Stochastic Weather Generation with Approximate Bayesian Computation

by

Branden Olson

B.S., University of Colorado Boulder, 2016

M.S., University of Colorado Boulder, 2016

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Master of Science

Department of Applied Mathematics

2016

This thesis entitled:
Stochastic Weather Generation with Approximate Bayesian Computation
written by Branden Olson
has been approved for the Department of Applied Mathematics

_____
Prof. William Kleiber

_____
Prof. Jem Corcoran

_____
Prof. Vanja Dukic

Date _____

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Olson, Branden (M.S., Applied Mathematics)

Stochastic Weather Generation with Approximate Bayesian Computation

Thesis directed by Prof. William Kleiber

Stochastic weather generators (SWGs) are designed to create simulations of synthetic weather data and are frequently used as input into physical models throughout many scientific disciplines. While the field of SWGs is vast, the search for better methods of spatiotemporal simulation of meteorological variables persists. We propose techniques to estimate SWG parameters based on an emerging set of methods called Approximate Bayesian Computation (ABC), which bypass the evaluation of a likelihood function. In this thesis, we begin with a review of the current state of ABC methods, including their advantages, drawbacks, and variations, and then apply ABC to the simulation of daily local maximum temperature, daily local precipitation occurrence, and daily precipitation occurrence over a spatial domain.

For temperature, we model the mean and variance as following a sinusoidal pattern which depends on the previous day. A similar approach is used for precipitation, but instead use a probit regression to model the probability that it rains on a given day of the year, based on an oscillatory mean function. For spatiotemporal precipitation occurrence, we employ a thresholded Gaussian process which reduces to our methods for local occurrence. In each scenario, we identify appropriate ABC penalization criteria to produce simulations whose statistical characteristics closely resemble those of the data. For our numerical case studies, we use daily temperature and precipitation records Colorado and Iowa, collected over the course of hundreds of years.

Dedication

To my family, for their endless love, understanding, and support.

Acknowledgements

No man is an island, and the challenges and triumphs of writing a thesis have been possible only through the abundance of excellent people who have offered me their support over the course of my degree. To start, I want to thank my family for their endless confidence in my academic pursuits, and their patience with my demanding and erratic schedule. I am indebted to National Science Foundation for awarding me an EXTREEMS grant which generously funded this research in its entirety. Thank you to Dr. Anne Dougherty, who has been a supreme advisor, professor, and mentor, and a critical contributor to my undergraduate and graduate academic careers. I extend my gratitude to Dr. Juan Restrepo, who introduced me to academic research and exposed me to the delights of an inquisitive frame of mind. I owe much of my interest in statistical theory to Dr. Jem Corcoran due to her captivating style of teaching and enthusiasm of the subject. Dr. Vanja Dukic followed suit with her instruction, catalyzing and refining my statistical modeling abilities and providing sound academic and professional counseling. Finally, I owe the greatest appreciation to my advisor, Dr. Will Kleiber. Through his example, he has given me the willpower to aim for the stars, as well as the resilience to keep my head up when things go awry. This thesis would not have been possible without his supervision and I am fortunate to have worked with a professor of his stature and geniality. His perpetual optimism and support have been an absolute inspiration to me, and I hope to continue to work together as colleagues and friends for years to come.

Contents

Chapter

1 Introduction 1

2 Approximate Bayesian Computation 5

2.1 History of ABC 5

2.2 Bayes's Theorem 6

2.3 Bayesian Inference 7

2.4 Approximate Bayesian Computation 8

2.5 ABC Using Markov Chain Monte Carlo Techniques 10

2.6 Further Extensions of ABC 16

3 Simulation of Local Daily Maximum Temperature with ABC 17

3.1 Introduction 17

3.2 Statistical Model 19

3.3 Metrics for Estimating Temperature 23

3.4 Numerical Results for Glenwood Springs, Colorado 24

3.5 Discussion 28

4 Simulation of Local Daily Precipitation Occurrence with ABC 40

4.1 Introduction 40

4.2 Statistical Model 40

4.3 Metrics for Estimating Precipitation Occurrence 42
4.4 An Analytic Solution of the True Posterior 44
4.5 Numerical Results for Bonny Dam, Colorado 47

