{\theta}$ may have the same likelihood, therefore, leading to an non-identifiability problem. In order to obtain identifiability, additional constraints are required and, following Freguglia et al. (2020), we adopt the zero-valued reference pair $((a,b) = (0,0))$ constraint

$$\theta_{0,0,\mathbf{r}} = 0 \text{ for all } \mathbf{r} \in \mathcal{R}.$$

Note that, while we still use the term $\theta_{0,0,\mathbf{r}}$ in some equations, for simplicity of notation, we will not consider these indexes in the vector $\boldsymbol{\theta}$. Additionally, the vector $\boldsymbol{\theta}$ can be expressed in terms of subvectors $\boldsymbol{\theta} = (\boldsymbol{\theta}_{\mathbf{r}})_{\mathbf{r}\in\mathcal{R}}$, where each subvector $\boldsymbol{\theta}_{\mathbf{r}} = (\theta_{a,b,\mathbf{r}})_{(a,b)\in\mathcal{Z}^2,(a,b)\neq(0,0)}$ corresponds to non-null potentials associated with a single relative position $\mathbf{r}$ and we denote by d the dimension of $\boldsymbol{\theta}_{\mathbf{r}}$, which is given by $d = (|\mathcal{Z}|)^2 - 1$.

3.2.2 Conditional probabilities and Pseudolikelihood

While the MRF model introduced is well-defined, inference for such model gets problematic on non-trivial cases due to the intractability of the normalizing constant, $\zeta(\boldsymbol{\theta})$, as it requires computing a sum of an exponential number of terms, $(|\mathcal{Z}|)^{n_1 n_2}$, which quickly becomes infeasible. For example, in practice, even for $n_1 = n_2 = 100$, which is not even considered large for common applications, computing the normalizing constant is impossible.

One of the most important features of MRF models is local dependence that makes probability functions decomposable into a product of functions that depend on $\mathbf{z}$ only through subsets of it, like pairs $(z_{\mathbf{i}}, z_{\mathbf{i}+\mathbf{r}})$, in the case of (3.1). This decomposition allows expressing the conditional probability of specific $z_{\mathbf{i}}$ given every other element $\mathbf{z}_{-\mathbf{i}} = \{z_{\mathbf{i}'} : \mathbf{i}' \in \mathcal{S}, \mathbf{i}' \neq \mathbf{i}\}$ as

$$f(z_{\mathbf{i}}|\mathbf{z}_{-\mathbf{i}}, \mathcal{R}, \boldsymbol{\theta}) = f(z_{\mathbf{i}}|\mathbf{z}_{\mathcal{N}_{\mathbf{i}}}, \mathcal{R}, \boldsymbol{\theta}) = \frac{\exp\left(\sum_{\mathbf{r}\in\mathcal{R}} \theta_{z_{\mathbf{i}}, z_{\mathbf{i}+\mathbf{r}}, \mathbf{r}} + \theta_{z_{\mathbf{i}-\mathbf{r}}, z_{i}, \mathbf{r}}\right)}{\sum_{a\in\mathcal{Z}} \exp\left(\sum_{\mathbf{r}\in\mathcal{R}} \theta_{a, z_{\mathbf{i}+\mathbf{r}}, \mathbf{r}} + \theta_{z_{\mathbf{i}-\mathbf{r}}, a, \mathbf{r}}\right)}, \tag{3.2}$$

where $\mathcal{N}_{\mathbf{i}} \subset \mathcal{S}$ denotes the set of neighbors of $\mathbf{i}$ based on the RPS $\mathcal{R}$, i.e., $\mathcal{N}_{\mathbf{i}} = \{\mathbf{i}' : \mathbf{i}' \in \mathcal{S} \text{ and } \mathbf{i}' = \mathbf{i} \pm \mathbf{r}, \mathbf{r} \in \mathcal{R}\}$.

The computationally simple expressions for conditional probabilities on (3.2) allows the use of alternative functions based on conditional probabilities instead of the joint probability. For problems with high-dimensional dependent data, when conditional probabilities are available and simple, a function widely used as a proxy for the likelihood function is the **Pseudolikelihood** function from Besag (1975), defined as the product of conditional probabilities evaluated at the observed values, $z_{\mathbf{i}}$,

$$\tilde{f}(\mathbf{z}|\mathcal{R}, \boldsymbol{\theta}) = \prod_{\mathbf{i}\in\mathcal{S}} f(z_{\mathbf{i}}|\mathbf{z}_{\mathcal{N}_{\mathbf{i}}}, \mathcal{R}, \boldsymbol{\theta}). \tag{3.3}$$

Note that while the normalizing constant $\zeta(\boldsymbol{\theta})$ from (3.1) requires a sum over $(|\mathcal{Z}|)^{|\mathcal{S}|}$ random field configurations, (3.3) involves $|\mathcal{S}|$ normalizing constants that are sums over $|\mathcal{Z}|$ terms. Thus, the computational cost for evaluating the Pseudolikelihood is $\mathcal{O}(|\mathcal{Z}| \times |\mathcal{S}|)$, while the exact likelihood function has a cost of order $\mathcal{O}\left(|\mathcal{Z}|^{|\mathcal{S}|}\right)$.

3.3 A Bayesian Framework for Sparse Interaction Structure Selection

In a Bayesian context, unobservable quantities, for example the parameters of a model, are considered unobserved random variables with specific prior distributions defined beforehand. Many Bayesian model selection methodologies extend this concept by assuming that not only a set of real-valued parameters (the vector of free potentials $\boldsymbol{\theta}$ within the scope of this work) is a vector of random variables, but also the model itself (interpreted as the RPS $\mathcal{R}$) is an unobserved random object with its given prior distribution.

Considering a collection of proper RPSs denoted $\mathcal{M}$, we can define a Bayesian system hierarchically by

$$\begin{aligned} \mathcal{R} &\sim q(\mathcal{R}), & \mathcal{R} &\in \mathcal{M}, \\ \boldsymbol{\theta}|\mathcal{R} &\sim \phi(\boldsymbol{\theta}|\mathcal{R}), & \boldsymbol{\theta} &\in \mathbb{R}^{d|\mathcal{R}|}, \\ \mathbf{z}|\mathcal{R},\boldsymbol{\theta} &\sim f(\mathbf{z}|\mathcal{R},\boldsymbol{\theta}), & \mathbf{z} &\in \mathcal{Z}^{|\mathcal{S}|}, \end{aligned}$$

where $q(\mathcal{R})$ is the prior distribution of the RPS, $\phi(\boldsymbol{\theta}|\mathcal{R})$ is the prior distribution of the parameter vector $\boldsymbol{\theta}$ given a particular RPS $\mathcal{R}$ and $f(\mathbf{z}|\mathcal{R},\boldsymbol{\theta})$ is the probability function of a MRF as in (3.1).

Given that the support of $\mathcal{R}$ represents the sets of interacting positions, a natural choice for the collection of candidate models $\mathcal{M}$ is the power set of a **maximal RPS**, denoted $\mathcal{R}_{\max}$,

$$\mathcal{M} = \{\mathcal{R}' : \mathcal{R}' \subset \mathcal{R}_{\max}\},$$

which contains $2^{|\mathcal{R}_{\max}|}$ possible neighborhoods. For simplicity of notation, we shall use $\boldsymbol{\theta}$ to denote a vector of varying dimension, which indexing is always associated with an interaction structure $\mathcal{R}$. The dimension, $d|\mathcal{R}|$, and indexing of $\boldsymbol{\theta} = (\boldsymbol{\theta}_{\mathbf{r}})_{\mathbf{r}\in\mathcal{R}}$ are always implicitly specified as the vector is consistently matched with an interaction structure $\mathcal{R}$ in every expression. Note that, within this scope, we are referring as a model to the RPS that defines the interaction structure of a MRF. This problem can also be interpreted as a variable selection problem as any vector of interaction coefficients associated with a RPS $\mathcal{R}$, with restrictions that $\boldsymbol{\theta}_{a,b,\mathbf{r}} = 0$ for all a, b for specific $\mathbf{r}$, can also be expressed (in terms of identical likelihood values) to a model excluding $\mathbf{r}$ from $\mathcal{R}$.

In a model selection context, our main interest is to find the marginal posterior distribution of a model $\pi(\mathcal{R}|\mathbf{z})$, which can be obtained by integrating the (complete) posterior distribution,

$$\pi(\mathcal{R},\boldsymbol{\theta}|\mathbf{z}) = \frac{q(\mathcal{R})\phi(\boldsymbol{\theta}|\mathcal{R})f(\mathbf{z}|\mathcal{R},\boldsymbol{\theta}))}{\sum_{\mathcal{R}'\in\mathcal{M}} q(\mathcal{R}')\int_{\mathbb{R}^{d|\mathcal{R}'|}} \phi(\boldsymbol{\theta}'|\mathcal{R}')f(\mathbf{z}|\mathcal{R}',\boldsymbol{\theta}')d\boldsymbol{\theta}'}, \tag{3.4}$$

with respect to $\boldsymbol{\theta}$.

Two main computational challenges arise from (3.4) making most direct analyses prohibitively complex: (I) $f(\mathbf{z}|\mathcal{R},\boldsymbol{\theta}))$ cannot be evaluated directly due to the intractable normalizing constant and (II) the denominator involves $2^{|\mathcal{M}|}$ integrations, possibly including many high-dimensional functions that have intractable normalizing constants. Because of these two sources of intractability, this type of posterior distribution is often referred in the literature as a doubly-intractable distribution (Murray et al., 2012; Caimo and Mira, 2015).

Monte Carlo Markov Chain methods are used to generate an ergodic Markov Chain which invariant distribution is equal to a specific target distribution which, in most cases, is a posterior distribution with intractable normalizing constant like (3.4).

Consider an ergodic Markov chain in the space that is a product of $\mathcal{M}$ by the space of real vectors with varying dimension directly associated with the element of $\mathcal{M}$, i.e., $(\mathcal{R}^{(1)}, \boldsymbol{\theta}^{(1)}), (\mathcal{R}^{(2)}, \boldsymbol{\theta}^{(2)}), \ldots$, such that $\boldsymbol{\theta}^{(t)} \in \mathbb{R}^{d|\mathcal{R}^{(t)}|}$, and invariant measure $\pi(\cdot, \cdot|\mathbf{z})$. Then, due to the ergodic theorem, for any bounded function g of the form

$$g : \mathcal{M} \times \bigcup_{k=0}^{|\mathcal{R}_{\max}|} \mathbb{R}^{dk} \to \mathbb{R},$$

we have

$$\sum_{t=1}^{n} g(\mathcal{R}^{(t)}, \boldsymbol{\theta}^{(t)}) \to \mathbb{E}_{\pi}\left(g(\mathcal{R}, \boldsymbol{\theta})|\mathbf{z}\right), \quad \text{a.s.} \tag{3.5}$$

where $\mathbb{E}_{\pi}\left(g(\mathcal{R}, \boldsymbol{\theta})|\mathbf{z}\right)$ is the conditional expected value of the random variable $g(\mathcal{R}, \boldsymbol{\theta})$ given the observed $\mathbf{z}$, under the (target) distribution $\pi(\mathcal{R}, \boldsymbol{\theta}|\mathbf{z})$. Some particular choices of g lead to interpretable quantities that are useful for evaluating the plausibility of interaction neighborhoods $\mathcal{R}$ based on their posterior distribution, such as $g(\mathcal{R}, \boldsymbol{\theta}) = \mathbb{1}(\mathcal{R} = \mathcal{R}^*)$, which results in (3.5) being the posterior probability of a particular neighborhood $\mathcal{R}^*$ or $g(\mathcal{R}, \boldsymbol{\theta}) = \mathbb{1}(\mathbf{r} \in \mathcal{R})$, which corresponds to the marginal posterior probability that a particular relative position $\mathbf{r}$ belongs to the RPS.

Given the estimated marginal posterior probabilities for each position in $\mathcal{R}_{\max}$, obtained from a Metropolis-Hastings sample of size T, and a threshold value c_{th}, a sparse estimator of the RPS, denoted $\hat{\mathcal{R}}_{\text{sp}}(c_{\text{th}})$, can be obtained by selecting the set of all positions with (estimated) posterior probability exceeding c_{th},

$$\hat{\mathcal{R}}_{\text{sp}}(c_{\text{th}}) = \{\mathbf{r} \in \mathcal{R}_{\max} : \frac{1}{T}\sum_{t=1}^{T} \mathbb{1}\left(\mathbf{r} \in \mathcal{R}^{(t)}\right) > c_{\text{th}}\}. \tag{3.6}$$

3.3.1 Pseudoposterior-based inference

In order to overcome the computational infeasibility due to the intractable normalizing constant of the likelihood function f defined in (3.1), many methods for inference on MRFs have been proposed. One of the commonly used approaches is to replace the likelihood function, f, for the pseudolikelihood, $\tilde{f}$, defined in (3.3), which can be evaluated directly.

When applied to the Bayesian system defined in the previous section, this replacement of the likelihood function leads to an alternative function referred as Pseudoposterior distribution, that is proportional to the product of prior distributions and the pseudolikelihood, and it is formally defined as (*cf.* with (3.4))

$$\tilde{\pi}(\mathcal{R}, \boldsymbol{\theta}|\mathbf{z}) = \frac{q(\mathcal{R})\phi(\boldsymbol{\theta}|\mathcal{R})\tilde{f}(\mathbf{z}|\mathcal{R}, \boldsymbol{\theta}))}{\sum_{\mathcal{R}' \in \mathcal{M}} q(\mathcal{R}') \int_{\mathbb{R}^{d|\mathcal{R}'|}} \phi(\boldsymbol{\theta}'|\mathcal{R}')\tilde{f}(\mathbf{z}|\mathcal{R}', \boldsymbol{\theta}')d\boldsymbol{\theta}'}. \tag{3.7}$$

It is valuable to note that, while the pseudolikelihood is a plausible proxy for the likelihood function in terms of optimization-related mathematical properties, these functions may have different overall shapes depending on how much dependence exists on the dataset considered, and composing functions using the pseudolikelihood instead of the likelihood alters how these functions are interpreted. As a consequence, Bayesian inference based on pseudoposterior produces useful quantities, but the information obtained cannot be interpreted in the conventional way. For example, integrating the pseudoposterior distribution over $\boldsymbol{\theta}$ does not result exactly in the posterior distribution of the RPSs $\mathcal{R}$, but a dfferent measure that conceivably be useful for evaluating the plausibility of the RPSs.

3.3.2 A Reversible Jump Proposal Kernel for Sparse Neighborhood Detection

Constructing a proposal kernel for the Metropolis-Hastings algorithm that can efficiently move through both the model space $\mathcal{M}$ and the space of interaction coefficients within a model is not a simple task. Green (1995) proposes the Reversible Jump Monte Carlo Markov Chain (RJMCMC) as a framework for Bayesian analysis of models and varying dimension parameters simultaneously. In general, the strategy consists of composing a proposal kernel which is a mixture of simpler kernels, some proposing within-model moves that only changes parameter values and others proposing reversible jumps between models that have good analytical or computational properties.

In this work, we construct a customized proposal kernel for the model inspired on properties and examples from Brooks et al. (2003) with additional features that are characteristics of the neighborhood selection for MRFs. This proposal kernel consists of a mixture of 4 types of moves described next, where each attempts to come up with states that may have higher pseudoposterior density then the current state with some probability.

Within-model random walk move

The first and simplest move consists of adding a random walk term to the current value of the parameter vector. Given a current pair $(\mathcal{R}, \boldsymbol{\theta})$, we keep the same neighborhood, $\mathcal{R}' = \mathcal{R}$, and propose a new vector of interaction coefficients $\boldsymbol{\theta}' \in \mathcal{Z}^{d|\mathcal{R}|}$ by adding a Gaussian noise term with matching dimension, where each coordinate is independent and identically distributed with mean 0 and variance σ_w^2, where σ_w^2 is a tuning parameter of the algorithm.

The transition kernel density for this move is given by

$$\kappa_w(\boldsymbol{\theta}', \mathcal{R}'|\boldsymbol{\theta}, \mathcal{R}) = \frac{1}{(2\pi\sigma_w^2)^{d|\mathcal{R}|/2}} \exp\left(-\frac{1}{2\sigma_w^2}\sum_{\mathbf{r}\in\mathcal{R}}(\boldsymbol{\theta}_{\mathbf{r}} - \boldsymbol{\theta}'_{\mathbf{r}})^\top(\boldsymbol{\theta}_r - \boldsymbol{\theta}'_r)\right)\mathbb{1}(\mathcal{R}' = \mathcal{R}),$$

what makes this proposal density not only reversible, but also symmetrical, i.e., $\kappa_w(\boldsymbol{\theta}', \mathcal{R}'|\boldsymbol{\theta}, \mathcal{R}) = \kappa_w(\boldsymbol{\theta}, \mathcal{R}|\boldsymbol{\theta}', \mathcal{R}')$. Since proposed states keep the same interaction structure, we also have

$q(\mathcal{R}) = q(\mathcal{R}')$, so the terms corresponding to the neighborhood interaction structure are also cancelled in acceptance ratios.

While this move does not contribute to jumping between RPSs, its goal is to add small incremental changes in the parameters value so that the chain gradually moves towards higher pseudoposterior density regions within a model. Typically, small values of σ_w^2 values are preferred so that the coefficients within a RPS are slowly drifting towards the maximum pseudoposterior vector for that RPS.

Birth and Death

We propose a jump move from a state $(\mathcal{R}, \boldsymbol{\theta})$ to a state $(\mathcal{R}', \boldsymbol{\theta}')$, $\mathcal{R}' \neq \mathcal{R}$, by either including a position from $\mathcal{R}_{\max}$ that is not already in $\mathcal{R}$, or by removing one of the positions in $\mathcal{R}$. We refer to these moves as Birth and Death of a relative position, respectively.

We define the RPS comparison operator $\overset{\mathbf{r}}{\prec}$ as

$$\mathcal{R} \overset{\mathbf{r}}{\prec} \mathcal{R}' \iff \mathcal{R} \subset \mathcal{R}', \mathbf{r} \notin \mathcal{R} \text{ and } \mathcal{R} \cup \{\mathbf{r}\} = \mathcal{R}',$$

which means that $\mathcal{R}'$ can be obtained by adding the position $\mathbf{r}$ to $\mathcal{R}$. Given a current RPS $\mathcal{R}$, we randomly select a position $\mathbf{r}^*$ from $\mathcal{R}_{\max}$ with uniform probabilities $\frac{1}{|\mathcal{R}_{\max}|}$. Then either a birth or death move is proposed depending on the selected $\mathbf{r}^*$.

- If $\mathbf{r}^* \in \mathcal{R}$, the proposed RPS $\mathcal{R}'$ is such that $\mathcal{R}' \overset{\mathbf{r}^*}{\prec} \mathcal{R}$, i.e., $\mathbf{r}^*$ is removed from $\mathcal{R}$. For the associated interaction coefficients $\boldsymbol{\theta}'$ to be proposed with $\mathcal{R}'$, all the values are kept the same $\boldsymbol{\theta}'_{\mathbf{r}} = \boldsymbol{\theta}_{\mathbf{r}}$ for $\mathbf{r} \in \mathcal{R}'$.
- If $\mathbf{r}^* \notin \mathcal{R}$, $\mathcal{R}'$ is proposed by including $\mathbf{r}^*$, i.e., $\mathcal{R} \overset{\mathbf{r}^*}{\prec} \mathcal{R}'$. For the proposed parameter $\boldsymbol{\theta}'$, we keep the values of the previous $\boldsymbol{\theta}'_{\mathbf{r}} = \boldsymbol{\theta}_{\mathbf{r}}$ for the previously included positions $\mathbf{r} \in \mathcal{R}$ and sample a new vector of i.i.d. Gaussian variables with mean 0 and variance σ^2_{bd} to assign to $\boldsymbol{\theta}'_{\mathbf{r}^*}$.

Note that, by this definition, transitions between states with two different RPSs $\mathcal{R}$ and $\mathcal{R}'$ are allowed if, and only if, $|\mathcal{R} \triangledown \mathcal{R}'| = 1$, where $\triangledown$ denotes the symmetrical difference operator for two sets. Therefore, the proposal kernel density for a birth/death jump move is given by

$$\kappa_{\text{bd}}(\boldsymbol{\theta}', \mathcal{R}' | \boldsymbol{\theta}, \mathcal{R}) = \begin{cases} \frac{1}{|\mathcal{R}_{\max}|} \frac{\exp\left(-\frac{1}{2\sigma^2_{\text{bd}}} {\boldsymbol{\theta}'}^{\top}_{\mathbf{r}^*} {\boldsymbol{\theta}'}_{\mathbf{r}^*}\right)}{\left(2\pi\sigma^2_{\text{bd}}\right)^{d/2}} \prod_{\mathbf{r} \in \mathcal{R}} \mathbb{1}(\boldsymbol{\theta}'_{\mathbf{r}} = \boldsymbol{\theta}_{\mathbf{r}}) & , \text{ if } \mathcal{R} \overset{\mathbf{r}^*}{\prec} \mathcal{R}', \\ \frac{1}{|\mathcal{R}_{\max}|} \prod_{\mathbf{r} \in \mathcal{R}'} \mathbb{1}(\boldsymbol{\theta}'_{\mathbf{r}} = \boldsymbol{\theta}_{\mathbf{r}}) & , \text{ if } \mathcal{R}' \overset{\mathbf{r}^*}{\prec} \mathcal{R}, \\ 0 & , \text{ if } |\mathcal{R} \triangledown \mathcal{R}'| \neq 1. \end{cases}$$

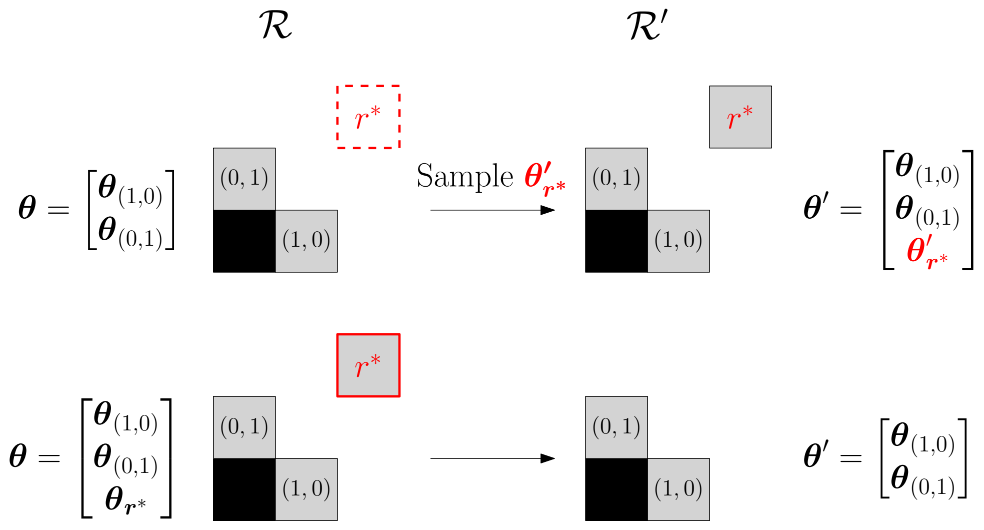

$\mathcal{R}$
$\mathcal{R}'$
r^*
(0, 1)
(1, 0)
Sample $\boldsymbol{\theta}'_{r^*}$
$\boldsymbol{\theta} = \begin{bmatrix}\boldsymbol{\theta}_{(1,0)} \\ \boldsymbol{\theta}_{(0,1)}\end{bmatrix}$
$\boldsymbol{\theta}' = \begin{bmatrix}\boldsymbol{\theta}_{(1,0)} \\ \boldsymbol{\theta}_{(0,1)} \\ \boldsymbol{\theta}'_{r^*}\end{bmatrix}$
$\boldsymbol{\theta} = \begin{bmatrix}\boldsymbol{\theta}_{(1,0)} \\ \boldsymbol{\theta}_{(0,1)} \\ \boldsymbol{\theta}_{r^*}\end{bmatrix}$
$\boldsymbol{\theta}' = \begin{bmatrix}\boldsymbol{\theta}_{(1,0)} \\ \boldsymbol{\theta}_{(0,1)}\end{bmatrix}$

Figure 2 – Illustration of a Birth/Death Jump proposal when sampling $\mathbf{r}^* \notin \mathcal{R}$ (top) and $\mathbf{r}^* \in \mathcal{R}$ (bottom).

Figure 2 illustrates how new states $(\mathcal{R}', \boldsymbol{\theta}')$ are proposed from a current $(\mathcal{R}, \boldsymbol{\theta})$ when a randomly selected position $\mathbf{r}^*$ is included or not included in $\mathcal{R}$. It is straightforward to conclude from the example that this type of jump can be reversed by selecting the same position $\mathbf{r}^*$ and, the case of adding a new position, sampling the appropriate $\boldsymbol{\theta}'_{\mathbf{r}^*}$, therefore, $\kappa_{\text{bd}}(\boldsymbol{\theta}', \mathcal{R}' | \boldsymbol{\theta}, \mathcal{R}) > 0$ if, and only if, $\kappa_{\text{bd}}(\boldsymbol{\theta}, \mathcal{R} | \boldsymbol{\theta}', \mathcal{R}') > 0$.

Position swap move

As second type of move constructed for proposing jumps between states with different RPS is to swap one of positions in the current RPS, $\mathbf{r}_{\text{in}} \in \mathcal{R}$, for another one $\mathbf{r}_{\text{out}} \in \mathcal{R}_{\max} \backslash \mathcal{R}$, while keeping all of the interaction coefficients the same, including the subvector associated with the swapped position, $\boldsymbol{\theta}'_{\mathbf{r}_{\text{out}}} = \boldsymbol{\theta}_{\mathbf{r}_{\text{in}}}$. Figure 3 illustrates an example of **Position swap move** proposal.

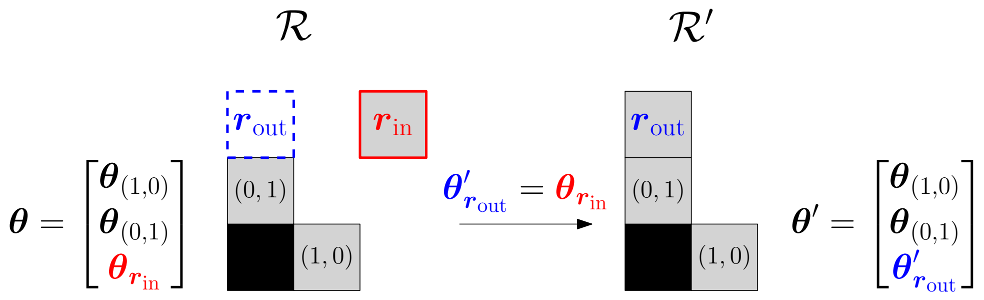

$\mathcal{R}$
$\mathcal{R}'$
r_{out}
r_{in}
(0, 1)
(1, 0)
$\boldsymbol{\theta}'_{r_{\text{out}}} = \boldsymbol{\theta}_{r_{\text{in}}}$
$\boldsymbol{\theta} = \begin{bmatrix}\boldsymbol{\theta}_{(1,0)} \\ \boldsymbol{\theta}_{(0,1)} \\ \boldsymbol{\theta}_{r_{\text{in}}}\end{bmatrix}$
$\boldsymbol{\theta}' = \begin{bmatrix}\boldsymbol{\theta}_{(1,0)} \\ \boldsymbol{\theta}_{(0,1)} \\ \boldsymbol{\theta}'_{r_{\text{out}}}\end{bmatrix}$

Figure 3 – Illustration of a Position swap move proposal.

The positions $\mathbf{r}_{\text{in}}$ and $\mathbf{r}_{\text{out}}$, for the swap move, are chosen independently and uniformly distributed on $\mathcal{R}$ and $\mathcal{R}_{\max} \backslash \mathcal{R}$, respectively. Therefore, for any states $(\boldsymbol{\theta}, \mathcal{R})$ and $(\boldsymbol{\theta}', \mathcal{R}')$ such that $|\mathcal{R}| = |\mathcal{R}'|$ and $|\mathcal{R} \nabla \mathcal{R}'| = 2$, differing only in the presence of relative positions $\mathbf{r}_{\text{in}} \in \mathcal{R}$ and $\mathbf{r}_{\text{out}} \in \mathcal{R}'$, the proposal density for the swap move is given by

$$\kappa_{\text{sw}}(\boldsymbol{\theta}', \mathcal{R}'|\boldsymbol{\theta}, \mathcal{R}) = \frac{1}{|\mathcal{R}||\mathcal{R}_{\text{max}} \backslash \mathcal{R}|} \prod_{\mathbf{r} \in \mathcal{R} \cap \mathcal{R}'} \mathbb{1}(\boldsymbol{\theta}_{\mathbf{r}} = \boldsymbol{\theta}'_{\mathbf{r}}) \mathbb{1}(\boldsymbol{\theta}_{\mathbf{r}_{\text{in}}} = \boldsymbol{\theta}'_{\mathbf{r}_{\text{out}}}). \tag{3.8}$$

Note that (3.8) is a symmetrical kernel since the inverse operation is proposed by selecting the same pair of positions $\mathbf{r}_{\text{in}}$ and $\mathbf{r}_{\text{out}}$ reversed, the RPS prior probability only depends on the size of the current RPS, $|\mathcal{R}|$, and we have the condition that $|\mathcal{R}| = |\mathcal{R}'|$ for every pair of states with positive proposal probability.

The rationale behind this move is that, due to the spatial dependence intrinsic to lattice-based indexing of the MRF model considered, the counts of pairwise configurations in some relative positions may present high correlation, especially when those relative positions are close (e.g. $\mathbf{r}$ and $\mathbf{r} + (1, 0)$) or a multiple one from another (e.g. $\mathbf{r}$ and $2\mathbf{r}$).

Note that while the same jumps proposed by swap moves could be achieved by a series of birth and death moves, one of those steps would be to exclude a relative position, $\mathbf{r}_{\text{in}}$, with associated interaction coefficient, $\boldsymbol{\theta}_{\mathbf{r}_{\text{in}}}$, possibly far from the zero vector, what would cause the acceptance of such move to be highly unlikely. Thus, swap moves is a proposal step that avoids the algorithm getting stuck at a local (with respect to RPSs) maxima, by adding direct connections to states with different RPSs that may have similar pseudoposterior values, taking into account very specific characteristics of the model.

Split and Merge

While the Position Swap Move is proposed in order to allow a relative position included in a state to be substituted by another one that is not included but has a similar pseudoposterior value, another type of local maxima may exist when two or more relative positions with highly correlated sufficient statistics are included in a model simultaneously.

The correlation between vectors of sufficient statistics may cause interaction weights for some relative positions $\boldsymbol{\theta}_{\mathbf{r}}$ to become very unstable, due to the possibility of compensating shifts in one direction for one of the vectors with equivalent shifts in the opposite direction.

We define the **Split and Merge** moves as a pair of reversible operations that not only allow jumps between different RPSs but also control the values of $\boldsymbol{\theta}_{\mathbf{r}}$ involved. This transition has the goal of redistributing the interactions coefficients, allowing smaller or larger RPSs with similar likelihood to be proposed with some probability. The Merge move permits excessive relative positions to be removed from the current state and possibly generates a proposed state that distributes the interaction weights $\boldsymbol{\theta}_{\mathbf{r}}$ of the position to be removed, by adding it to the remaining positions, hopefully keeping the pseudolikelihood values on similar levels. The key premise on this pair of moves is that, for a pair of states

$(\mathcal{R}, \boldsymbol{\theta})$ and $(\mathcal{R}', \boldsymbol{\theta}')$, we have

$$\sum_{\mathbf{r}\in\mathcal{R}} \boldsymbol{\theta}_{\mathbf{r}} = \sum_{\mathbf{r}\in\mathcal{R}'} \boldsymbol{\theta}'_{\mathbf{r}'}.$$

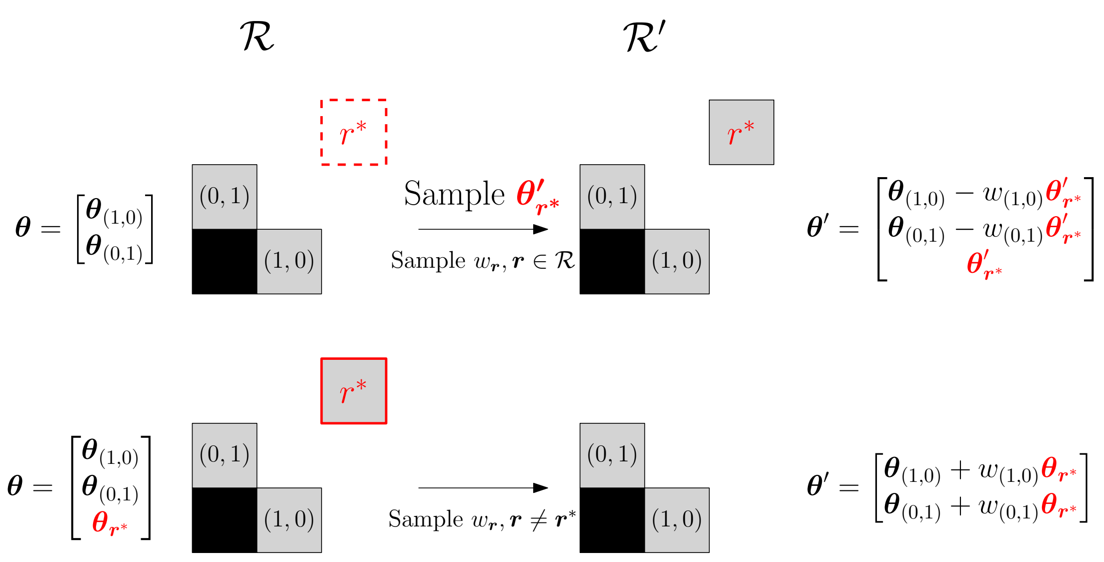

$\mathcal{R}$
$\mathcal{R}'$
r^*
r^*
$\boldsymbol{\theta} = \begin{bmatrix}\boldsymbol{\theta}_{(1,0)}\\ \boldsymbol{\theta}_{(0,1)}\end{bmatrix}$
(0, 1)
(1, 0)
Sample $\boldsymbol{\theta}'_{r^*}$
Sample $w_r, r \in \mathcal{R}$
(0, 1)
(1, 0)
$\boldsymbol{\theta}' = \begin{bmatrix}\boldsymbol{\theta}_{(1,0)} - w_{(1,0)}\boldsymbol{\theta}'_{r^*}\\ \boldsymbol{\theta}_{(0,1)} - w_{(0,1)}\boldsymbol{\theta}'_{r^*}\\ \boldsymbol{\theta}'_{r^*}\end{bmatrix}$
r^*
$\boldsymbol{\theta} = \begin{bmatrix}\boldsymbol{\theta}_{(1,0)}\\ \boldsymbol{\theta}_{(0,1)}\\ \boldsymbol{\theta}_{r^*}\end{bmatrix}$
(0, 1)
(1, 0)
Sample $w_r, r \neq r^*$
(0, 1)
(1, 0)
$\boldsymbol{\theta}' = \begin{bmatrix}\boldsymbol{\theta}_{(1,0)} + w_{(1,0)}\boldsymbol{\theta}_{r^*}\\ \boldsymbol{\theta}_{(0,1)} + w_{(0,1)}\boldsymbol{\theta}_{r^*}\end{bmatrix}$

Figure 4 – Illustration of Split (top) and Merge (bottom) moves.

Given a current state $(\mathcal{R}, \boldsymbol{\theta})$, the process of proposing a state $(\mathcal{R}', \boldsymbol{\theta}')$ with a **Split** move is composed by the steps of

1. Sample a new position $\mathbf{r}^*$ to be included from $\mathcal{R}_{\max}\backslash\mathcal{R}$ with uniform probability.
2. Generate a new interaction coefficient vector $\boldsymbol{\theta}'_{\mathbf{r}*}$ from a d-dimensional independent Gaussian distribution with variances σ_s^2, where the split variance, σ_s^2, is a tuning parameter of the algorithm.
3. Generate a vector of weights $\boldsymbol{w} = (w_{\mathbf{r}})_{\mathbf{r}\in\mathcal{R}}$ from a symmetric Dirichlet distribution with all parameters equal to ν, where ν is another tuning parameter of the algorithm.
4. Propose $\mathcal{R}' = \mathcal{R} \cup \mathbf{r}^*$ and $\boldsymbol{\theta}'$ such that $\boldsymbol{\theta}'_{\mathbf{r}*}$ is the generated vector and $\boldsymbol{\theta}'_{\mathbf{r}} = \boldsymbol{\theta}_{\mathbf{r}} - w_{\mathbf{r}}\boldsymbol{\theta}'_{\mathbf{r}*}$ for every other relative position $\mathbf{r} \in \mathcal{R}$.

The proposal density for a Split move, κ_{s}, is given by

$$\kappa_{\mathrm{s}}(\boldsymbol{\theta}', \mathcal{R}'|\boldsymbol{\theta}, \mathcal{R}) = \frac{\prod_{\mathbf{r}^*\in\mathcal{R}_{\max}\backslash\mathcal{R}} \mathbb{1}(\mathcal{R} \overset{\mathbf{r}^*}{<} \mathcal{R}')}{|\mathcal{R}_{\max}\backslash\mathcal{R}|} \frac{\exp\left(\frac{-(\boldsymbol{\theta}'_{\mathbf{r}*}{}^{\top}\boldsymbol{\theta}'_{\mathbf{r}*})}{2\sigma_s^2}\right)}{(2\pi\sigma_s^2)^{d/2}}$$

$$\frac{\Gamma\left(|\mathcal{R}|\nu\right)}{\prod_{\mathbf{r}\in\mathcal{R}}\left(\Gamma(\nu)\right)} \prod_{\mathbf{r}\in\mathcal{R}} \left[\left(\frac{\theta_{0,1,\mathbf{r}} - \theta'_{0,1,\mathbf{r}}}{\theta'_{0,1,\mathbf{r}*}}\right)^{\nu-1} \times \frac{\mathbb{1}\left(0 < \frac{\theta_{0,1,\mathbf{r}} - \theta'_{0,1,\mathbf{r}}}{\theta'_{0,1,\mathbf{r}*}} < 1\right)}{\theta'_{0,1,\mathbf{r}*}} \right]$$

$$\mathbb{1}\left(\sum_{\mathbf{r}\in\mathcal{R}} (\boldsymbol{\theta}_{\mathbf{r}} - \boldsymbol{\theta}'_{\mathbf{r}}) = \boldsymbol{\theta}'_{\mathbf{r}*}\right) \prod_{a,b\in\mathcal{Z}^2} \mathbb{1}\left(\frac{\theta_{a,b,\mathbf{r}} - \theta'_{a,b,\mathbf{r}}}{\theta'_{a,b,\mathbf{r}*}} = \frac{\theta_{0,1,\mathbf{r}} - \theta'_{0,1,\mathbf{r}}}{\theta'_{0,1,\mathbf{r}*}}\right).$$

The **Merge** proposal move is the reverse of the Split move and can be described by the following sequence of steps

1. Sample a state $\mathbf{r}^*$ from $\mathcal{R}$, which interaction coefficient vector will be merged into the others, with uniform probabilities.
2. Generate a vector of weights $\boldsymbol{w} = (w_{\mathbf{r}})_{\mathbf{r}\in(\mathcal{R}\backslash\mathbf{r}^*)}$ with Dirichlet distribution with all parameters equal to ν.
3. Propose $\mathcal{R}' = \mathcal{R}\backslash\mathbf{r}^*$ and $\boldsymbol{\theta}'$ such that $\boldsymbol{\theta}'_{\mathbf{r}} = \boldsymbol{\theta}_{\mathbf{r}} + w_{\mathbf{r}}\boldsymbol{\theta}_{\mathbf{r}^*}$ for each $\mathbf{r} \in \mathcal{R}'$.

The proposal density for a Merge move is

$$\begin{aligned}\kappa_{\mathrm{m}}(\boldsymbol{\theta}', \mathcal{R}'|\boldsymbol{\theta}, \mathcal{R}) =& \frac{\prod_{\mathbf{r}^*\in\mathcal{R}} \mathbb{1}(\mathcal{R}' \overset{\mathbf{r}^*}{\prec} \mathcal{R})}{|\mathcal{R}|} \\ & \frac{\Gamma\left(|\mathcal{R}'|\nu\right)}{\prod_{\mathbf{r}\in\mathcal{R}'} (\Gamma(\nu))} \prod_{\mathbf{r}\in\mathcal{R}'} \left[\left(\frac{\theta'_{0,1,\mathbf{r}} - \theta_{0,1,\mathbf{r}}}{\theta_{0,1,\mathbf{r}^*}}\right)^{\nu-1} \times \frac{\mathbb{1}\left(0 < \frac{\theta'_{0,1,\mathbf{r}} - \theta_{0,1,\mathbf{r}}}{\theta_{0,1,\mathbf{r}^*}} < 1\right)}{\theta_{0,1,\mathbf{r}^*}} \right] \\ & \mathbb{1}\left(\sum_{\mathbf{r}\in\mathcal{R}'} \boldsymbol{\theta}'_{\mathbf{r}} - \boldsymbol{\theta}_{\mathbf{r}} = \boldsymbol{\theta}_{\mathbf{r}^*}\right) \prod_{a,b\in\mathcal{Z}^2} \mathbb{1}\left(\frac{\theta_{a,b,\mathbf{r}} - \theta'_{a,b,\mathbf{r}}}{\theta'_{a,b,\mathbf{r}^*}} = \frac{\theta_{0,1,\mathbf{r}} - \theta'_{0,1,\mathbf{r}}}{\theta'_{0,1,\mathbf{r}^*}}\right).\end{aligned}$$

It is easy to see that an accepted Split move is reversed by a Merge move if the sampled relative position $\mathbf{r}^*$ and the generated vector of weights $\boldsymbol{w}$ is the same for both moves. This fact also produces an analytical simplification in the ratio of proposal densities, required for computing the acceptance ratio in the Metropolis-Hastings algorithm, for these moves as

$$\frac{\kappa_{\mathrm{m}}(\boldsymbol{\theta}, \mathcal{R}|\boldsymbol{\theta}', \mathcal{R}')}{\kappa_{\mathrm{s}}(\boldsymbol{\theta}', \mathcal{R}'|\boldsymbol{\theta}, \mathcal{R})} = \frac{|\mathcal{R}_{\max}\backslash\mathcal{R}|}{|\mathcal{R}'|} \frac{\exp\left(\frac{-(\boldsymbol{\theta}'^{\top}_{\mathbf{r}^*}\boldsymbol{\theta}'_{\mathbf{r}^*})}{2\sigma_s^2}\right)}{(2\pi\sigma_s^2)^{d/2}}, \tag{3.9}$$

for any pair of states $(\mathcal{R}, \boldsymbol{\theta})$ and $(\mathcal{R}', \boldsymbol{\theta}')$ such that $\kappa_{\mathrm{s}}(\boldsymbol{\theta}', \mathcal{R}'|\boldsymbol{\theta}, \mathcal{R}) > 0$. The ratio of proposal densities for the reversed transitions can be achieved by applying the inverse of (3.9).

Mixture proposal density

Each move described previously has a different goal in terms of how a region (in terms of both RPS and interaction coefficients) of much higher posterior density could be proposed with some probability improving the rate of convergence of the Metropolis-Hastings algorithm to a region corresponding to a global maximum. In order to assemble the five groups of moves into a single transition kernel, we define a proposal density κ that is composed by a mixture of the of five described densities

$$\kappa(\boldsymbol{\theta}', \mathcal{R}'|\boldsymbol{\theta}, \mathcal{R}) = \sum_{\Psi\in\{\mathrm{w},\mathrm{bd},\mathrm{sw},\mathrm{m},\mathrm{s}\}} p_{\Psi}(\mathcal{R})\kappa_{\Psi}(\boldsymbol{\theta}', \mathcal{R}'|\boldsymbol{\theta}, \mathcal{R}), \tag{3.10}$$

where $p_\Psi(\mathcal{R})$ corresponds to the probability of selecting a move Ψ when the current state has the RPS component as $\mathcal{R}$ and $\sum_\Psi p_\Psi(\mathcal{R}) = 1$. Having the mixture probabilities depend on the current RPS is required to avoid undefined behaviors such as proposing a random walk move κ_{w} when we have an empty RPS, $\mathcal{R} = \varnothing$.

Following Green (1995) and Brooks et al. (2003), we can describe the acceptance of the Metropolis-Hastings (Reversible Jump) algorithm when using a mixture proposal density by using the ratio of the sampled move Ψ at each step,

$$\mathcal{A}_\Psi(\boldsymbol{\theta}', \mathcal{R}' | \boldsymbol{\theta}, \mathcal{R}) = \frac{q(\mathcal{R}')}{q(\mathcal{R})} \frac{\pi(\boldsymbol{\theta}'|\mathcal{R}')}{\pi(\boldsymbol{\theta}|\mathcal{R})} \frac{\tilde{f}(\boldsymbol{\theta}', \mathcal{R}')}{\tilde{f}(\boldsymbol{\theta}, \mathcal{R})} \frac{p_{\Psi'}(\mathcal{R})}{p_\Psi(\mathcal{R})} \frac{\kappa_{\Psi'}(\boldsymbol{\theta}, \mathcal{R}|\boldsymbol{\theta}', \mathcal{R}')}{\kappa_\Psi(\boldsymbol{\theta}', \mathcal{R}'|\boldsymbol{\theta}, \mathcal{R})}, \tag{3.11}$$

where Ψ' is the inverse move of the move Ψ, i.e., $\Psi' = \Psi$ if $\Psi =$ w, bd or sw and swapped for $\Psi =$ s or m. The complete procedure is described in Algorithm 1. In practice, computations involving $\mathcal{A}_\Psi(\cdot, \cdot|\cdot, \cdot)$ can be performed in logarithmic scale for both analytical and numerical simplicity.

Algorithm 1 – Metropolis-Hastings algorithm with mixture proposal density.

```
Set the initial state $(\mathcal{R}^{(0)}, \boldsymbol{\theta}^{(0)})$;
**foreach** $t = 0, ..., n_{iter}$ **do**
    Sample a random move $\Psi$ from $\{$w, bd, sw, s, m$\}$ with probabilities
      $p_{\text{w}}(\mathcal{R}^{(t)}), p_{\text{bd}}(\mathcal{R}^{(t)}), p_{\text{sw}}(\mathcal{R}^{(t)}), p_{\text{s}}(\mathcal{R}^{(t)}), p_{\text{m}}(\mathcal{R}^{(t)})$;
    Propose a new state $(\mathcal{R}', \boldsymbol{\theta}')$ by sampling from the proposal density
      $\kappa_\Psi(\cdot, \cdot|\mathcal{R}^{(t)}, \boldsymbol{\theta}^{(t)})$;
    Compute the Acceptance Ratio $\mathcal{A}(\mathcal{R}', \boldsymbol{\theta}'|\mathcal{R}^{(t)}, \boldsymbol{\theta}^{(t)})$ from (3.11);
    **if** $U < \mathcal{A}(\mathcal{R}', \boldsymbol{\theta}'|\mathcal{R}^{(t)}, \boldsymbol{\theta}^{(t)})$ **then**
        $(\mathcal{R}^{(t+1)}, \boldsymbol{\theta}^{(t+1)}) \leftarrow (\mathcal{R}', \boldsymbol{\theta}')$
    **else**
        $(\mathcal{R}^{(t+1)}, \boldsymbol{\theta}^{(t+1)}) \leftarrow (\mathcal{R}^{(t)}, \boldsymbol{\theta}^{(t)})$
    **end**
**end**
```

3.3.3 Prior Distributions Specification

An important element of Bayesian Inference is the choice of prior distributions for the unobserved quantities. While these distributions are meant to reflect previous information that can be accounted in the model, the general forms of these distributions are often restricted to a specific family, that have good analytical and computational properties, while still preserving some flexibility to include prior information in the form of hyper-parameters that may have useful interpretations depending on the family of prior distributions.

In this work, for the prior distribution of the RPS, $q(\mathcal{R})$, we consider a class of probability functions proportional to a function that penalizes complex models

$$q(\mathcal{R}) = \frac{\beta^{-\alpha d|\mathcal{R}|}}{\eta_{\mathcal{M}}(\alpha, \beta)}, \tag{3.12}$$

where $\alpha > 0$ and $\beta > 0$ are hyper-parameters that can be interpreted as composing a multiplicative penalty introduced for each additional relative position introduced and $\eta_{\mathcal{M}}(\alpha, \beta) = \sum_{\mathcal{R}' \in \mathcal{M}} \beta^{-\alpha d|\mathcal{R}|}$ is a normalizing constant so that q is a proper distribution on $\mathcal{M}$.

Note that the ratio for two RPSs, $\mathcal{R}$ and $\mathcal{R}'$, $q(\mathcal{R})/q(\mathcal{R}') = \beta^{\alpha d(|\mathcal{R}'|-|\mathcal{R}|)}$, is an important quantity for the acceptance ratios of the Reversible-Jump proposal described in (3.11). There are key interpretations for this value such as, any two RPSs with the same number of relative positions have the same values for the function q, simplifying the expression for moves that preserve the complexity of the RPS. Also, $q(\mathcal{R})/q(\mathcal{R}') = \beta^{-\alpha d}$ when $|\mathcal{R}| = |\mathcal{R}'| + 1$, which corresponds to all the other proposal moves that include or exclude positions one at time.

For the prior distribution of $\boldsymbol{\theta}$ given a RPS, $\phi(\boldsymbol{\theta}|\mathcal{R})$, a standard choice is to use independent normal distributions, with a given fixed prior variance, σ_p^2, and mean equal to the zero vector, i.e.,

$$\phi(\boldsymbol{\theta}|\mathcal{R}) = \frac{1}{(2\pi\sigma_p^2)^{|\mathcal{R}|d/2}} \exp\left(-\frac{1}{2\sigma_p^2} \sum_{\mathbf{r} \in \mathcal{R}} \boldsymbol{\theta}_{\mathbf{r}}^\top \boldsymbol{\theta}_{\mathbf{r}} \right). \tag{3.13}$$

One of the main advantages of this prior distribution is that ratios, $\phi(\boldsymbol{\theta}|\mathcal{R})/\phi(\boldsymbol{\theta}'|\mathcal{R}')$, used for computing acceptance ratios of the Reversible Jump algorithm described previously, can be computed more efficiently due to advantages from good analytical properties. For example, the ratio is always 1 for Position Swap moves and the density (or the inverse of) of a d-dimensional Normal distribution for Birth and Death moves. Since those computations are carried in logarithmic scale, most of the terms involving $\boldsymbol{\theta}$ will be sums of quadratic forms that are simple to evaluate.

3.4 Simulation Study

In order to validate the practical use of the proposed Reversible-Jump algorithm and understand the effect of choices of the RPS prior distribution on the selected models, we conducted a simulation study where three MRFs on a 150×150 lattice, $\mathbf{Z}^{(i)}$, were simulated using sparse interaction structures $\mathcal{R}^{(i)}$, $i = 1, 2, 3$, and alphabet $\mathcal{Z} = \{0, 1, 2\}$. The sparse RPSs considered have increasing complexity and are specified as follows:

- $\mathcal{R}_1 = \{(1, 0), (0, 1)\}$,

- $\mathcal{R}_2 = \{(1,0),(0,1),(3,3)\}$,
- $\mathcal{R}_3 = \{(1,0),(0,1),(3,3),(3,0)\}$,

and for the maximal RPS we considered $\mathcal{R}_{\max}$ containing all relative positions within a maximum distance of 5 from the origin, excluding positions that are the opposite of another one included to ensure that it is a proper RPS, as illustrated in Figure 5.

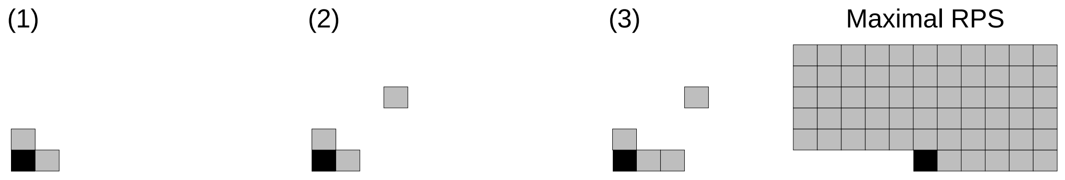


Figure 5 – Interaction structures considered in simulations $\mathcal{R}_i$, $i = 1,2,3$ and maximal interaction structure $\mathcal{R}_{\max}$ considered in the reversible jump algorithm.

With $|\mathcal{Z}| = 3$, a total of $d = 8$ coefficients may vary for each relative position $\mathbf{r}$, therefore, $\mathcal{R}_{\max}$, which has a total of 60 positions, is associated with 480 free interaction coefficients when every relative position is included, whereas $\mathcal{R}_3$, the most complex of the three RPSs that generated the data, represents a model with 32 free interaction coefficients. This reduction from 480 to 32 (or less) free quantities in a model may be extremely useful for inference by reducing complexity and computational cost, as long the interactions within the selected set of relative positions can capture most of the dependence structure of the data.

We considered the same interaction coefficients $\theta_{a,b,\mathbf{r}}$ across simulations for overlapping positions $\mathbf{r}$ with values described in Table 1 and the simulated observations are presented in Figure 6. The coefficient values were selected to generate different patterns between sampled images and it is not clear, especially for the second and third simulated fields, which set of relative position best describes the patterns generated.

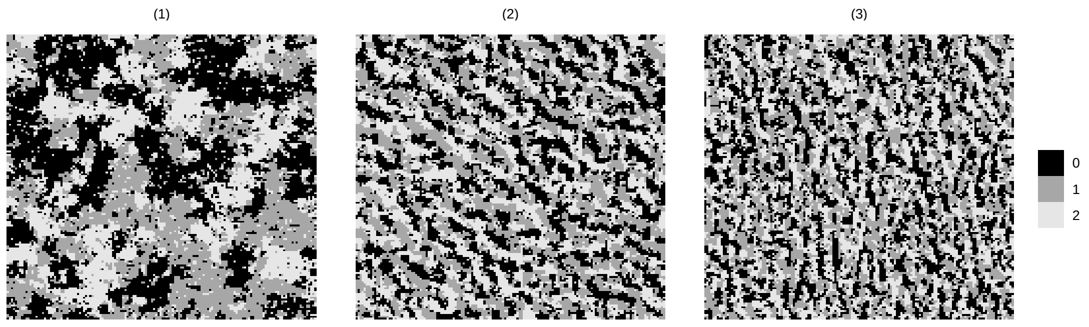


Figure 6 – Simulated 150×150 MRFs $\mathbf{z}^{(i)}$, $i = 1,2,3$.

Table 1 – $\theta_{a,b,\mathbf{r}}$ used in simulations.

$\mathbf{r}$	a	$b=0$	$b=1$	$b=2$	$\mathbf{r}$	a	$b=0$	$b=1$	$b=2$
	0		-1.0	-1.0		0		-1.0	-1.0
$(1,0)$	1	-1.0	0.0	-1.0	$(1,0)$	1	-1.0	0.0	-1.0
	2	-1.0	-1.0	0.0		2	-1.0	-1.0	0.0
	0		0.3	0.3		0		0.3	0.3
$(3,3)$	1	0.3	0.0	0.3	$(3,0)$	1	0.3	0.0	0.3
	2	0.3	0.3	0.0		2	0.3	0.3	0.0

Prior Distributions and Algorithm tuning

For the prior distributions we considered $q(\mathcal{R})$ as in (3.12) for the RPS prior, considering $\beta = |\mathcal{S}|$ and we ran the algorithm over a grid of 10 values for α (0, 0.1, 0.5, 1, 1.5, 2, 2.5, 3, 5, 10) for each simulation. This specific setup corresponds to a prior distribution over the RPS that can interpreted as a penalty to larger-complexity models similar to the penalty introduced when using the Bayesian Information Criterion (BIC). When looking at the pseudoposterior in logarithmic scale, an additive term, $-\alpha d|\mathcal{R}|\log(|\mathcal{S}|)$, appears and $d|\mathcal{R}|$ is the number of free parameters of the model and $|\mathcal{S}|$ can be roughly interpreted as the sample size. The value of α controls the penalty on the prior probability for more complex models and it is directly related to how complex the higher pseudoposterior models tend to be, with a larger value of α leading to more mass concentrated on simpler models. Since the coefficients themselves are hardly interpretable, we chose to use vague priors for varying-dimensional vector $\boldsymbol{\theta}$, considering independent Gaussian priors with mean 0 variance and prior variance of $\sigma_p^2 = 10$ for each of its components, regardless of what is the RPS associated with it.

As for the parameters involved in the proposal kernel, we executed multiple short pilot runs to evaluate whether high pseudoposterior regions (RPS and coefficients) were reached within a sensible number of iterations and acceptance rates were at a reasonable level. We concluded that $\sigma_{\mathrm{s}}^2 = \sigma_{\mathrm{bd}}^2 = 0.15$, $\sigma_{\mathrm{w}}^2 = 0.005$ and $\nu = 0.1$ resulted in good balance between exploring the complex space that is composed by the RPS and the varying-dimension coefficient vector while maintaining the acceptance rate at reasonable levels. Low values for ν should be used in order to "concentrate" the weights sampled from the Dirichlet distribution in a few values when proposing a Split move.

For the mixture probabilities probabilities, we chose $p_\Psi(\mathcal{R})$ always proportional to 4 for $\Psi = \mathrm{w}$ and 1 for the remaining types of moves that are valid for the current state $\mathcal{R}$, resulting in

$$\begin{aligned} p_{\mathrm{w}}(\mathcal{R}) &\propto 4\mathbb{1}(\mathcal{R} \neq \varnothing) \qquad & p_{\mathrm{bd}}(\mathcal{R}) &\propto 1 \\ p_{\mathrm{sw}}(\mathcal{R}) &\propto \mathbb{1}(\mathcal{R} \neq \mathcal{R}_{\max}, \mathcal{R} \neq \varnothing) & & \\ p_{\mathrm{m}}(\mathcal{R}) &\propto \mathbb{1}(\mathcal{R} \neq \mathcal{R}_{\max}, \mathcal{R} \neq \varnothing) \qquad & p_{\mathrm{s}}(\mathcal{R}) &\propto \mathbb{1}(\mathcal{R} \neq \mathcal{R}_{\max}, \mathcal{R} \neq \varnothing). \end{aligned} \tag{3.14}$$

It is important to note that not every type of move is well-defined and not constant for every RPS. For example, a random walk move cannot propose anything meaningful when $\mathcal{R} = \varnothing$, regardless of how it is defined, and a swap move cannot be completed with $\mathcal{R} = \mathcal{R}_{\max}$. This "prohibitions" introduced by setting some probabilities to zero ensure that we never propose moves that result in the exacts same values of the current state, which would add no meaningful information to the sampled chain while demanding computing resources.

The **initial state** of the chain is another factor that should be controlled in order reach equilibrium with less iterations. Since the proposal distribution was designed mainly to propose new points that jump between the modes of the coefficient distributions for different RPSs that follow certain properties, better mixing is achieved when the initial state has coefficients near the mode of the pseudoposterior for the initial RPS. In order to obtain a reasonable initial state, we chose to use a warm-up run of 5000 iterations starting with $\mathcal{R}_{\max}$ and only proposing points using the random walk move, i.e., $p_{\mathrm{w}}(\mathcal{R}_{\max}) = 1$ in the warm-up run. This results in a chain that slowly drifts towards the region of the mode of coefficient values associated with $\mathcal{R}_{\max}$. We used the last sampled state of this warm-up run as the initial state for the actual Reversible Jump run.

Results

In all thirty simulations (3 models $\times$ 10 α-values), the Markov chain sampled in the warm-up stage quickly converged to a stable distribution suggesting that the initial state used for the RJMCMC was close to a sample taken from the coefficients pseudoposterior for the maximal RPS.

Figure 7 illustrates the sampled chain behavior in each simulation for a specific value of α and each interaction coefficient. Coefficients associated with some relevant positions are highlighted and colored so they can be tracked across iterations. The values for iterations in the warm-up stage (indexed by $t < 0$) cannot be clearly identified as the coefficients for all 60 positions are included in this stage, but as the main Reversible Jump run starts, the number of positions included quickly reduces to no more than 4 within a few iterations. In this figure, lines may "appear" or "disappear" as relative positions are included or excluded, respectively, from the sampled RPS, and lines may change colors when swap moves are accepted.

For every chain sampled with more strict RPS priors ($\alpha \geqslant 1$), after moves between different RPSs, the Reversible Jump algorithm reached a state where only moves within the same RPS (random walk moves) were accepted, even when different relative positions were being constantly proposed. This "final" RPS varies depending on the simulation (as in the examples of Figure 7) and the value of α. As α increases, the chains converge to smaller RPSs excluding some positions that were part of the RPS used for

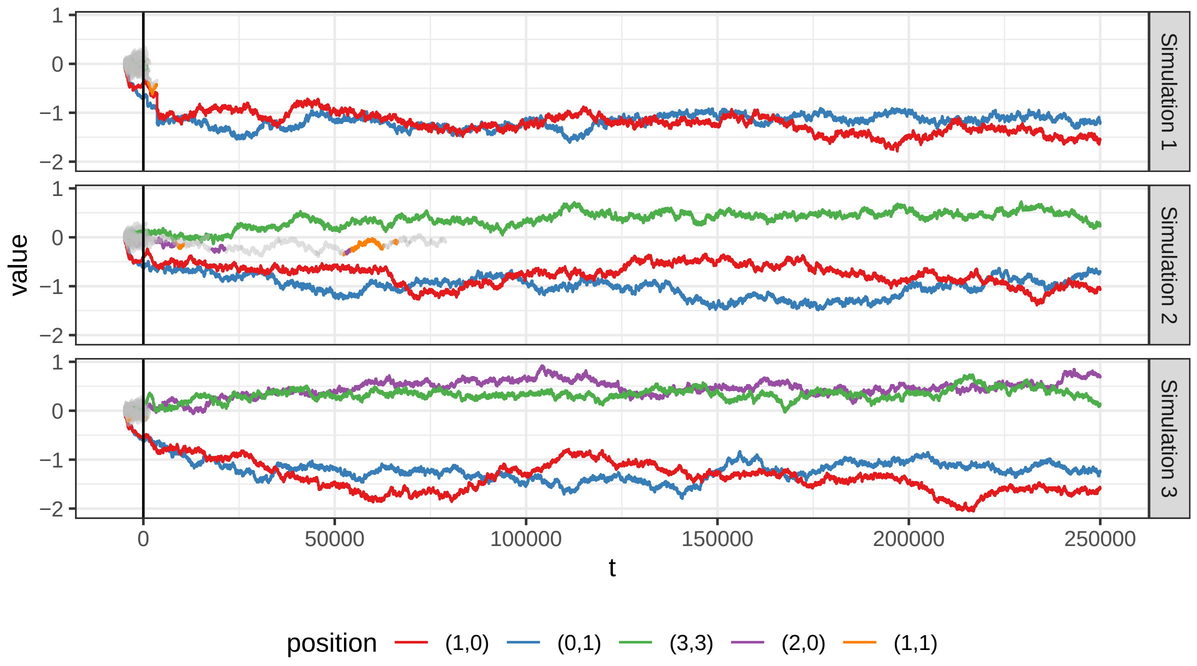


Figure 7 – Values of the RJMCMC sampled for the interaction coefficients $\boldsymbol{\theta}_{\mathbf{r},1,0}$ for multiple relative positions $\mathbf{r}$ in each simulation with $\alpha = 1.5$. Coefficients associated with some particular set of relative positions are highlighted and colored and the remaining ones are represented with gray lines. Iterations of the warm-up run are indexed -4999 to 0.

simulating the data. Under less penalizing priors ($\alpha \leqslant 1$), supersets of the RPS used in the simulation were obtained in the chains, with the exceeding relative positions constantly switching.

While visualizing sets of relative positions or the multiple coefficients with varying dimensions involved is not possible, we can use the marginal proportions which each relative position appears in the sampled chain as a quick way to understand which positions are more relevant for that run. Figures 8, 9 and 10 present these proportions for all the values of α used in simulations 1, 2 and 3, respectively. These proportions were computed with a burn-in of 100000 iterations and sampling only multiples of 10 to reduce serial correlation, so that we can conclude that a run mostly concentrated in a few relative positions not only had those positions included in most of the iterations, but also converged to that final RPS quickly removing unnecessary positions in the burn-in period.

Considering the results obtained, we conclude that for the simulated scenarios, a prior distribution with $\alpha = 1.5$ or 2 converges rapdly and leads to consistent results concentrating high pseudoposterior mass on the RPS that was used to generate the data. In fact, in the proposed study, the algorithm was sucessfulin identifying the RPS that generated the data when applying the sparse estimator from (3.6) for many levels of trhesholding constants especially for $\alpha = 1.5$ and 2. It is an interesting fact that $\alpha = 2$

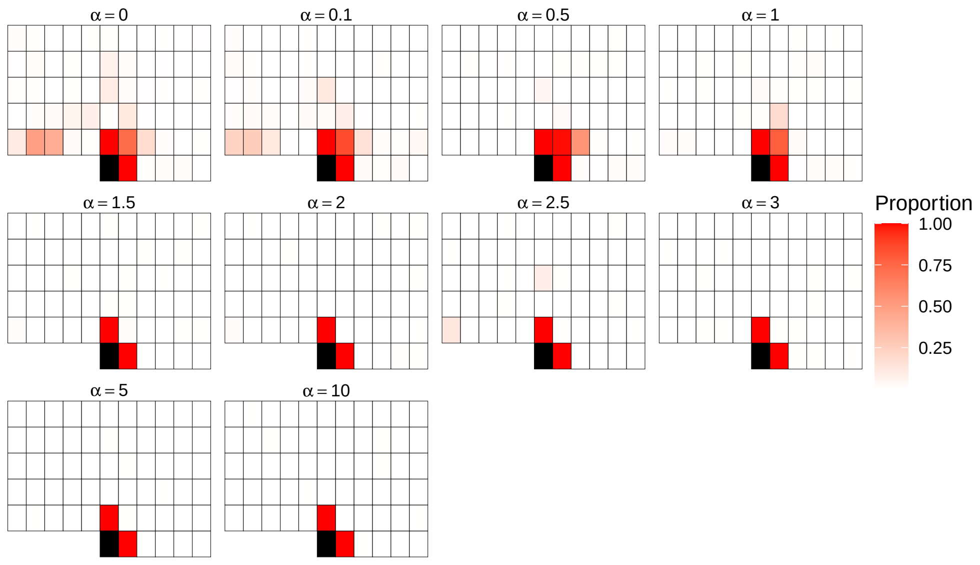


Figure 8 – Marginal pseudoposterior probabilities estimated of the inclusion of each relative position for Simulation 1, varying the prior distribution of the RPS, $q(\mathcal{R})$, according to α.

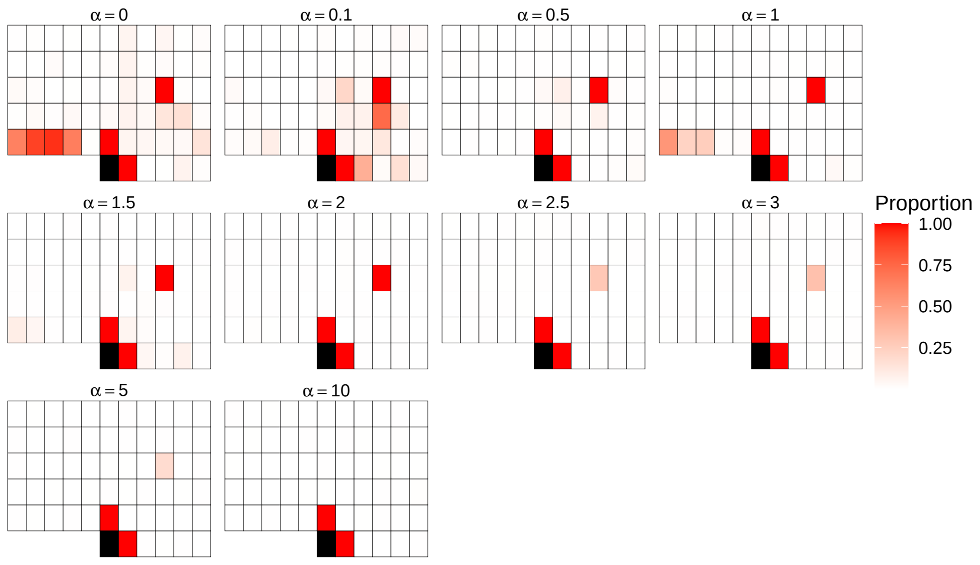


Figure 9 – Marginal pseudoposterior probabilities estimated of the inclusion of each relative position for Simulation 3, varying the prior distribution of the RPS, $q(\mathcal{R})$, according to α.

makes the prior distribution numerically equivalent to the penalty used by the Bayesian Information Criterion (BIC) used for model (variable) selection.

3.5 Application to Synthesis of Texture Image

In order to evaluate the proposed model selection methodology in a real data context, we apply the proposed algorithm to a discrete texture image obtained as a result

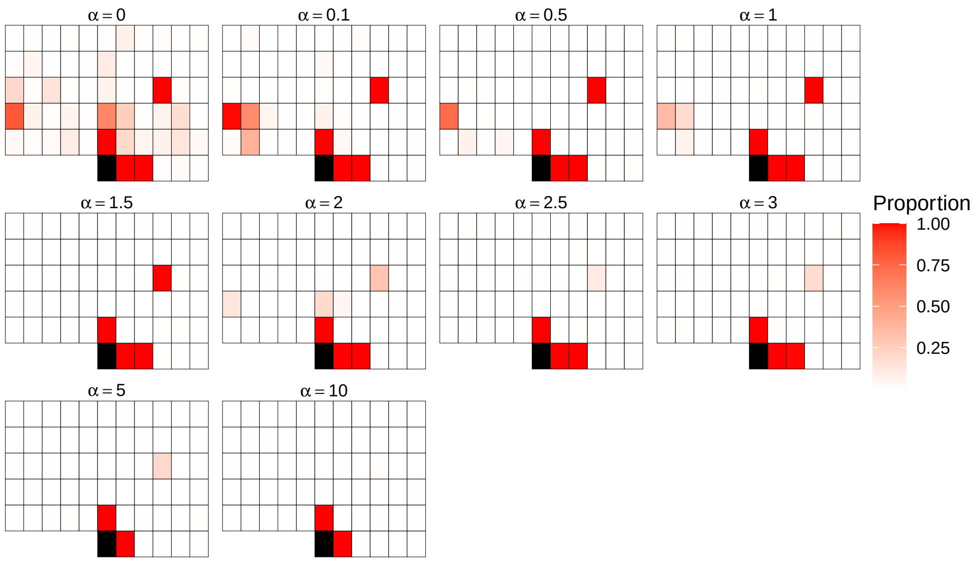


Figure 10 – Marginal pseudoposterior probabilities estimated of the inclusion of each relative position for Simulation 3, varying the prior distribution of the RPS, $q(\mathcal{R})$, according to α.

of the textile images analysis in Freguglia et al. (2020). In the original work, a Gaussian mixture with 5 components, driven by the Markov Random Field model described in section 3.2, is used to describe grayscale continuous-valued images of dyed textiles and one of the products of the analysis is a pixel-wise segmentation of which mixture component was estimated as the most probable. The interaction coefficients of the hidden MRF were originally estimated considering a complete region, with every position within a maximum distance of 5, but we are interested in investigating whether similar interactions for the mixtures components could be described by sparser interaction structures, producing synthetic texture images that have the same patterns as the reference image. We will consider the 200 by 200 subset of one of the estimated discrete images presented in Figure 11, and denote it as $\mathbf{z}^*$.

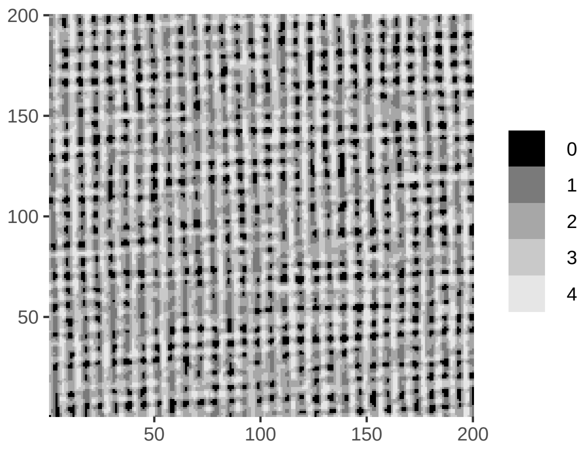


Figure 11 – A 200 by 200 pixels texture image with 5 colors ($C = 4$), denoted $\mathbf{z}^*$.

Considering as input the image data from Figure 11, our goal is to determine whether the complete interaction structure is necessary to properly model the observed random field or a sparser interaction structure could be used, without significant differences in terms of statistical inference.

The search for sparse neighborhoods for modeling texture has been the subject of several papers in the literature, among them, Cross and Jain (1983) and Gimel'farb (1996). However, the selection methods used are mostly heuristic. In this work, we propose a novel statistical methodology that allows to perform proper statistical inference.

To run the algorithm, we consider the maximal interaction structure of the algorithm, $\mathcal{R}_{\max}$, as the neighborhood used in the original paper, which includes positions with maximum norm up to 5. Hyper-parameters of prior distributions and of the Reversible Jump algorithm were selected with the same values of the simulation study from section 3.4 and we fix the RPS prior distribution hyper-parameter α as 0.25. While values of α in this range lead to most of the pseudoposterior mass being concentrated on a superset of the RPS used to simulated the datasets, there was still a significant reduction in the number of positions included. In practice, our goal is to keep all the positions required to probabilistically describe the texture pattern while controlling the number of free coefficients by selecting a RPS that is sparse when compared to the complete region originally used.

We ran 500,000 iterations of the proposed Reversible-Jump algorithm after 10,000 warm-up iterations where the random walk move was selected with probability 1, starting from the maximal RPS. In the same way as described for the simulation study from section 3.4, we stated the RJMCMC in an initial state close to a sample of the pseudoposterior for the maximal $\mathcal{R}_{\max}$. The proportions each relative position in $\mathcal{R}_{\max}$ appear in the sampled chain are presented in Figure 12. Given the pseudoposterior distribution, we used the thresholding RPS estimator strategy from (3.6) with a value of $c_{\text{th}} = 0.5$ to select one sparse interaction structure to be used in our result analysis, with a reduction from 60 to 15 relative positions, which corresponds to a reduction of $45 \times 24 = 1125$ free coefficients in the model. Other results could be drawn from RPS pseudoposterior distribution for model selection purposes, such as the set of highest pseudoposterior RPSs or the pseudoposterior probability of a group of RPSs with specific characteristics, depending on the interests of the analysis being made.

Evaluating the results

Differently from the analysis made for the simulation study presented in section 3.4, we do not have the interaction structure that generated the data available to compare to the RPS pseudoposterior distribution obtained. Goodness of fit evaluation strategies in a Bayesian context are proposed in Gelman et al. (1996) and Bayarri and

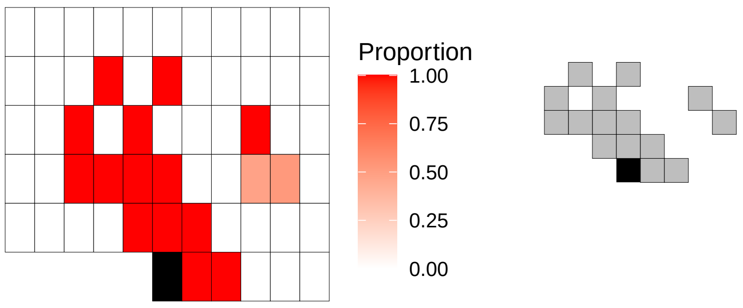


Figure 12 – Map with the proportions of times each relative position is included in the RPS sampled within final (left) and the sparse interaction structure selected with a 0.5 thresholding value $\hat{\mathcal{R}}_{\text{sp}}(0.5)$ (right).

Berger (2000), where metrics can be derived by generating realizations of the model by sampling from the posterior distribution and comparing key statistics from the reference dataset $\mathbf{z}^*$ and those realizations. These methods are not directly applicable in our context because we only have the pseudoposterior distribution available, rather the true posterior distribution (although the differences between these two distributions could be mitigated by using methods as the pseudoposterior adjustment from Bouranis et al. (2017)). Moreover, we cannot define a small number of key statistic to use for the tests, as MRF texture images have a high-dimensional vector of pairwise counts as sufficient statistics.

As an alternative approach to evaluate a single sparse model that was obtained analysing the pseudoposterior distribution, we ran maximum likelihood estimation via stochastic approximation, a standard inference method used in the context of MRFs, and compared results of such inference using the selected sparse interaction structure against the same analyses in scenarios where such sparse RPS is not available. Applying the maximum likelihood estimation via stochastic approximation, we obtain the estimated coefficients, then we generate realizations of the MRF using the estimated coefficients and compare useful statistics for describing the texture from the generated samples and the target dataset $\mathbf{z}^*$.

We considered 3 different reference RPSs for comparison:

1. $\mathcal{R}_{\text{nn}} = \{(1,0),(0,1)\}$: A nearest-neighbor RPS that we will use as a benchmark in comparisons.
2. $\mathcal{R}_{\text{sp}}$: The sparse RPS obtained using the thresholding estimator for our specific thresholding constant choice presented in Figure 12.
3. $\mathcal{R}_{\text{max}}$: The maximal set of relative positions, containing all relative positions within maximum distance of 5, as used in the original paper.

and our goal is to evaluate whether completing an analysis using $\mathcal{R}_{\text{sp}}$ leads to results at least as accurate as obtained using $\mathcal{R}_{\max}$, and at the same time understand how relevant the differences were when compared to the estimates obtained when using a naive model choice with $\mathcal{R}_{\text{nn}}$.

We used the Stochastic Approximation algorithm (Robbins and Monro, 1951) to obtain a Maximum Likelihood estimate of the coefficients for each of the three described RPSs. The algorithm consists of iteratively updating the solution according to a step size sequence $\gamma^{t}_{t\geqslant 0}$ and an estimate of the gradient function, that depends on the sufficient statistic of the model, $T(\cdot)$, computed on the reference dataset $\mathbf{z}^*$ and on a realization, $\mathbf{z}^{(t)}$, of the random field simulated from the current coefficients (see Freguglia and Garcia (2022) for more details on the Stochastic Approximation algorithm used). The algorithm is described by the recursion

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \gamma^{(t)}\left(T(\mathbf{z}^*) - T(\mathbf{z}^{(t)})\right), \tag{3.15}$$

with $\gamma^{(t)}$ being a decreasing sequence. Note that the sufficient statistics $T(\cdot)$ has the same dimension as the vector of free coefficients $\boldsymbol{\theta}$ and, therefore, its indexing and dimension also depends on the associated RPS. We used the proper definitions of $T(\cdot)$ for each of the three RPSs used.

We ran 1500 steps of (3.15) with $\gamma^{(t)} = \frac{1500-t}{1500}$ starting from the zero-valued coefficient vector $\boldsymbol{\theta}_{a,b,\mathbf{r}} = 0$ for every $a, b, \mathbf{r}$ for each of the three RPSs to obtain maximum likelihood estimates of coefficients in each case. Then, we generated 100 samples from each of three models with the estimated coefficients, which we will denote $\tilde{\mathbf{z}}_{\text{nn}} = \left(\mathbf{z}^{(v)}_{\text{nn}}\right)$, $\tilde{\mathbf{z}}_{\text{sp}} = \left(\mathbf{z}^{(v)}_{\text{sp}}\right)$ and $\tilde{\mathbf{z}}_{\max} = \left(\mathbf{z}^{(v)}_{\max}\right)$, $v = 1, \ldots, 100$, for $\mathcal{R}_{\text{nn}}$, $\mathcal{R}_{\text{sp}}$ and $\mathcal{R}_{\max}$ respectively, (the tilde symbol is used to stress the fact that it is a set of MRF realizations). Examples of one of the simulated images for each of the three scenarios considered are presented in Figure 13. Notice that the first image generated from the Nearest-Neighbor model completely misses the features of the texture, whereas there is not much difference between the one generated with the sparse model and the full one.

Finally, we define $\rho_{a,b,\mathbf{r}}(\mathbf{z}) = \sum_{\mathbf{i}\in\mathcal{S}} \mathbb{1}(z_{\mathbf{i}} = a, z_{\mathbf{i}+\mathbf{r}} = b)$ as the count of occurrences of the pair (a, b) within relative position $\mathbf{r}$ (this is part of the vector of sufficient statistics of the model when $\mathbf{r}$ is included in the RPS). To summarize this information, we compute the average counts for a set of realizations as

$$\bar{\rho}_{a,b,\mathbf{r}}(\tilde{\mathbf{z}}) = \sum_{v=1}^{100} \frac{\rho_{a,b,\mathbf{r}}(\mathbf{z}^{(v)})}{100},$$

and the metric

$$\Delta(\tilde{\mathbf{z}}, \mathbf{z}^*) = \log\left(\sqrt{\sum_{a\in\mathcal{Z}}\sum_{b\in\mathcal{Z}}\sum_{\mathbf{r}\in\mathcal{R}_{\max}} \left(\rho_{a,b,\mathbf{r}}(\mathbf{z}^*) - \bar{\rho}_{a,b,\mathbf{r}}(\tilde{\mathbf{z}})\right)^2}\right),$$

which represents the logarithm of the Euclidean distance between the vector containing all the pairwise counts for relative positions in the maximal RPS and the average vector of

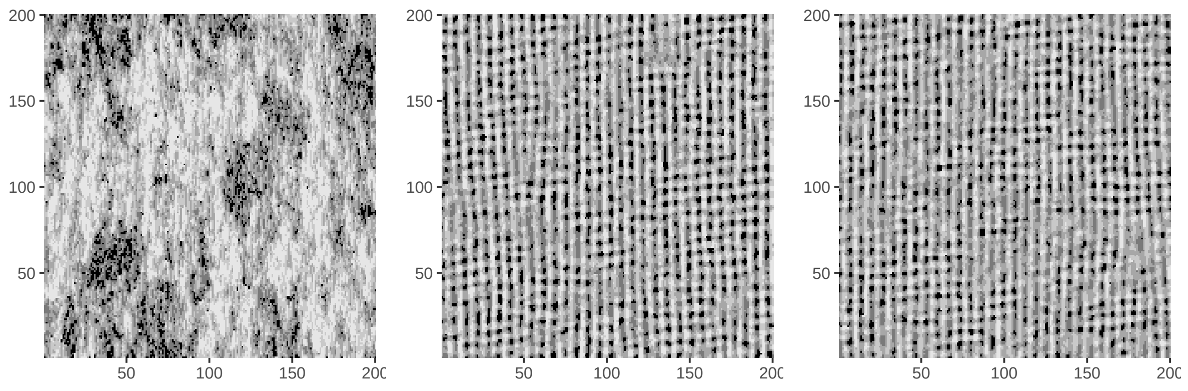


Figure 13 – Simulated realizations of the MRF model obtained from the Maximum Likelihood estimate considering a Nearest-Neighborhood structure (left), Estimated Sparse interaction structure (middle) and Maximal interaction structure (right).

pairwise counts in the set of samples $\tilde{\mathbf{z}}$ for the same relative positions. This is a measure of similarity between the patterns generated from each model and the observed sample $\mathbf{z}^*$. The smaller the value of $\Delta(\tilde{\mathbf{z}}, \mathbf{z}^*)$, the closest the pairwise counts of the reference field $\mathbf{z}^*$ to their expected values are (approximated by the samples average) under a model with that specific RPS and its associated maximum likelihood estimators for the coefficients. Note that we have used the counts in every relative position of the maximal RPS, $\mathcal{R}_{\text{max}}$, because we want to ensure not only the counts of relative positions included in the RPS used for estimation are similar to $\mathbf{z}^*$, but also for a larger common set of statistics across the three scenarios.

Table 2 – $\Delta(\tilde{\mathbf{z}}, \mathbf{z}^*)$ values considering the three sets of samples generated from the model estimated with different RPSs.

RPS	$\mathcal{R}_{\text{nn}}$	$\mathcal{R}_{\text{sp}}$	$\mathcal{R}_{\text{max}}$
$\Delta(\tilde{\mathbf{z}}, \mathbf{z}^*)$	10.6092	8.1320	9.5734

As pointed before, looking at the generated samples displayed in Figure 13, the image generated from $\mathcal{R}_{\text{nn}}$ have a completely different pattern than the original dataset $\mathbf{z}^*$ and both the examples generated $\mathcal{R}_{\text{sp}}$ and $\mathcal{R}_{\text{max}}$ presented patterns very similar when compared to $\mathbf{z}^*$. This is confirmed comparing the values of the computed values $\Delta(\tilde{\mathbf{z}}, \mathbf{z}^*)$ presented in Table 2. We see that, as expected, the model estimated using the benchmark nearest-neighbor structure presented the highest distance between pairwise counts of $\mathbf{z}^*$ and their expected values in the model, and the model estimated from $\mathcal{R}_{\text{sp}}$ had their expected statistics closer to the observed in $\mathbf{z}^*$ than the ones computed using $\mathcal{R}_{\text{max}}$.

Therefore, we conclude that estimating coefficients from a sparse interaction structure, obtained by thresholding the marginal RPS pseudoposterior distribution obtained from our proposed algorithm, has lead to improved maximum likelihood estimation, both in terms of computational costs, since there was significantly less coefficients to compute, and in terms of how similar the expected value of wide set of statistics is from the observed field used for estimation.

3.6 Conclusion

We propose a novel approach to model selection in Markov Random Field models with pairwise interactions in a Bayesian context by using a Reversible Jump Markov Chain algorithm with a proposal distribution that was developed especially aiming for efficient jumping between sets of relative positions, quickly reaching the models with highest posterior mass.

In order to overcome the intractability of the normalizing constant inherent from MRF models, we have used pseudolikelihood and proceeded with the analyses based on the pseudoposterior distribution as a proxy of the true posterior distribution that cannot be evaluated directly. It must be clear that the algorithm proposed can be directly adapted for different strategies that may be used to deal with the intractable constants and approximate likelihood function, such as Adjusted Pseudolikelihoods (Bouranis et al., 2018) and Monte-Carlo approximations (Atchadé et al., 2013), but not every approximation method is well suited for the varying-dimension feature of the model selection framework.

The main contributions of this work are:

1. Proposing a Bayesian framework for both the interaction structure and the parameters of the Markov Random Field model considered.
2. Constructing a Reversible Jump proposal kernel that is especially useful for models within the proposed framework.
3. Exploring the effects of the prior distribution specification in some scenarios to understand their properties while still attaining enough flexibility in the methods so that any choice of prior distribution can be considered.

We have used artificially generated datasets and an application to real data in the context of texture synthesis, to evaluate the strengths of our method and study how the algorithm behaves under different configurations and concluded that our method leads to promising results for selecting sparse interaction structures for MRFs.

As future work it should be possible to develop new methods for approximating the likelihood function (and the posterior distribution as a consequence) that can produce good approximations across the varying-dimension spaces, as an alternative to the pseudolikelihood. Also, to develop a constant free procedure for model selection. That is, to provide a solution for the problem of optimal choice of the penalty constant α.

The reproducible R language code is available as part of the R package **mrf2dbayes**, available in `https://github.com/Freguglia/mrf2dbayes`. It leverages the data structures provided by the **mrf2d** package (Freguglia and Garcia, 2022) and extends it by introducing additional Bayesian analysis elements such as the likelihood approximation method (in this work, the pseudolikelihood) and Metropolis-Hastings algorithms. The source code used to reproduce the results in this work is available upon request.

Acknowledgments

This work was funded by Fundação de Amparo à Pesquisa do Estado de São Paulo - FAPESP grant 2017/25469-2 and by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

4 Detecting Renewal States in Chains of Variable Length via Intrinsic Bayes Factors

This chapter corresponds to the pre-print research article Freguglia and Garcia (2021) available in the `arxiv.org` repository.

4.1 Introduction

Markov Chains with variable length are useful stochastic models that provide a powerful framework for describing transition probabilities for finite-valued sequences due the possibility of capturing long-range interactions while keeping some parsimony in the number of free parameters. These models were introduced in the seminal paper of Rissanen (1983) for data compression and became known in the statistics literature as Variable Length Markov Chain (VLMC) by Bühlmann and Wyner (1999), and as Probabilistic Suffix Trees (PST) in the machine learning literature (Ron et al., 1996). The idea is that, for each past, only a finite suffix of the past is enough to predict the next symbol. Rissanen called *context*, the relevant ending string of the past. The set of all contexts can be represented by the set of leaves of a rooted tree if we require that no context is a proper suffix of another context. For a fixed set of contexts, estimation of the transition probabilities can be easily achieved. The problems lies into estimating the set of contexts from the available data. In his seminal 1983 paper, Rissanen introduced the Context algorithm, which estimates the context tree by aggregating irrelevant states in the history of the process using a sequential procedure. A nice introductory guide to this type of models and particularly to the Context algorithm can be found in Galves and Löcherbach (2008).

Many of the tree model methods related to data compression tasks involve obtaining better predictions based on weighting over multiple models. A classical example is the Context-Tree Weighting (CTW) algorithm (Willems et al., 1995), which computes the marginal probability of a sequence by weighting over all context trees and all probability vectors using computationally convenient weights. Using CTW, Csiszár and Talata (2006) showed that context trees can be consistently estimated in linear time using the Bayesian information criterion (BIC). These weighting strategies can be translated to a Bayesian framework where unobserved parameters of a probabilistic system are treated as additional random components with given prior distribution and inference is based on integrating over the nuisance parameters, which is a form of weighting over these quantities based on the prior distribution. Nonetheless, inference performed following the Bayesian paradigm for VLMC models is a relatively recent topic of research. Some works that explicitly use Bayesian statistics in combination with VLMC models are Dimitrakakis (2010) which introduced an online prediction scheme by adding a prior, conditioned on context, on the Markov order of the chain, and Kontoyiannis et al. (2020) which provided more general tools such as posterior sampling through Metropolis-Hastings algorithm and Maximum a Posteriori context tree estimation focusing on model selection, estimation, and sequential prediction. A Bayesian approach for model selection in high-order Markov chains, allowing conditional probabilities to be partitioned into more general structures than the tree-based structures of VLMC models, is also proposed in Xiong et al. (2016).

As aforementioned, the effort was mostly concentrated in estimating the context tree structure. On the other hand, hypothesis testing for VLMC is a difficult topic, first tackled by Balding et al. (2009) and pursued further by Busch et al. (2009) using a Kolmogorov-Smirnov-type goodness-of-fit test, to compare if two samples come from the same distribution. Under the Bayesian paradigm, hypothesis testing is done though Bayes Factor, but computing Bayes Factors may depend on integrations that require enormous computational effort depending on the random objects and hypotheses involved. Particularly for VLMC problems, Bayes Factors require summing over the set of all possible context trees, which cardinality grows doubly exponentially with the maximum depth considered, quickly becoming intractable. To avoid such intractable quantities, we use the Monte Carlo approximations of the Intrinsic Bayes Factor (Berger and Pericchi, 1996), which is based on averaging over posterior distributions, that tend to be highly concentrated within a small set of context trees for VLMC models, and have been used recently in many different fields with the same purpose, such as Cabras et al. (2015), Charitidou et al. (2018) and Villa and Walker (2021). Alternatives to Bayes Factors based on using posterior distributions instead of the prior distribution in integrations have been evolving over the past decades, the classical method using this strategy is the Posterior Bayes Factor (Aitkin, 1991), with applications in a variety of models such as Aitkin (1993) and Aitkin et al. (1996).

In this work, we focus on one characteristic of interest in a context tree, the presence of a renewal state or a renewal context. Renewal states play an important role in some computational methods frequently used in statistical analysis such as designing Bootstrap schemes and defining proper cross-validation strategies based on blocks. Therefore, having some methodology not only to detect renewal states, but also to quantify how plausible these assumptions are, can improve the robustness of analysis at the cost of some pre-processing computations. For example, Galves et al. (2012) proposed a constant free algorithm (Smallest Maximizer Criterion) to find the tree that maximizes the BIC based on a Bootstrap scheme that uses the renewal property of one of the states. To the best of our knowledge, using Bayes Factors for evaluating hypotheses involving probabilistic context trees is a topic that has not been explored.

4.2 Variable-Length Markov Chains

4.2.1 Model Description

Let $\mathcal{A}$ be an *alphabet* of m symbols, and without loss of generality, consider $\mathcal{A} = \{0, 1, \ldots, (m-1)\}$ for simplicity. For $t_2 > t_1$, a string $(z_{t_1}, \ldots, z_{t_2}) \in \mathcal{A}^{t_2 - t_1 + 1}$ will be denoted by $\mathbf{z}_{t_1}^{t_2}$ and its length by $\ell(\mathbf{z}_{t_1}^{t_2}) = t_2 - t_1 + 1$. A sequence $\mathbf{s}_{n-l}^{n}$ is a *suffix* of a string $\mathbf{z}_1^n$ if $s_j = z_j$ for all $j = n-l, \ldots, n$. If $l < n$ we say that s_{n-l}^n is a *proper suffix* of the string $\mathbf{z}_1^n$.

Definition 1. *Let $L > 0$ and $\tau \subset \cup_{j=1}^{L} \mathcal{A}^j$ be a set of strings formed by symbols in $\mathcal{A}$. We say that τ satisfies the* suffix property *if, for every string $\mathbf{s}_{-j+j'}^{-1} = (s_{-j+j'}, \ldots, s_{-1}) \in \mathcal{A}^{j-j'}$, $\mathbf{s}_{-j+j'}^{-1} \in \tau$ implies that $\mathbf{s}_{-j}^{-1} \notin \tau$ for $j > 1, j' = 1, \ldots, j$.*

Definition 2. *Let $L > 0$ and $\tau \subset \cup_{j=1}^{L} \mathcal{A}^j$ be a set of strings formed by symbols in $\mathcal{A}$. We say that τ is an* irreducible tree *if, no string belonging to τ can be replaced by a proper suffix without violating the suffix property.*

Definition 3. *Let τ be an irreducible tree. We say that τ is* full *if, for each string $\mathbf{s} \in \tau$, any concatenation of a symbol $k \in \mathcal{A}$ and a suffix of $\mathbf{s}$ is the suffix of a string $\mathbf{s}' \in \tau$.*

Examples

Suppose that we have a binary alphabet $\mathcal{A} = \{0, 1\}$, then:

- $\tau = \{0, 1, 11\}$ does **not satisfy the suffix property** because it contains both the strings 1 and 11.
- $\tau = \{0, 01\}$ is **not an irreducible tree**, because the string 01 can be replace by its suffix, 1, without violating the suffix property, as the set $\{0, 1\}$ satisfies the suffix property.
- $\tau = \{0, 011, 111\}$ **is irreducible**, but it is **not full** because 1 is a suffix of a string in τ (either 011 or 111), but 01 (the concatenation of $0 \in \mathcal{A}$ and 1) is not.
- $\tau_1 = \{0, 100, 101, 110, 111\}$, $\tau_2 = \{01, 00, 10, 11\}$ and $\tau_3 = \{0, 10, 110, 1110, 1111\}$ are **full irreducible** trees.

A full irreducible tree τ can be represented by the set of leaves of a rooted tree with a finite set of labeled branches such that

(1) The root node has no label,

(2) each node has either 0 or m children (fullness) and

(3) when a node has m children, each child has symbol of the alphabet $\mathcal{A}$ as a label.

The elements of τ will be called *contexts* and we will refer to full irreducible trees as *context trees* henceforth. Figure 14 presents 3 examples of contexts trees. The *depth* ℓ of a tree τ is given by the maximal length of a context belonging to τ, defined as

$$\ell(\tau) = \max\{\ell(\mathbf{z}); \mathbf{z} \in \tau\}.$$

In this work we will assume that the depth of the tree is bounded by an integer L. In this case, it is straightforward to conclude that, for any string $\mathbf{z}_{t_1}^{t_2}$ with at least $L+1$ symbols, there exist a suffix $\mathbf{z}_{t_2-l}^{t_2}$ and a leaf of τ such that the symbols between the leaf (including) and the root node are exactly $\mathbf{z}_{t_2-l}^{t_2}$. Galves et al. (2012) referred to this property as the *properness* of a context tree.

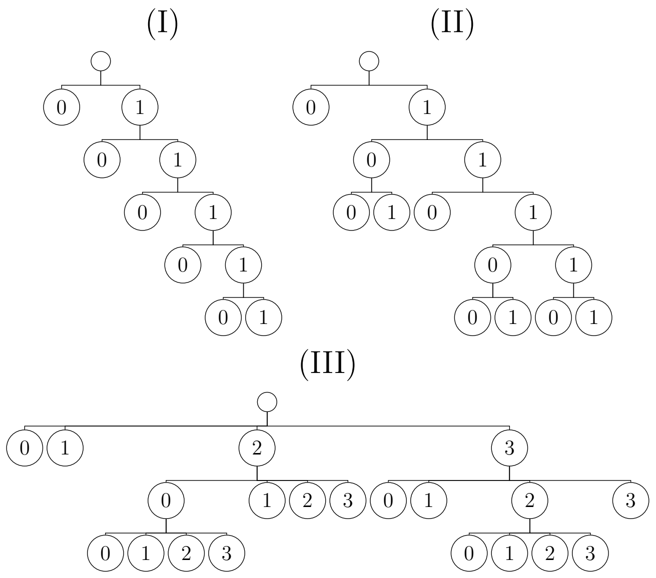


Figure 14 – Examples of context trees.

For each context tree τ, we can associate a family of probability measures indexed by elements of τ,

$$\mathbf{p} = \{p(\cdot|\mathbf{s}) : \mathcal{A} \to [0,1]; \mathbf{s} \in \tau\}.$$

The pair $(\tau, \mathbf{p})$ is called a *probabilistic context tree.*

Given a tree τ with the described properties and depth bounded by L, define a *suffix mapping function* $\eta_\tau : \cup_{j=L+1}^{\infty} \mathcal{A}^j \to \tau$ such that $\eta_\tau(\mathbf{z}_{t_1}^{t_2}) = \mathbf{z}_{t_2-l}^{t_2}$ is the unique suffix $\mathbf{z}_{t_2-l}^{t_2} \in \tau$.

Definition 4. *A sequence of random variables $\mathbf{Z} = (Z_t)_{t=1}^{T}$ with state space $\mathcal{A}$ is a Variable Length Markov Chain (VLMC) compatible with the probabilistic tree $(\tau, \mathbf{p})$ if it satisfies*

$$\mathbb{P}\left(Z_t = k|\mathbf{Z}_1^{t-1} = \mathbf{z}_1^{t-1}\right) = \mathbf{p}(k|\eta_\tau(\mathbf{z}_1^{t-1})), \tag{4.1}$$

for all $L < t \leqslant T$, $\mathbf{z}_1^{t-1} \in \mathcal{A}^{t-1}$, where $\eta_\tau(\mathbf{z}_1^{t-1}) \in \tau$ is the suffix of $\mathbf{z}_1^{t-1}$.

4.2.2 Likelihood Function

In order to extend the scope of VLMC models introduced previously to data involving multiple sequences, we define a VLMC dataset of size I, denoted as $\widetilde{\mathbf{Z}}$, as a set of independent VLMC sequences $\widetilde{\mathbf{Z}} = (\mathbf{Z}^{(i)})_{i=1,\dots,I}$ and $\widetilde{\mathbf{z}} = (\mathbf{z}^{(i)})_{i=1,\dots,I}$ will denote its observed realizations.

For each sequence $\mathbf{Z}^{(i)}$, with length T_i, we will consider its first L elements as constant values, allowing us to write the joint probabilities as a product of transition probabilities in (4.1), without requiring additional parameters for consistently defining the probabilities of the first symbols in each sequence. Hence, the likelihood function is given by

$$f(\widetilde{\mathbf{z}}|\tau, \mathbf{p}) = \prod_{\mathbf{s}\in\tau} \prod_{k=0}^{m-1} (p(k|\mathbf{s}))^{n_{\mathbf{s}k}(\widetilde{\mathbf{z}})}, \tag{4.2}$$

where $n_{\mathbf{s}k}(\widetilde{\mathbf{z}}) = \sum_{i=1}^{I} \sum_{t=L+1}^{T_i} \mathbb{1}\left(z_t^{(i)} = k, \eta_\tau(\mathbf{z}_1^{t-1(i)}) = \mathbf{s}\right)$ counts the number of occurrences of the symbol k after strings with suffix $\mathbf{s}$ across all sequences.

4.2.3 Renewal States

A symbol $a \in \mathcal{A}$ is called a *renewal state* if

$$\mathbb{P}\left(\mathbf{Z}_{t+1}^{t'} = \mathbf{z}_{t+1}^{t'} | \mathbf{Z}_1^{t-1} = \mathbf{z}_1^{t-1}, Z_t = a\right) = \mathbb{P}\left(\mathbf{Z}_{t+1}^{t'} = z_{t+1}^{t'} | Z_t = a\right),$$

for all $t > L$, $t' > t+1$, $\mathbf{z}_1^{t-1} \in \mathcal{A}^{t-1}$, $\mathbf{z}_{t+1}^{t'} \in \mathcal{A}^{t'-t}$. That is, conditioning on $Z_t = a$, the distribution of the chain after t, $(Z_u)_{u>t}$, is independent from the past $(Z_u)_{u<t}$.

This property of conditional independences in a Markov Chain can be directly associated to the structure of the context tree of a VLMC model. For a VLMC model with associated context tree τ, a state $a \in \mathcal{A}$ is a renewal state if a does not appear in any inner node of the context tree. That is, for any context $\mathbf{s} \in \tau$, expressing $\mathbf{s}$ as the concatenation of l symbols $\mathbf{s} = s_l \dots s_2 s_1$, $s_i \neq a$ for $i = 1, \dots, l-1$. In this case, we say that the tree τ is *a-renewing.*

Two out of the three trees displayed in Figure 14 present renewal states. Tree (I) has 0 as a renewal state, Tree (II) has no renewal states due to the presence of the contexts 001 and 00111, Tree (III) has only 1 as a renewal state. If, in Tree (III), the branch formed by contexts 002, 102, 202 and 302 was pruned and substituted by the context 02 only, then 0 would also be a renewal state. Note that a VLMC may contain multiple renewal states.

A remarkable consequence of a being a renewal state is that the random blocks between two occurrences of a are independent and identically distributed. This feature allows the use of block Bootstrap methods, enables a straightforward construction of cross-validation schemes and any other technique that relies on exchangeability properties.

4.3 Bayesian Renewal Hypothesis Evaluation

A VLMC model is fully specified by the probabilistic context tree $(\tau, \mathbf{p})$. The dimension of $\mathbf{p}$ depends on the branches of τ. Both these unobserved components $(\tau, \mathbf{p})$ can be treated as random elements with given prior distribution to carry out inference under the Bayesian paradigm.

From now on, we will use the following notation, for each $\mathbf{s} \in \tau$,

$$\mathbf{p_s} = (p(0|\mathbf{s}), \ldots, p(m-1|\mathbf{s})) \in \Delta_m$$

where Δ_m denotes the m-simplex,

$$\Delta_m = \left\{ \mathbf{x} \in \mathbb{R}^{\{0,1,\ldots,m-1\}} : \sum_{k=0}^{m-1} x_k = 1 \text{ and } \forall j, x_j \geqslant 0 \right\}.$$

In this section we discuss the prior specification for the probabilistic context tree $(\tau, \mathbf{p})$, as well as the resultant posterior distribution and how to perform hypothesis testing using partial and intrinsic Bayes factor.

4.3.1 A Bayesian Framework for VLMC models

We consider a general prior distribution for τ proportional to any arbitrary non-zero function $h : \mathcal{T}_L \to [0, \infty)$ and, given τ, for each $\mathbf{s} \in \tau$, $\mathbf{p_s}$ will have independent Dirichlet priors.

The complete Bayesian system can be described by the hierarchical structure

$$\begin{aligned} \tau &\sim \frac{h(\tau)}{\zeta(h, L)}, & \tau &\in \mathcal{T}_L, \\ \mathbf{p}|\tau &\sim \prod_{\mathbf{s}\in\tau} \frac{\Gamma(\sum_{k=0}^{m-1} \alpha_{\mathbf{s}k})}{\prod_{k=0}^{m-1} \Gamma(\alpha_{\mathbf{s}k})} \prod_{k=0}^{m-1} (p(k|\mathbf{s}))^{\alpha_{\mathbf{s}k}-1}, & \mathbf{p} &\in \Delta_m^{|\tau|}, \\ \widetilde{\mathbf{Z}}|\tau, \mathbf{p} &\sim f(\widetilde{\mathbf{z}}|\tau, \mathbf{p}), & \mathbf{z}^{(i)} &\in \mathcal{A}^{T_i}, \end{aligned} \tag{4.3}$$

where

$$\zeta(h, L) = \sum_{\tau \in \mathcal{T}_L} h(\tau) \tag{4.4}$$

is the normalizing constant of the tree prior distribution and $f(\widetilde{\mathbf{z}}|\tau, \mathbf{p})$ is given by (4.2). We are assuming that the prior distribution for the transition probabilities $\mathbf{p_s}$ are independent Dirichlet distribution with hyperparameters $\boldsymbol{\alpha}_\mathbf{s} = (\alpha_{\mathbf{s}0}, \ldots, \alpha_{\mathbf{s}(m-1)})$. Therefore, the joint distribution of $(\tau, \mathbf{p}, \widetilde{\mathbf{Z}})$ is given by

$$\pi(\tau, \mathbf{p}, \widetilde{\mathbf{z}}) = \frac{h(\tau)}{\zeta(h, L)} \prod_{\mathbf{s}\in\tau} \frac{\Gamma(\sum_{k=0}^{m-1} \alpha_{\mathbf{s}k})}{\prod_{k=0}^{m-1} \Gamma(\alpha_{\mathbf{s}k})} \prod_{k=0}^{m-1} (p(k|\mathbf{s}))^{n_{\mathbf{s}k}(\widetilde{\mathbf{z}}) + \alpha_{\mathbf{s}k} - 1}.$$

Since our interest lies in making inferences about the dependence structure represented by τ rather than the transition probabilities, we can simplify our analysis by marginalizing the joint probability function over $\mathbf{p}$, $\pi(\tau, \widetilde{\mathbf{z}}) = \int \pi(\tau, \mathbf{p}, \widetilde{\mathbf{z}}) d\mathbf{p}$, obtaining a function that depends only on the context tree and the data. The product of Dirichlet densities, assigned as the prior distribution of $\mathbf{p}$, conjugates to the likelihood function, allowing us to express the integrated distribution in closed-form as

$$\pi(\tau, \widetilde{\mathbf{z}}) = \frac{h(\tau)}{\zeta(h, L)} \prod_{\mathbf{s} \in \tau} \frac{\Gamma(\sum_{k=0}^{m-1} \alpha_{\mathbf{s}k}) \prod_{k=0}^{m-1} \Gamma(n_{\mathbf{s}k}(\widetilde{\mathbf{z}}) + \alpha_{\mathbf{s}k})}{\prod_{k=0}^{m-1} \Gamma(\alpha_{\mathbf{s}k}) \, \Gamma(\sum_{k=0}^{m-1} n_{\mathbf{s}k}(\widetilde{\mathbf{z}}) + \alpha_{\mathbf{s}k})}, \tag{4.5}$$

obtained by multiplying the appropriate normalizing constant to achieve Dirichlet densities with parameters $(\alpha_{\mathbf{s}0} + n_{\mathbf{s},0}(\widetilde{\mathbf{z}}), \ldots, \alpha_{\mathbf{s}(m-1)} + n_{\mathbf{s},m-1}(\widetilde{\mathbf{z}}))$ for each $\mathbf{s} \in \tau$, so that the integration is done on a proper density. For a less convoluted notation, we shall denote

$$q(\tau, \widetilde{\mathbf{z}}) = \prod_{\mathbf{s} \in \tau} \frac{\Gamma(\sum_{k=0}^{m-1} \alpha_{\mathbf{s}k}) \prod_{k=0}^{m-1} \Gamma(n_{\mathbf{s}k}(\widetilde{\mathbf{z}}) + \alpha_{\mathbf{s}k})}{\prod_{k=0}^{m-1} \Gamma(\alpha_{\mathbf{s}k}) \, \Gamma(\sum_{k=0}^{m-1} n_{\mathbf{s}k}(\widetilde{\mathbf{z}}) + \alpha_{\mathbf{s}k})}, \tag{4.6}$$

and use, from now on, the shorter expression $\pi(\tau, \widetilde{\mathbf{z}}) = \frac{h(\tau)}{\zeta(h,L)} q(\tau, \widetilde{\mathbf{z}})$.

Finally, the model evidence (marginal likelihood) can now be obtained by summing (4.5) over all trees in $\mathcal{T}_L$,

$$\mathcal{E}(\widetilde{\mathbf{z}}; h) = \sum_{\tau \in \mathcal{T}_L} \pi(\tau, \widetilde{\mathbf{z}}) = \sum_{\tau \in \mathcal{T}_L} \frac{h(\tau)}{\zeta(h, L)} q(\tau, \widetilde{\mathbf{z}}). \tag{4.7}$$

Note that we explicitly describe the model evidence in terms of the prior distribution h as we will be interested in evaluating hypotheses based on different prior distributions.

4.3.2 Bayes Factors for Renewal State Hypothesis

Let $\widetilde{\mathbf{z}} = (\mathbf{z}^{(i)})_{i=1,\ldots,I}$ be a VLMC sample compatible with a probabilistic context tree $(\tau, \mathbf{p})$ where τ has maximum depth L. We will call *maximal tree* the complete tree with depth L and let $a \in \mathcal{A}$ be a fixed state of the alphabet. Our goal is to use Bayes Factors (Kass and Raftery, 1995) to evaluate the evidence in favor of the null hypothesis H_a that τ is *a-renewing* against an alternative hypothesis $H_{\bar{a}}$ that τ is not *a-renewing*. We denote $\mathcal{T}_L^a \subset \mathcal{T}_L$ the set of a-renewing trees with depth no more than L and $\bar{\mathcal{T}}_L^a$ the set of trees with a as an inner node and, consequently, a is not a renewal state for those trees.

We are interested in defining a metric for evaluating the hypothesis $H_a : \tau \in \mathcal{T}_L^a$ against its complement $H_{\bar{a}} : \tau \in \bar{\mathcal{T}}_L^a$ in a Bayesian framework. These hypotheses can be expressed in terms of special prior distributions proportional to functions h_a and $h_{\bar{a}}$, respectively, such that $h_a(\tau) = 0$ if, and only if, $\tau \in \bar{\mathcal{T}}_L^a$. Similarly, $h_{\bar{a}}(\tau) = 0$ if, and only if, $\tau \in \mathcal{T}_L^a$.

The Bayes Factor for H_a against $H_{\bar{a}}$ is defined as

$$\mathrm{BF}_{a,\bar{a}}(\widetilde{\mathbf{z}}) = \frac{\mathcal{E}(\widetilde{\mathbf{z}}; h_a)}{\mathcal{E}(\widetilde{\mathbf{z}}; h_{\bar{a}})} = \frac{\zeta(h_{\bar{a}}, L)}{\zeta(h_a, L)} \frac{\sum_{\tau \in \mathcal{T}_L} h_a(\tau) q(\tau, \widetilde{\mathbf{z}})}{\sum_{\tau \in \mathcal{T}_L} h_{\bar{a}}(\tau) q(\tau, \widetilde{\mathbf{z}})}, \tag{4.8}$$

where $\zeta(\cdot, L)$ is given by (4.4) and q is given by (4.6).

Kass and Raftery (1995) proposed the following interpretation for the quantity $\log_{10}(BF_{a,\bar{a}}(\widetilde{\mathbf{z}}))$ as a measure of evidence provided by the data $\widetilde{\mathbf{z}}$ in favor of the hypothesis that corresponds to a-renewing trees as opposed to the alternative one. A value between 0 and 1/2 is considered to provide evidence that is "Not worth more than a bare mention", "Substantial" for values between 1/2 and 1, "Strong" if they are between 1 and 2 and "Decisive" for values greater than 2. By symmetry, these intervals with negative sign provide the same amount of evidence but reversing the hypotheses considered. Therefore, the sign of $\log_{10}(BF_{a,\bar{a}}(\widetilde{\mathbf{z}}))$ provides a straightforward measure whether the data provides more evidence that the chain is compatible with a context tree τ that is a-renewing or that τ belongs to $\bar{\mathcal{T}}_L^a$.

4.3.3 Metropolis-Hastings algorithm for context tree posterior sampling

Before further development of methods to compute the Bayes Factors from (4.8), we need to introduce a Metropolis-Hastings algorithm for sampling from the marginal posterior distribution of context trees, $\pi(\tau|\widetilde{\mathbf{z}})$. From (4.5) and the Bayes rule we obtain

$$\pi(\tau|\widetilde{\mathbf{z}}) = \frac{\frac{h(\tau)}{\zeta(h,L)} q(\tau, \widetilde{\mathbf{z}})}{\mathcal{E}(\widetilde{\mathbf{z}}; h)} \propto h(\tau) q(\tau, \widetilde{\mathbf{z}}), \tag{4.9}$$

which has a simple expression up to the intractable proportionality terms, suggesting that the Metropolis-Hastings algorithm (Hastings, 1970; Chib and Greenberg, 1995) is an appropriate strategy to obtain an empirical sample from the posterior distribution given by (4.9).

The main step for constructing the algorithm is defining a suitable proposal kernel $\kappa(\tau'|\tau), \tau', \tau \in \mathcal{T}_L$ to move to new context trees from a current tree τ. We propose the use of a graph-based kernel that can be viewed as a modification of the Monte Carlo Markov Chain Model Composition (MC^3) method from Madigan et al. (1995) by defining a neighborhood system $\mathcal{N}$ over $\mathcal{T}_L$ and constructing a proposal kernel that allows transitions only between neighboring trees.

We first specify a set **directed** edges $\mathcal{N}_d$ such that, an edge (τ, τ'), from τ to τ', is included if, and only if, $|\tau \nabla \tau'| = m + 1$ and $|\tau'| > |\tau|$, where ∇ denotes the symmetric difference operator $A \nabla B = (A \cap B^c) \cup (A^c \cap B)$. An equivalent definition is that an edge from τ to τ' is obtained substituting one of the contexts $\mathbf{s} \in \tau$, by the contexts associated with its m children nodes, $\{k\mathbf{s}, k \in \mathcal{A}\}$, in τ'. We refer to this substitution as *growing a branch* from $\mathbf{s}$.

Additionally, we define the *grow* ($\oplus$) and *prune* ($\ominus$) operators, as

$$\oplus(\tau, h) = \{\tau' \in \mathcal{T}_L : (\tau, \tau') \in \mathcal{N}_d \text{ and } h(\tau') > 0\},$$
$$\ominus(\tau, h) = \{\tau' \in \mathcal{T}_L : (\tau', \tau) \in \mathcal{N}_d \text{ and } h(\tau') > 0\}.$$

The operator $\oplus$ maps a tree τ to the set of trees with positive prior distribution that can be obtained by growing new branches from τ, whereas $\ominus$ maps τ to the set of trees in $\mathcal{T}_L$ from which τ can be obtained after growing a branch.

Some important properties that can be easily checked are

1. For every $\tau \in \mathcal{T}_L$, if $\tau' \in \oplus(\tau, h)$ and $h(\tau) > 0$, then $\tau \in \ominus(\tau', h)$.
2. For every $\tau \in \mathcal{T}_L$, if $\tau' \in \ominus(\tau, h)$ and $h(\tau) > 0$, then $\tau \in \oplus(\tau', h)$.
3. For any finite sequence $\tau^{(1)}, \tau^{(2)}, \ldots, \tau^{(N)}$ such that $h(\tau^{(1)}) > 0$ and $\tau^{(n+1)} \in \oplus(\tau^{(n)}, h) \cup \ominus(\tau^{(n)}, h)$, we have $\tau^{(n)} \in \oplus(\tau^{(n+1)}, h) \cup \ominus(\tau^{(n+1)}, h)$.

It follows from Properties 1 and 2 that any context tree τ can be recovered by applying sequentially grow and prune operations. Property 3 is a direct consequence of Properties 1 and 2 and means that any sequence of context trees obtained by a sequence of grow or prune operations can also be visited in reverse order with a sequence of grow and prune operations. These properties also suggest that combining $\oplus$ and $\ominus$ for constructing a set of transitions with positive probabilities in a proposal kernel is a good strategy in order to achieve the irreducibility condition $\kappa(\tau|\tau') > 0$ if, and only if, $\kappa(\tau'|\tau) > 0$.

We define a transition kernel κ as

$$\kappa(\tau'|\tau) = \begin{cases} \frac{1}{|\oplus(\tau,h)|}\mathbb{1}\left(\tau' \in \oplus(\tau, h)\right), & \text{if } \ominus(\tau, h) = \varnothing, \\ \frac{1}{|\ominus(\tau,h)|}\mathbb{1}\left(\tau' \in \ominus(\tau, h)\right), & \text{if } \oplus(\tau, h) = \varnothing, \\ \frac{1}{2}\frac{1}{|\oplus(\tau,h)|}\mathbb{1}\left(\tau' \in \oplus(\tau, h)\right) + \frac{1}{2}\frac{1}{|\ominus(\tau,h)|}\mathbb{1}\left(\tau' \in \ominus(\tau, h)\right), & \text{o/w}, \end{cases}$$

which allows us to propose a tree τ' in a simple two-step process. First, pick the operator to be applied to τ, $\oplus$ or $\ominus$ with probabilities $1/2$ if both lead to non-empty set of trees, otherwise pick the operation that produces a non-empty set. Then, pick τ' from $\oplus(\tau, h)$ or $\ominus(\tau, h)$ with uniform probabilities.

The idea of a proposal kernel for context trees based on growing and pruning nodes of trees was already used in Kontoyiannis et al. (2020) for a specific prior distribution h. The complete Metropolis-Hastings algorithm described in Algorithm 2, not only formally defines the construction in terms of graphs, but also extends it to accommodate arbitrary prior distributions.

Algorithm 2 – Metropolis-Hastings algorithm for sampling context trees from $\pi(\tau|\widetilde{\mathbf{z}})$ under a tree prior distribution proportional to h.

```
Set an initial tree $\tau^{(0)} \in \mathcal{T}_L$;
for $t = 1, \dots, n_{iter}$ do
    Sample an operation $\oplus$ or $\ominus$ which leads to a non-empty set when applied to
     $\tau^{(t-1)}$, with equal probabilities;
    Randomly pick a proposed tree $\tau'$ from $\oplus(\tau^{(t-1)}, h)$ or $\ominus(\tau^{(t-1)}, h)$;
    Compute the acceptance ratio
        $$A(\tau'|\tau^{(t-1)}) = \min\left(\frac{h(\tau')q(\tau', \widetilde{\mathbf{z}})}{h(\tau^{(t-1)})q(\tau^{(t-1)}, \widetilde{\mathbf{z}})} \frac{\kappa(\tau^{(t-1)}|\tau')}{\kappa(\tau'|\tau^{(t-1)})}, 1\right);$$
    Generate a random variable $U \sim \mathrm{Unif}(0, 1)$;
    if $U < A(\tau|\tau^{(t-1)})$ then
        $\tau^{(t)} \leftarrow \tau'$
    else
        $\tau^{(t)} \leftarrow \tau^{(t-1)}$
    end
end
```

4.3.4 Partial Bayes Factors and Intrinsic Bayes Factors

While $\mathrm{BF}_{a,\bar{a}}(\widetilde{\mathbf{z}})$ given by (4.8) provides a measure of the plausibility of one hypothesis with respect to another, computing this quantity may present an enormous computational cost as the sum over $\mathcal{T}_L$ involves a doubly exponential number of terms. In fact, even the normalizing constant of the tree prior distribution $\zeta(h, L)$ is intractable in the general case for moderate values L, hindering the evaluation of the model evidence $\mathcal{E}(\widetilde{\mathbf{z}}; h)$.

Previous algorithms proposed in the literature that use similar ideas to compute the marginal likelihood, like the Context Tree Weighting (CTW) algorithm from Willems et al. (1995), are not suitable for our purposes. While we are aiming to compute (4.7) for an arbitrary prior, their weighting of context trees correspond to a very specific choice of prior distributions as $h(\tau)$ such that (4.7) can be computed recursively based on the nodes of the maximal tree rather than every subtree. To overcome this difficulty, we consider the Partial Bayes Factor (PBF) described in O'Hagan (1995) as an alternative approach for model comparison.

The methodology consists of dividing the data $\widetilde{\mathbf{z}}$ into two independent chunks, $\widetilde{\mathbf{z}}_{\text{train}}$ and $\widetilde{\mathbf{z}}_{\text{test}}$ and then computing the Bayes Factor based on part of the data, $\widetilde{\mathbf{z}}_{\text{test}}$, conditioned on $\widetilde{\mathbf{z}}_{\text{train}}$ as follows,

$$\mathrm{PBF}_{a,\bar{a}}(\widetilde{\mathbf{z}}_{\text{test}}|\widetilde{\mathbf{z}}_{\text{train}}) = \frac{\sum_{\tau \in \mathcal{T}_L} \pi_a(\tau|\widetilde{\mathbf{z}}_{\text{train}}) q(\tau, \widetilde{\mathbf{z}}_{\text{test}})}{\sum_{\tau \in \mathcal{T}_L} \pi_{\bar{a}}(\tau|\widetilde{\mathbf{z}}_{\text{train}}) q(\tau, \widetilde{\mathbf{z}}_{\text{test}})}, \tag{4.10}$$

where $\pi_a(\tau|\widetilde{\mathbf{z}}_{\text{train}})$ and $\pi_{\bar{a}}(\tau|\widetilde{\mathbf{z}}_{\text{train}})$ are the posterior distributions of τ conditioned on the training data $\widetilde{\mathbf{z}}_{\text{train}}$ under the hypotheses H_a and $H_{\bar{a}}$, respectively.

Although the original goal of using PBF is to avoid undefined behaviors when evaluating the ratio of terms involving improper priors, that are replaced by the posterior distributions conditioned on the training sample, we can see that the same strategy is very useful to avoid the intractable normalizing constant from the prior distribution.

Note that, even though (4.10) still involves sums over $\mathcal{T}_L$, the terms

$$\sum_{\tau \in \mathcal{T}_L} \pi_a(\tau|\widetilde{\mathbf{z}}_{\text{train}}) q(\tau, \widetilde{\mathbf{z}}_{\text{test}}) \quad \text{and} \quad \sum_{\tau \in \mathcal{T}_L} \pi_{\bar{a}}(\tau|\widetilde{\mathbf{z}}_{\text{train}}) q(\tau, \widetilde{\mathbf{z}}_{\text{test}})$$

can be written as expected values $\mathbb{E}_{H_a}(q(\tau, \widetilde{\mathbf{z}}_{\text{test}})|\widetilde{\mathbf{z}}_{\text{train}})$ and $\mathbb{E}_{H_{\bar{a}}}(q(\tau, \widetilde{\mathbf{z}}_{\text{test}})|\widetilde{\mathbf{z}}_{\text{train}})$, which can be obtained from ergodic Markov Chains $(\tau_a^{(t)})_{t \geqslant 1}$ and $(\tau_{\bar{a}}^{(t)})_{t \geqslant 1}$with invariant measures $\pi_a(\tau|\widetilde{\mathbf{z}}_{\text{train}})$ and $\pi_{\bar{a}}(\tau|\widetilde{\mathbf{z}}_{\text{train}})$, respectively.

Therefore, we can use MCMC methods to approximate Partial Bayes Factors by sampling two Markov Chains $(\tau_a^{(t)})$ and $(\tau_{\bar{a}}^{(t)})$ and using the ratio of empirical averages instead of the expected values

$$\widehat{\text{PBF}}_{a,\bar{a}}(\widetilde{\mathbf{z}}_{\text{test}}|\widetilde{\mathbf{z}}_{\text{train}}) = \frac{\sum_{t=1}^{n_{\text{iter}}} q(\tau_a^{(t)}, \widetilde{\mathbf{z}}_{\text{test}})}{\sum_{t=1}^{n_{\text{iter}}} q(\tau_{\bar{a}}^{(t)}, \widetilde{\mathbf{z}}_{\text{test}})}. \tag{4.11}$$

To avoid the arbitrary segmentation of the dataset into train and test subsets, Berger and Pericchi (1996) proposed the Intrinsic Bayes Factor (IBF), which averages PBFs obtained using different segmentations, based on minimal training samples. Denote by $\mathcal{I}_v$ the collection of subsets of $\{1, \dots, I\}$ of size v, i.e.,

$$\mathcal{I}_v = \{\{i_1, i_2, \dots, i_v\} \subset \{1, 2, \dots, I\}\}.$$

The dataset is divided into minimal training samples, which we will consider a v-tuple of sequences $i_1^v = \{i_1, i_2, \dots, i_v\} \in \mathcal{I}_v$, denoted $\widetilde{\mathbf{z}}^{(i_1^v)}$ and the remaining $I - v$ sequences, denoted as $\widetilde{\mathbf{z}}^{(-i_1^v)}$ to be used as the test sample. For each possible subset of v sequences i_1^v, we compute the Monte Carlo approximation of the PBF in (4.11) and take either the arithmetic average to obtain the Arithmetic Intrinsic Bayes Factor (AIBF) or the geometric average for the Geometric Intrinsic Bayes Factor (GIBF).

Denoting by $(\tau_{a,i_1^v}^{(t)})$ and $(\tau_{\bar{a},i_1^v}^{(t)})$, $t = 1, \dots, n_{\text{iter}}$ the Markov Chains of context trees obtained using Algorithm 2 with target distribution $\pi_a(\tau|\mathbf{z}^{(i_1^v)})$ and $\pi_{\bar{a}}(\tau|\mathbf{z}^{(i_1^v)})$ (considering prior distribution proportional to h_a and $h_{\bar{a}}$), respectively, the AIBF and GIBF are defined as

$$\text{AIBF}_{a,\bar{a}}(\widetilde{\mathbf{z}}) = \sum_{i_1^v \in \mathcal{I}_v} \frac{1}{\binom{I}{v}} \left(\frac{\sum_{t=1}^{n_{\text{iter}}} q\left(\tau_{a,i_1^v}^{(t)}, \widetilde{\mathbf{z}}^{(-i_1^v)}\right)}{\sum_{t=1}^{n_{\text{iter}}} q\left(\tau_{\bar{a},i_1^v}^{(t)}, \widetilde{\mathbf{z}}^{(-i_1^v)}\right)} \right),$$

and

$$\text{GIBF}_{a,\bar{a}}(\widetilde{\mathbf{z}}) = \prod_{i_1^v \in \mathcal{I}_v} \left(\frac{\sum_{t=1}^{n_{\text{iter}}} q\left(\tau_{a,i_1^v}^{(t)}, \widetilde{\mathbf{z}}^{(-i_1^v)}\right)}{\sum_{t=1}^{n_{\text{iter}}} q\left(\tau_{\bar{a},i_1^v}^{(t)}, \widetilde{\mathbf{z}}^{(-i_1^v)}\right)} \right)^{\binom{I}{v}^{-1}}.$$

The complete procedure for obtaining these quantities for a given VLMC dataset is described in Algorithm 3.

Note that, while a single sequence ($v = 1$) is theoretically sufficient to identify the context tree and can be considered a minimal training sample, computing posterior distributions using small datasets may result in posterior distributions that assign very low probabilities to context trees with long branches due to smaller total counts on those longer branches. Therefore, using more sequences (higher value for v), may lead to more consistent results as the posterior distribution used in each PBF is more likely to capture long-range contexts. On the other hand, the number of PBFs to be computed is $|\mathcal{I}_v| = \binom{I}{v}$ which quickly becomes prohibitive when v increases. The choice of v is a trade-off between computational cost and the deepness of contexts to be captured by partial posterior distributions.

Algorithm 3 – Complete algorithm for computing AIBF and GIBF with MCMC approximations of Partial Bayes Factors for a dataset $\widetilde{\mathbf{z}}$ based on v-tuples.

```
for $i_1^v \in \mathcal{I}_v$ do
    Generate $\left(\tau^{(t)}_{a,i_1^v}\right), t = 1, \ldots, n_{\text{iter}}$ with the Context Tree Metropolis Hastings
      algorithm with target distribution $\pi_a(\tau|\mathbf{z}^{(i_1^v)})$;
    Generate $\left(\tau^{(t)}_{\bar{a},i_1^v}\right), t = 1, \ldots, n_{\text{iter}}$ with target distribution $\pi_{\bar{a}}(\tau|\mathbf{z}^{(i_1^v)})$;
    Compute the Partial Bayes Factor for the $v$-tuple $i_1^v$
        $$\widehat{\text{PBF}}_{a,\bar{a}}(\widetilde{\mathbf{z}}^{(-i_1^v)}|\mathbf{z}^{(i_1^v)}) = \frac{\sum_{t=1}^{n_{\text{iter}}} q(\tau^{(t)}_{a,i_1^v}, \widetilde{\mathbf{z}}^{(-i_1^v)})}{\sum_{t=1}^{n_{\text{iter}}} q(\tau^{(t)}_{\bar{a},i_1^v}, \widetilde{\mathbf{z}}^{(-i_1^v)})}.$$
end
Return the averages
    $$\text{AIBF}_{a,\bar{a}}(\widetilde{\mathbf{z}}) = \frac{1}{\binom{I}{v}} \sum_{i_1^v \in \mathcal{I}_v} \widehat{\text{PBF}}_{a,\bar{a}}(\widetilde{\mathbf{z}}^{(-i_1^v)}|\mathbf{z}^{(i_1^v)})$$
    $$\text{GIBF}_{a,\bar{a}}(\widetilde{\mathbf{z}}) = \prod_{i_1^v \in \mathcal{I}_v} \left(\widehat{\text{PBF}}_{a,\bar{a}}(\widetilde{\mathbf{z}}^{(-i_1^v)}|\mathbf{z}^{(i_1^v)})\right)^{\binom{I}{v}^{-1}}$$
```

4.4 Simulation Studies and Application

To show the strength of our method, we analyzed artificial VLMC datasets generated from two binary models models and a real one coming from the field of Linguistics.

4.4.1 Simulation for binary models

In this section, the primary goal is to examine the performance of the AIBF and GIBF for evaluating the evidence in favor of a null hypothesis of τ being *a-renewing* considering aspects of the effect of the number of independent samples, the size of each chain, and discrimination ability when similar trees are considered. We consider simulations for binary VLMC models with three different sample sizes $I = 3, 10, 25$. For each scenario we sample I chains of equal length, but three different values $T_i = 1000, 2500, 5000$. A dataset was simulated for each combination of I and T_i, resulting in 9 datasets.

The two models considered are presented in Figure 15. Model 1 has a depth equal to 6 with $a = 0$ being a renewal state, while Model 2 is a modified version with an additional branch grown from the node 01111, which is substituted by the two suffixes 001111 and 101111. Therefore, in Model 2, 0 is no longer a renewal state although both trees are very similar.

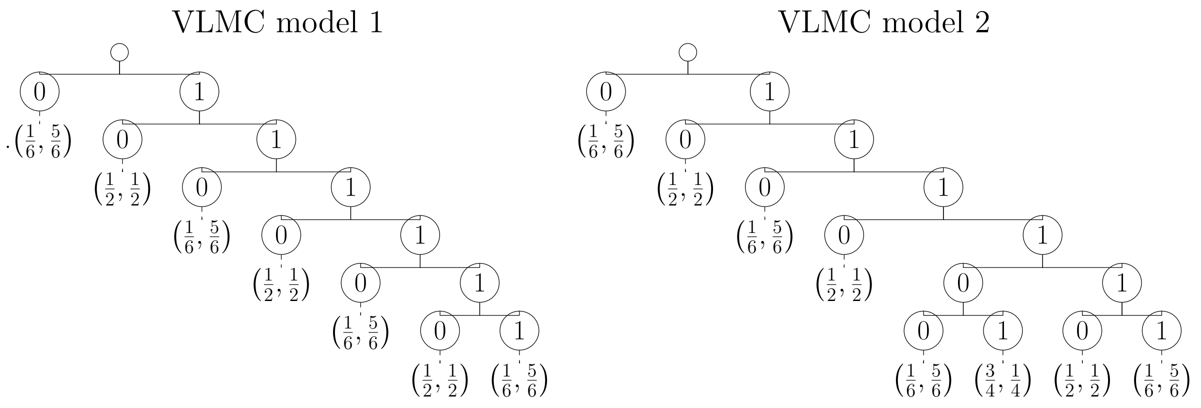

VLMC model 1
VLMC model 2
0
1
$(\frac{1}{6}, \frac{5}{6})$
$(\frac{1}{2}, \frac{1}{2})$
$(\frac{3}{4}, \frac{1}{4})$

Figure 15 – Probabilistic Context Trees for Model 1 and Model 2. The pair of values below each leaf corresponds to the transition probabilities for the suffix associated with that leaf.

For each possible renewal state $a = 0$ or $a = 1$, we compute both Intrinsic Bayes Factors (AIBF and GIBF) using Algorithm 3 considering prior distributions proportional to

$$h_a(\tau) = \mathbb{1}(\tau \in \mathcal{T}_L^a) \qquad \text{and} \qquad h_{\bar{a}}(\tau) = \mathbb{1}(\tau \in \bar{\mathcal{T}}_L^a)$$

which correspond to the uniform distribution in the space of context trees that are allowed under H_a and $H_{\bar{a}}$, respectively.

For the hyper-parameter $\boldsymbol{\alpha}$, we choose $\alpha_{\mathbf{s}k} = 0.001$ for all $\mathbf{s}$ and k, resulting in symmetrical prior distributions for the transition probabilities and a higher density for vectors that are more concentrated.

In each scenario, we considered $v = 1$ and $v = 2$ as the number of sequences to be used for the minimal training sample and ran $n_{\text{iter}} = 10^5$ Metropolis-Hastings steps for each PBF Monte Carlo approximation.

Table 3 – AIBF and GIBF computed in $\log_{10}$ scale for simulations for Model 1.

a	I	v	$T_i = 1000$ AIBF	$T_i = 1000$ GIBF	$T_i = 2500$ AIBF	$T_i = 2500$ GIBF	$T_i = 5000$ AIBF	$T_i = 5000$ GIBF
0	3	1	4.28	2.39	2.35	2.03	1.69	1.65
0	3	2	1.04	0.66	1.54	1.20	1.88	1.86
0	10	1	25.32	6.00	1.97	1.94	2.55	2.53
0	10	2	4.05	2.31	1.96	1.90	2.49	2.46
0	25	1	68.98	11.87	4.40	2.90	2.59	2.57
0	25	2	67.29	3.54	2.63	2.60	2.59	2.57
1	3	1	-46.87	-49.03	-137.09	-147.48	-285.28	-287.18
1	3	2	-17.82	-20.41	-61.80	-68.78	-135.72	-138.05
1	10	1	-248.24	-265.45	-683.99	-695.28	-1401.88	-1411.24
1	10	2	-234.44	-238.54	-596.78	-616.72	-1237.09	-1252.95
1	25	1	-688.51	-741.06	-1921.76	-1951.27	-3834.79	-3862.61
1	25	2	-705.66	-719.25	-1822.97	-1869.77	-3653.59	-3701.06

Table 4 – AIBF and GIBF computed in $\log_{10}$ scale for simulations for Model 2.

a	I	v	$T_i = 1000$ AIBF	$T_i = 1000$ GIBF	$T_i = 2500$ AIBF	$T_i = 2500$ GIBF	$T_i = 5000$ AIBF	$T_i = 5000$ GIBF
0	3	1	2.58	2.45	14.38	0.06	-7.51	-8.24
0	3	2	1.34	1.22	7.40	0.05	-2.94	-3.29
0	10	1	13.58	4.13	26.40	-7.77	70.50	-38.47
0	10	2	12.49	2.39	26.27	-2.94	65.34	-34.02
0	25	1	46.19	9.11	94.14	13.73	230.50	-95.80
0	25	2	46.49	5.54	93.36	-26.54	227.65	-111.42
1	3	1	-38.24	-40.92	-101.68	-114.08	-216.82	-233.08
1	3	2	-15.16	-18.32	-52.24	-54.20	-111.01	-113.50
1	10	1	-215.74	-222.35	-539.57	-573.34	-1062.60	-1167.87
1	10	2	-190.76	-201.17	-479.56	-509.43	-930.81	-1041.44
1	25	1	-566.23	-601.98	-1432.13	-1528.50	-2867.44	-3186.70
1	25	2	-535.09	-583.43	-1365.51	-1502.80	-2746.98	-3091.29

From Table 3, we can see that, for Model 1, both AIBF and GIBF lead to the correct decision for both renewal states tested. For $a = 0$, which was in fact a renewal state, all cases returned a value, in $\log_{10}$ scale, greater than 1 (except one equals 0.66). The discrepancy between AIBF and GIBF diminishes as T_i and I increase converging to a value around 2.5 which was considered Decisive Evidence in the Kass-Raftery scale. On the other hand, for $a = 1$, which was not a renewal state, all cases reported AIBF and GIBF in $\log_{10}$ scale smaller than -15, converging to values around -3000 when $T_i = 500$ and $I = 25$.

The results for Model 2, which includes a new pair of contexts causing 0 to be no longer a renewal state, presented in Table 4, showed a similar performance rejecting the 1-renewing hypothesis when compared to Model 1, which is expected due to the similarity of both models with respect to the short-length nodes that do not involve 1.

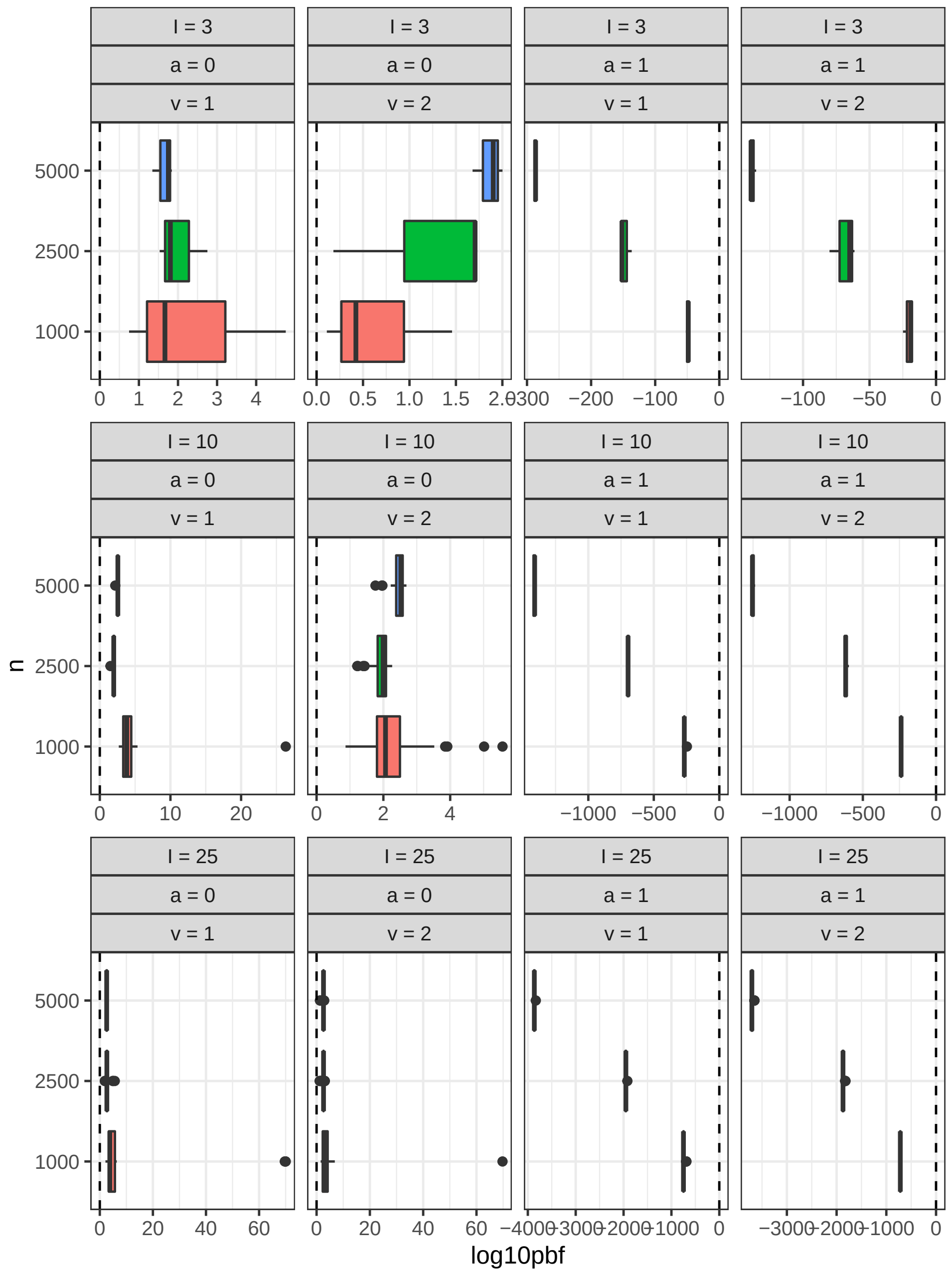


Figure 16 – Computed PBFs for Model 1 of the simulation study. Only 0 is a renewal state for this model, therefore, the panels with $a = 0$ are expected to have positive PBFs in logarithmic scale and negative values for $a = 1$.

The main difference occurred in the computed values for the 0-renewing hypothesis, where the computed GIBFs were positive in logarithmic scale for $T_i = 1000$, for all values of I and v. Moreover, we can see that for $T_i = 5000$, GIBF was negative for all scenarios whereas AIBF was strongly positive for larger values of I. For intermediate value of the sequences sizes ($T_i = 2500$), AIBF pointed to the wrong direction in all scenarios while GIBF identified the right hypothesis in three cases ($I = 10$ and $v = 1$ and 2 and $I = 25$ and $v = 2$). This suggests that for smaller sample sizes, the posterior distributions obtained from the training samples were insufficient to capture the long-range contexts 111100 and

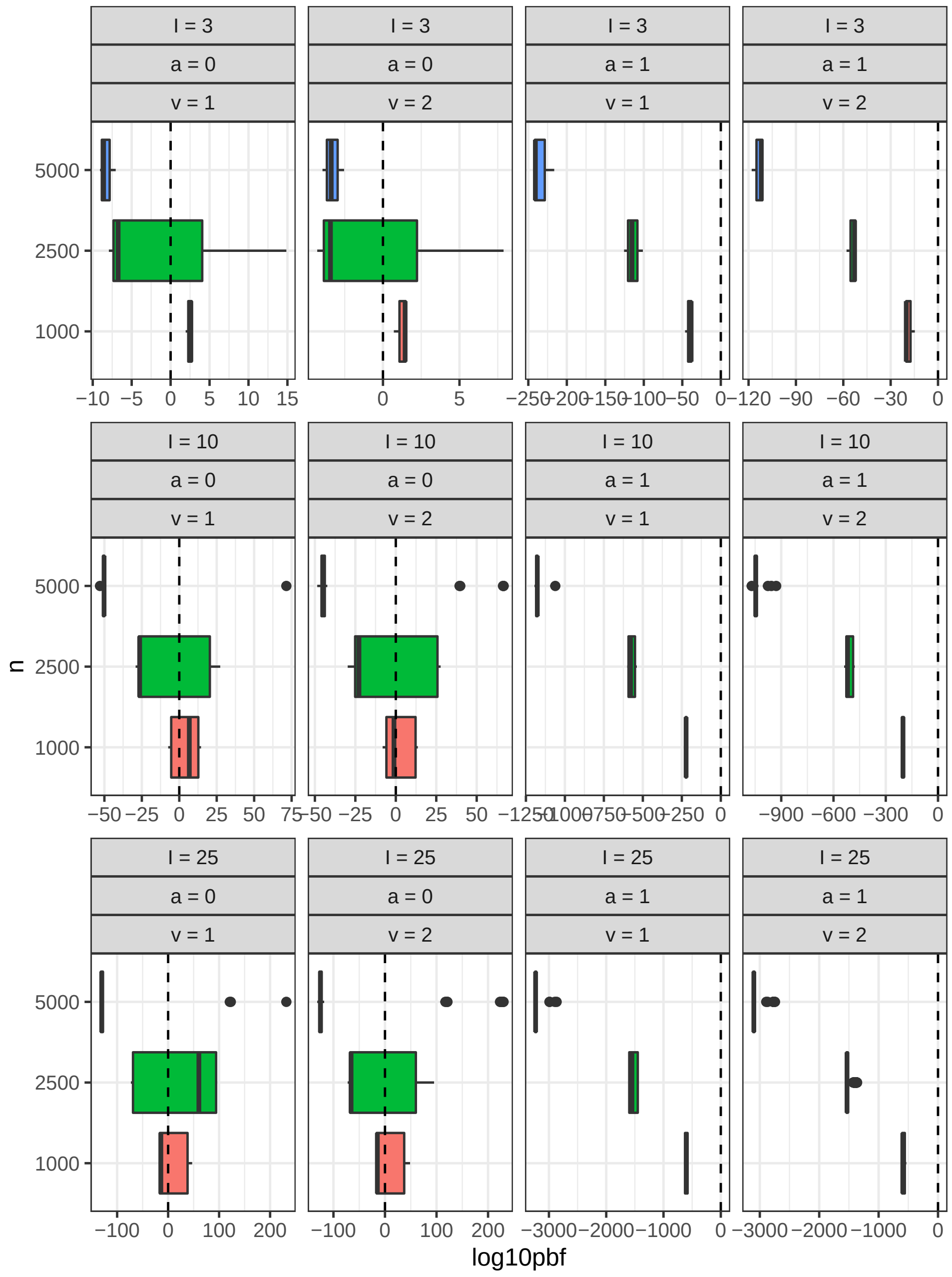


Figure 17 – Computed PBFs for Model 2 of the simulation study. Neither 0 or 1 are renewal states for this model, therefore, all panels are expected to have negative PBFs in logarithmic scale. The structure of the tree used makes the renewal state hypothesis violation for $a = 0$ to not be captured (or does not produce strong evidence against the renewal state hypothesis) for lower sample sizes.

111101 that break the renewal condition of the state 0.

Figures 16 and 17 present the empirical distributions of PBFs computed in $\log_{10}$ scale for each scenario and each model. In general, the strength of the evidence tends to be larger for $v = 1$ as more data is being used on the test set, but on the other hand, $v = 2$ leads to more stable PBFs computed. The same behavior is observed as the number of independent sequences I increases, what is expected as adding more data has an impact in the scale of the marginal likelihood function, also rescaling the Bayes Factors.

With $T_i = 2500$, $\log_{10}$ PBFs tend to be distributed around 0, with high variance, as can be observed in Figure 17, resulting in a very unstable average, leading to correct results in some of the scenarios, and incorrect ones in others. As the sample size increases, those long-range contexts are more likely to be captured by the posterior distribution. With $T_i = 5000$, we have decisive evidence that 0 is not a renewal state, except for the scenario with $I = 3$ and $v = 1$, where the value of 0.13 provides very weak evidence, and, in general, the distribution of $\log_{10}$ PBFs was highly concentrated with negative sign, except for a few outliers. AIBF was highly affected by outliers and resulted in incorrect conclusions with high evidence for some scenarios.

Note that, especially for the datasets with small samples ($T_i = 1000$), outliers with large values are observed, having great effect on the computed averages, although the conclusions are not affected. For large datasets ($T_i = 5000$), we have smaller variance in the computed PBFs among different training sets compared to scenarios with sequences of smaller sizes.

Therefore, we conclude that a decision based on GIBF leads to, at least, strong evidence in the scale from Kass and Raftery (1995) (greater than 1/2 in $\log_{10}$ scale) for the correct hypothesis in all cases where we had $v = 2$ and $T_i = 2500$ or 5000. The AIBF was not robust to the presence of outliers in the set of PBFs, leading to incorrect conclusions in the scenarios where the correct detection of the renewal state is harder task and the sample sequences are shorter. The present of outliers also suggests other functions to summarize PBFs other than geometric and arithmetic averages may be useful for avoiding having results highly influenced for the results obtained for particular test samples, like trimmed averages, removing the most extreme values from the IBF computation, or using the median PBF, which corresponds to a trimmed average trimming all but one value, as used in Charitidou et al. (2018). Context trees that break renewal state condition on long-ranged contexts require larger samples (T_i or v) in order to be captured and result in evidence against the renewal state hypothesis, while states that break renewal state condition in short contexts can be immediately identified, even with short observed sequences.

4.4.2 Application to rhythm analysis in Portuguese texts

It is known that Brazilian and European Portuguese (henceforth BP and EP) have different syntaxes. For example, Galves et al. (2005) infered that the placement of clitic pronouns in BP and EP differ in two senses, one of them being: “EP clitics, but not BP clitics, are required to be in a non-initial position with respect to some boundary". However, the question remains: are the choices of word placement related to different stress patterns preferences? This question was addressed by Galves et al. (2012) that found distinguishing rhythmic patterns for BP and EP based on written journalistic texts. The data consists of 40 BP texts and 40 EP texts randomly extracted from an encoded corpus of newspaper articles from the 1994 and 1995 editions of *Folha de São Paulo* (Brazil) and *O Público* (Portugal). Texts were encoded according on rhythmic features resulting in discrete sequences with around 2500 symbols each and are available at `http://dx.doi.org/10.1214/11-AOAS511SUPP`. After a preprocessing of the texts (removing foreign words, rewriting of symbols, dates, compound words, etc) the syllables were encoded by assigning one of four symbols according to whether or not (i) the syllable is stressed; (ii) the syllable is the beginning of a prosodic word (a lexical word (noun, verbs,...) together with the functional non-stressed words (articles, prepositions, ...) which precede or succeed it). This classification can be represented by 4 symbols. Additionally an extra symbol was assigned to encode the end of each sentence. The alphabet $\mathcal{A} = \{0, 1, 2, 3, 4\}$ was obtained as follows.

- 0 = non-stressed, non-prosodic word initial syllable;
- 1 = stressed, non-prosodic word initial syllable;
- 2 = non-stressed, prosodic word initial syllable;
- 3 = stressed, prosodic word initial syllable;
- 4 = end of each sentence.

For example, the sentence *O sol brilha forte agora.* (The sun shines bright now.) is coded as

Sentence	O	sol	bri	lha	for	te	a	go	ra	.
Code	2	1	3	0	3	0	2	1	0	4

The Smallest Maximizer Criteria proposed by Galves et al. (2012) to select the best tree for BP and EP, uses the fact that the symbol 4 appears as a renewal state to perform Bootstrap sampling. Moreover, they conclude that "the main difference between the two languages is that whereas in BP both 2 (unstressed boundary of a phonological word) and 3 (stressed boundary of a phonological word) are contexts, in EP only 3 is a context." These are exactly the type of questions to be addresses by the renewal state detection algorithm.

Due to the encoding used, the grammar of the language, and the general structure of written texts, some transitions are not possible. For example, two end of sentences (symbol 4) cannot happen consecutively, therefore, a transition from 4 to 4 is not allowed. Furthermore, there is one, and only one, stressed syllable in each prosodic word. Table 5 summarizes the allowed and prohibited one-step transitions.

Table 5 – Allowed transitions for each encoded symbol.

From/To	0	1	2	3	4
0	yes	yes	yes	yes	yes
1	yes	no	yes	yes	yes
2	yes	yes	no	no	no
3	yes	no	yes	yes	yes
4	no	no	yes	yes	no

These prohibited transition conditions are included in the model with proper modifications to the prior distribution, assigning zero probability to some context trees and forcing the probabilities related to prohibited transitions to be zero. The modifications are:

1. If a transition from k to k' is prohibited and a context $\mathbf{s}$ has k as its last symbol, we force $p_{\mathbf{s},k} = 0$ in our prior distribution. The remaining probabilities associated with allowed transitions are then a priori distributed as a Dirichlet distribution with lower dimension.

   For example, for a context 102, the only allowed transitions from 2 are to 0 and 1, we have $p_{102,2} = p_{102,3} = p_{102,4} = 0$ with prior probability 1, and the free probabilities, $(p_{102,0}, p_{102,1})$, distributed as a 2-dimensional Dirichlet distribution with hyper-parameters $(\alpha_{102,0}, \alpha_{102,1})$.

2. If $\mathbf{s} \in \tau$ includes a prohibited transition, then $n_{\mathbf{s},k}(\tilde{\mathbf{z}}) = 0$ as there will not be any occurrences of such sequence in the sample. As a consequence, these suffixes have no contribution in $q(\tau, \tilde{\mathbf{z}})$ as the term related to $\mathbf{s}$ of the product in (4.5) gets cancelled.

3. We define $\mathcal{T}_L^*$ the space of context trees that do not have prohibited transitions in inner nodes. For example, since the transition 44 is prohibited, a tree that contains a suffix 044 cannot be in $\mathcal{T}_L^*$ because the transition from 4 to 4 is not in a leaf (final node), whereas a tree in $\mathcal{T}_L$ can contain the suffix 44.

   Note that, allowing final nodes to contain a prohibited transition is necessary to keep the consistency of our definition based on **full** m-ary trees, as full tree contains the suffix 34 (allowed) if, and only if, it contains another suffix ending in 44 (prohibited), what is not a problem because this prohibited suffix will not contribute to the marginal likelihood.

For each set of sequences in BP and EP, we compute Intrinsic Bayes Factors as evidence for the five hypotheses that 0, 1, 2, 3, and 4 are renewal states. We took $L = 5$ which should be enough to cover all relevant context trees based on the results from the original paper. For prior distributions, we used the same uniform distributions as in the simulation experiment, but restricted to the trees that do not include prohibited transitions on inner nodes, i.e.,

$$h_a(\tau) = \mathbb{1}(\tau \in \mathcal{T}_5^a \cap \mathcal{T}_5^*) \qquad \text{and} \qquad h_{\bar{a}}(\tau) = \mathbb{1}(\tau \in \bar{\mathcal{T}}_5^a \cap \mathcal{T}_5^*).$$

We also set $\alpha_{\mathbf{s}k} = 0.001$ for every $\mathbf{s} \in \tau$ and $k \in \mathcal{A}$. The algorithm ran for $n_{\text{iter}} = 10^6$ iterations for computing each tree posterior distribution under each hypothesis, using $v = 2$ sequences (around 5000 symbols in each training sample) for each Partial Bayes Factor, resulting in a total of $\binom{40}{2} = 780$ posterior distributions for each hypothesis and PBFs to average.

Due to the numerical instabilities caused by outliers in the set of estimated PBFs as identified in the simulation study, especially in the AIBF, when some particular texts $\mathbf{z}^{(i)}$ are used as the training sample, we also computed a trimmed version of AIBF (and GIBF), which consists of computing the arithmetic (and geometric) average excluding the 10% lowest and 10% highest PBFs, this strategy was also used in Berger and Pericchi (1996). The empirical distributions for the estimated Partial Bayes Factors for all renewal state hypotheses after the 10% trimming are shown in Figure 18.

From the results presented in Table 6, we can see that decisive evidence was obtained when evaluating the renewal hypothesis for states 2, 3 and 4 for the BP dataset and 3 and 4 for the EP dataset which are consistent with the results from Galves et al. (2012).

Table 6 – AIBF and GIBF for the BP and EP datasets.

		AIBF		GIBF	
	a	Untrimmed	Trimmed	Untrimmed	Trimmed
BP	0	-10321.12	-10591.73	-10720.48	-10727.80
BP	1	5.88	-7.92	-7.56	-8.55
BP	2	3.49	2.00	1.64	1.64
BP	3	19.41	17.69	14.17	14.10
BP	4	7.96	6.61	6.68	6.61
EP	0	-11487.31	-11641.82	-11744.90	-11747.73
EP	1	7.87	-10.51	-11.67	-11.19
EP	2	1.52	-2.05	-2.56	-2.60
EP	3	18.37	13.49	13.40	13.37
EP	4	12.52	6.66	6.64	6.57

Finally, for completeness of the Bayesian analysis of this example, we ran the Metropolis Hastings algorithm for the entire BP and EP datasets, considering the same uniform prior distribution on every tree in $\mathcal{T}_5^*$ (no renewal state hypothesis considered, but controlling for prohibited transitions) and with $\alpha_{\mathbf{s},k} = 0.001$ for each valid pair $(\mathbf{s}, k)$. The two context trees with highest posterior probabilities for BP and EP are presented in Figure 19 and Figure 20, respectively. Leaves corresponding to contexts that contain prohibited transitions, or other contexts with no occurrences are omitted from the trees in the figures for interpretability purposes.

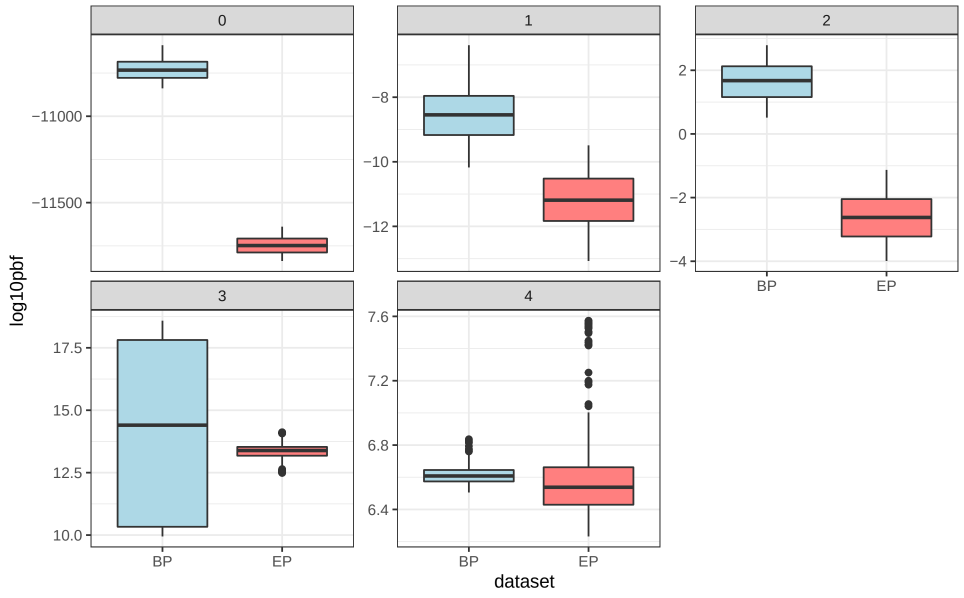


Figure 18 – Boxplots presenting the distribution of estimated Partial Bayes Factors $\widehat{\mathrm{PBF}}$ for the $\binom{40}{2}$ pairs of sequences used as training sets, in each of the BP and EP datasets after a trimming of 10% of the highest and lowest PBFs. Each pane corresponds to a hypothesis of a different state being evaluated as a renewal state.

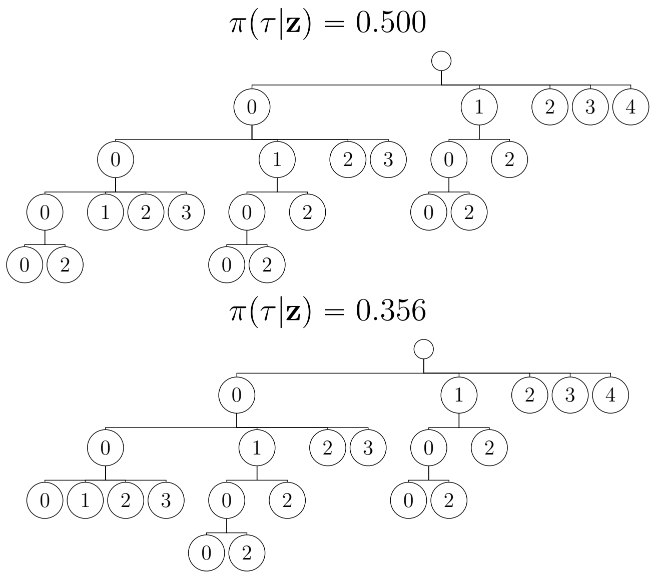

$\pi(\tau|\mathbf{z}) = 0.500$
$\pi(\tau|\mathbf{z}) = 0.356$

Figure 19 – Highest posterior probability context trees for the BP dataset.

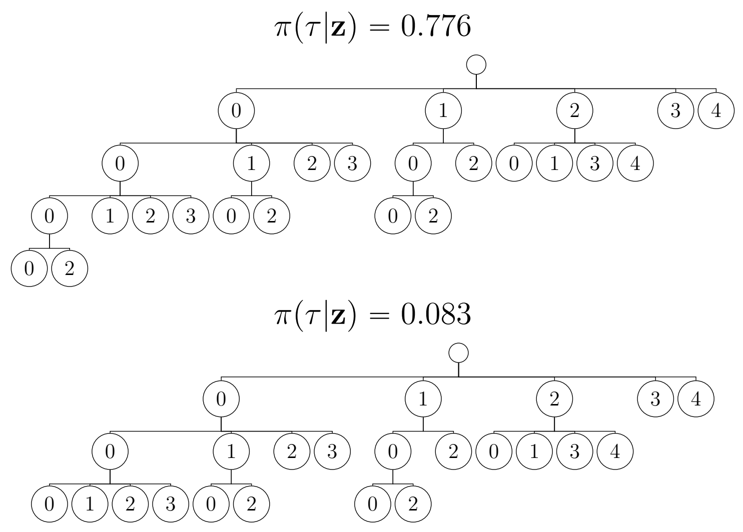

$\pi(\tau|\mathbf{z}) = 0.776$
$\pi(\tau|\mathbf{z}) = 0.083$

Figure 20 – Highest posterior probability context trees for the EP dataset.

4.5 Conclusions

We propose a Bayesian approach to Variable-Length Markov Chain models with random context trees that allows us to evaluate evidence in favor of renewal hypothesis based on a priori distributions that assign positive probability to each tree on a subset of trees that depends on the renewal state being considered. The main novelties of this work are:

1. The use of Bayes Factor to test the renewal hypothesis for VLMC models;
2. The use of Intrinsic Bayes Factor to evaluate this evidence and overcame the problem of intractable normalizing constant from the prior distribution;
3. The proposal of a Metropolis-Hastings algorithm for sampling context trees that can be performed under a tree prior distribution proportional to any arbitrary function h. This freedom allows not only to incorporate experts prior information about the possible context trees, but also to exclude forbidden trees just by assigning zero probability to them.

To show the strength of our method, we analyzed artificial datasets generated from two binary models and one example coming from the field of Linguistics. The analysis of the artificial datasets suggests that evidence becomes stronger as the number of replicates increases and/or the size of the sequences increases. However, it is possible to obtain good results with as few as 3 replicates of each chain. In the linguistics example we could observe that trimmed GIBF is more robust to possible outliers in the sample.

An R package containing functions for all the computations used in this work is available at `https://github.com/Freguglia/ibfvlmc` and R scripts used to reproduce the simulation study are available upon request.

Acknowledgments

This work was funded by Fundação de Amparo à Pesquisa do Estado de São Paulo - FAPESP grants 2017/25469-2 and 2017/10555-0, CNPq grant 304148/2020-2 and by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. We thank Helio Migon and Alexandra Schmidt for fruitful discussions.

5 Final Considerations and Future Work

This work addresses the problem of Bayesian model selection for two different models with local dependence, Markov Random Fields and Variable-Length Markov Chains, based on a common principle of treating the dependence structure of the model as a random object.

In Chapter 2, we have introduced the computational framework used for MRF models, the **mrf2d** package. The set of building blocks for inferential tools that compose the package allowed us to quickly explore algorithms and experiment with many methods proposed in the literature, using common data structures. A Bayesian extension of the package was developed, **mrf2dbayes**, to implement the methods proposed in Chapter 3, with many additional features that can be used in future works, such as the possibility of using different approximations for the Likelihood function (only the Pseudolikelihood was used in the article), and running the Metropolis-Hastings algorithm under a variety of configurations.

Chapter 3 uses the RJMCMC algorithm as a strategy for model selection in MRFs, using Pseudoposteriors. The (pseudo)posterior distribution of the Relative Position Sets, that represent interaction structures, obtained by using the efficient proposal kernel proposed for the RJMCMC, allows the use of many Bayesian analyses techniques that depend on the marginal distribution of RPSs, such as selecting the maximum a posteriori sets. We applied the proposed method to a textile image dataset to find a sparse interaction neighborhood suited for describing the features of the texture in the image.

Further research that arise from the work presented in Chapter 3 include applying adjustments to the Pseudolikelihood function when using the RJMCMC as in Bouranis et al. (2018), but the method cannot be applied directly because the optimal corrections may vary depending on the RPS, and there are typically $2^{|\mathcal{R}_{\max}|}$, leading to an excessive computational cost, since the approximations rely and maximum Likelihood estimation and Monte Carlo approximations that require sampling multiple random field configurations, and both require iterative methods to achieve. A second topic to be further explored is to find approximations for the integral of posterior distribution with respect to the coefficients, to obtain the marginal probability (up to a proportionality constant) of each RPS directly, fitting in the Monte Carlo Markov Chain Model Composition framework from Madigan et al. (1995), what would allow us to use a Metropolis-Hastings algorithm over the space of RPSs instead of using RJMCMC considering the joint space of RPSs and coefficients.

In Chapter 4 we address the problem of Bayesian inference for the context tree of a VLMC model. Using a Bayesian system with random context trees and, by using Dirichlet distributions for the transition probabilities prior distributions, we could obtain the marginal posterior distribution of context trees up to a normalizing constant. A Metropolis-Hastings algorithm on the space of context trees was used, using a similar proposal kernel from Kontoyiannis et al. (2020), but allowing any prior distribution for the context tree to be used, instead of only a prior equivalent to the weighting from the Context Tree Weighting algorithm (Willems et al., 1995). The context tree posterior sample obtained was used for producing Monte Carlo estimates of key expected values involved in the Intrinsic Bayes Factor, which was used instead of the Bayes Factor for evaluating hypothesis because it averages over the posterior distribution, that is concentrated in a few context trees, instead of the prior distribution and, therefore, is more likely cover most relevant parts of the distribution with a reasonable number of samples. The method was used to evaluate whether a particular state of an alphabet is a renewal state of the Markov Chain, which is an important property to enable many inference methods. We applied the proposed algorithm to a dataset of multiple sequences of European Portuguese and Brazilian Portuguese written texts, encoded by a 5-symbol alphabet, from Galves et al. (2012).

Future works regarding Bayesian inference for context trees include developing clustering methods for VLMC sequences, based on a notion of distance between the context tree posterior distribution of each individual sequence, what allows grouping sets of VLMC sequences that have a similar underlying context tree and obtaining theoretical results on the convergence rates of the Metropolis-Hastings algorithm for context trees. Since the algorithm is essentially a special case of a graph random walk, with a graph that is intrinsically attached to the proposal kernel considered for the Metropolis-Hastings, results related to random walks on graphs may be useful for obtaining theoretical results regarding the optimality of the proposal kernel for this scenario, or finding more efficient strategies to improve the mixing of the Markov Chain of context trees obtained.

The main contributions of this thesis are:

1. Introduce a robust and extensible computational environment, where researchers can work with a large class of Markov Random Field models for applications in many probabilistic image modeling contexts.
2. Propose a interaction structure selection methodology for Markov Random Fields, based on a Bayesian foundation that considers a random set of relative positions.
3. Define an efficient proposal kernel for the RJMCMC algorithm that is specially suited for the proposed MRF model selection method.
4. Propose a novel methodology for detection of renewal states in Markov Chains of variable-length in a Bayesian context.

5. Describe a Metropolis-Hastings algorithm to sample context trees under an arbitrary choice of prior distribution.

Bibliography

Aitkin, M. (1991). Posterior bayes factors. *Journal of the Royal Statistical Society: Series B (Methodological)*, 53(1):111–128.

Aitkin, M. (1993). Posterior bayes factor analysis for an exponential regression model. *Statistics and Computing*, 3(1):17–22.

Aitkin, M., Finch, S., Mendell, N., and Thode, H. (1996). A new test for the presence of a normal mixture distribution based on the posterior bayes factor. *Statistics and Computing*, 6(2):121–125.

Arnesen, P. and Tjelmeland, H. (2017). Prior specification of neighbourhood and interaction structure in binary markov random fields. *Statistics and Computing*, 27(3):737–756.

Atchadé, Y. F., Lartillot, N., and Robert, C. (2013). Bayesian computation for statistical models with intractable normalizing constants. *Brazilian Journal of Probability and Statistics*, 27(4):416–436.

Balding, D., Ferrari, P. A., Fraiman, R., and Sued, M. (2009). Limit theorems for sequences of random trees. *Test*, 18(2):302–315.

Bayarri, M. and Berger, J. O. (2000). P values for composite null models. *Journal of the American Statistical Association*, 95(452):1127–1142.

Berger, J. O. and Pericchi, L. R. (1996). The intrinsic bayes factor for model selection and prediction. *Journal of the American Statistical Association*, 91(433):109–122.

Besag, J. (1975). Statistical analysis of non-lattice data. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 24(3):179–195.

Boland, A., Friel, N., and Maire, F. (2018). Efficient mcmc for gibbs random fields using pre-computation. *Electronic Journal of Statistics*, 12(2):4138–4179.

Bouranis, L., Friel, N., and Maire, F. (2017). Efficient bayesian inference for exponential random graph models by correcting the pseudo-posterior distribution. *Social Networks*, 50:98–108.

Bouranis, L., Friel, N., and Maire, F. (2018). Bayesian model selection for exponential random graph models via adjusted pseudolikelihoods. *Journal of Computational and Graphical Statistics*, 27(3):516–528.

Brooks, S. P., Giudici, P., and Roberts, G. O. (2003). Efficient construction of reversible jump markov chain monte carlo proposal distributions. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 65(1):3–39.

Bühlmann, P. and Wyner, A. J. (1999). Variable length markov chains. *The Annals of Statistics*, 27(2):480–513.

Busch, J. R., Ferrari, P. A., Flesia, A. G., Fraiman, R., Grynberg, S. P., and Leonardi, F. (2009). Testing statistical hypothesis on random trees and applications to the protein classification problem. *The Annals of Applied Statistics*, 3(2):542–563.

Cabras, S., Castellanos, M. E., and Perra, S. (2015). A new minimal training sample scheme for intrinsic bayes factors in censored data. *Computational Statistics & Data Analysis*, 81:52–63.

Caimo, A. and Mira, A. (2015). Efficient computational strategies for doubly intractable problems with applications to bayesian social networks. *Statistics and Computing*, 25(1):113–125.

Charitidou, E., Fouskakis, D., and Ntzoufras, I. (2018). Objective bayesian transformation and variable selection using default bayes factors. *Statistics and Computing*, 28(3):579–594.

Chib, S. and Greenberg, E. (1995). Understanding the metropolis-hastings algorithm. *The american statistician*, 49(4):327–335.

Cross, G. R. and Jain, A. K. (1983). Markov random field texture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (1):25–39.

Csiszár, I. and Talata, Z. (2006). Consistent estimation of the basic neighborhood of markov random fields. *The Annals of Statisticxs*, 34(1):123–145.

Csiszár, I. and Talata, Z. (2006). Context tree estimation for not necessarily finite memory processes, via bic and mdl. *IEEE Transactions on Information theory*, 52(3):1007–1016.

Dimitrakakis, C. (2010). Bayesian variable order markov models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 161–168. JMLR Workshop and Conference Proceedings.

Freguglia, V. and Garcia, N. (2021). Detecting renewal states in chains of variable length via intrinsic bayes factors. *arXiv preprint arXiv:2110.07430*.

Freguglia, V. and Garcia, N. L. (2022). Inference tools for Markov random fields on lattices: The R package mrf2d. *Journal of Statistical Software*, 101(8):1–36.

Freguglia, V., Garcia, N. L., and Bicas, J. L. (2020). Hidden markov random field models applied to color homogeneity evaluation in dyed textile images. *Environmetrics*, 31(4):e2613.

Galves, A., Galves, C., Garcia, J. E., Garcia, N. L., and Leonardi, F. (2012). Context tree selection and linguistic rhythm retrieval from written texts. *The Annals of Applied Statistics*, 6(1):186–209.

Galves, A. and Löcherbach, E. (2008). Stochastic chains with memory of variable length. *Festschrift for Jorma Rissanen, Grünwald et al. (eds)*, TICSP Series 38:117–133.

Galves, C., Moraes, M. A. T., and Ribeiro, I. (2005). Syntax and morphology in the placement of clitics in european and brazilian portuguese. *Journal of Portuguese Linguistics*, 4(2).

Gelman, A., Meng, X.-L., and Stern, H. (1996). Posterior predictive assessment of model fitness via realized discrepancies. *Statistica sinica*, pages 733–760.

Geyer, C. J. and Thompson, E. A. (1992). Constrained monte carlo maximum likelihood for dependent data. *Journal of the Royal Statistical Society B*, 54(3):657–683.

Gimel'farb, G. L. (1996). Texture modeling by multiple pairwise pixel interactions. *IEEE Transactions on pattern analysis and machine intelligence*, 18(11):1110–1114.

Green, P. J. (1995). Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, 82(4):711–732.

Hassner, M. and Sklansky, J. (1981). The use of markov random fields as models of texture. In *Image Modeling*, pages 185–198. Elsevier.

Hastings, W. K. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika*.

Held, K., Kops, E. R., Krause, B. J., Wells, W. M., Kikinis, R., and Muller-Gartner, H.-W. (1997). Markov random field segmentation of brain mr images. *IEEE transactions on medical imaging*, 16(6):878–886.

Ji, C. and Seymour, L. (1996). A consistent model selection procedure for markov random fields based on penalized pseudolikelihood. *The annals of applied probability*, 6(2):423–443.

Kass, R. E. and Raftery, A. E. (1995). Bayes factors. *Journal of the american statistical association*, 90(430):773–795.

Kontoyiannis, I., Mertzanis, L., Panotopoulou, A., Papageorgiou, I., and Skoularidou, M. (2020). Bayesian context trees: Modelling and exact inference for discrete time series. *arXiv preprint arXiv:2007.14900*.

Madigan, D., York, J., and Allard, D. (1995). Bayesian graphical models for discrete data. *International Statistical Review/Revue Internationale de Statistique*, pages 215–232.

Murray, I., Ghahramani, Z., and MacKay, D. (2012). Mcmc for doubly-intractable distributions. *arXiv preprint arXiv:1206.6848*.

O'Hagan, A. (1995). Fractional bayes factors for model comparison. *Journal of the Royal Statistical Society: Series B (Methodological)*, 57(1):99–118.

Pensar, J., Nyman, H., Niiranen, J., and Corander, J. (2017). Marginal pseudo-likelihood learning of discrete markov network structures. *Bayesian analysis*, 12(4):1195–1215.

R Core Team (2020). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Ravikumar, P., Wainwright, M. J., and Lafferty, J. D. (2010). High-dimensional ising model selection using l1-regularized logistic regression. *The Annals of Statistics*, 38(3):1287–1319.

Rissanen, J. (1983). A universal data compression system. *IEEE Transactions on information theory*, 29(5):656–664.

Robbins, H. and Monro, S. (1951). A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407.

Ron, D., Singer, Y., and Tishby, N. (1996). The power of amnesia: Learning probabilistic automata with variable memory length. *Machine learning*, 25(2):117–149.

Villa, C. and Walker, S. G. (2021). An objective bayes factor with improper priors. *Computational Statistics & Data Analysis*, page 107404.

Willems, F. M., Shtarkov, Y. M., and Tjalkens, T. J. (1995). The context-tree weighting method: Basic properties. *IEEE transactions on information theory*, 41(3):653–664.

Xiong, J., Jääskinen, V., and Corander, J. (2016). Recursive learning for sparse markov models. *Bayesian analysis*, 11(1):247–263.

Zhang, Y., Brady, M., and Smith, S. (2001). Segmentation of brain MR images through a hidden markov random field model and the expectation-maximization algorithm. *IEEE transactions on medical imaging*, 20(1):45–57.