5 Simulation of Daily Precipitation Occurrence across a Spatial Domain with ABC 56
5.1 Introduction 56
5.2 Statistical Model 57
5.3 Spatial Estimation with Variograms 59
5.4 Gaussian Process Simulation 61
5.5 Metrics for Estimating Spatiotemporal Precipitation Occurrence 63
5.6 Numerical Results for the State of Iowa 66
5.7 Discussion 69

Bibliography 76

Figures

Figure

3.1 Daily maximum temperature in Glenwood, Colorado over the span of 3 years. . . . 18

3.2 Daily empirical mean of maximum temperature in Glenwood, Colorado. 20

3.3 Daily empirical standard deviation of maximum temperature in Glenwood, Colorado. 20

3.4 Daily empirical standard deviation data for the Campo 7, Colorado site along with its MLE sinusoidal fit. 22

3.5 Prior and approximate posterior distributions for β 27

3.6 Prior and approximate posterior distributions for α 27

3.7 Monthly mean maximum temperature in degrees Celsius for observed data as well as for 3 simulations parametrized by the empirical posterior means 29

3.8 Monthly standard deviation of maximum temperature in degrees Celsius for observed data as well as for 3 simulations parametrized by the empirical posterior means 29

3.9 Boxplot of simulated average monthly maximum temperature values in degrees Celsius for β as well as those for observed values 30

3.10 Boxplot of simulated average monthly maximum temperature standard deviations in degrees Celsius for α as well as observed values 31

3.11 Counts of hot spells over 30 degrees Celsius for observations and simulated data for Glenwood Springs, CO. 32

3.12 Counts of hot spells over 35 degrees Celsius for observations and simulated data for Glenwood Springs, CO. 33
3.13 Counts of hot spells over 40 degrees Celsius for observations and simulated data for Glenwood Springs, CO. 34
3.14 Counts of cold spells below 5 degrees Celsius for observations and simulated data for Glenwood Springs, CO. 35
3.15 Counts of cold spells below 0 degrees Celsius for observations and simulated data for Glenwood Springs, CO. 36
3.16 Counts of cold spells below -5 degrees Celsius for observations and simulated data for Glenwood Springs, CO. 37
3.17 Realization of our SWG for daily maximum temperature for Glenwood Springs from January 10, 1988 to January 10, 1991. Note the masking of missing data into our simulations. 38

4.1 The prior, true posterior (via MCMC sampling), and approximate posterior densities for each mean parameter β_i 49
4.2 The mean function of precipitation occurrence for the ABC and true posterior estimates as well as the observed empirical probability of precipitation for Bonny Dam, Colorado. 50
4.3 The empirical probability of precipitation for the observations and ABC simulation for Bonny Dam, Colorado. 51
4.4 The empirical standard deviation of precipitation (in mm) for the ABC simulation, true posterior simulation, and observations for Bonny Dam, Colorado. 52
4.5 Wet spell counts and the logarithm of wet spell counts by spell length for the observations and ABC simulation. 54
4.6 Dry spell counts and the logarithm of dry spell counts by spell length for the observations and ABC simulation. 55

5.1 The prior and ABC posterior densities for each parameter β_i via (5.3). 70

5.2 The empirical probability of precipitation by day averaged over each month and over all of the 22 locations in Iowa (represented by the box plots), as well as the mean daily probability of precipitation given by our $\boldsymbol{\beta}_{\text{ABC-MCMC}}$ estimate (represented by the solid blue line). 71

5.3 The prior and ABC posterior densities for parameters α_i from (5.4), as well as the nugget effect τ^2. Note that the density represents the square root of the nugget effect, $\tau = \sqrt{\tau^2}$, an artifact of our method of implementation. 72

5.4 The aggregate variogram $\widehat{\Gamma}(m)$ of our simulated thresholded Gaussian process for each month for the state of Iowa (represented by the box plots), as well as the observed aggregate variogram for the state of Iowa (represented by the solid blue lines). 73

5.5 Sample daily precipitation occurrence simulation over Iowa from January 20 through January 31. 74

Chapter 1

Introduction

Our obsession with the weather is not a new development in human history. From our stone age ancestors to the modern dwellers of the city, human beings remain at the mercy of the weather. Perhaps we want to know the forecast for our upcoming holiday picnic, or which districts should take shelter from an oncoming tornado. Indeed, it's a fact of life, accepted at an early age, that the weather dictates our agendas. Consequently, it's no surprise that many academic and professional disciplines require models of the weather that incorporate various geospatial quantities over spatiotemporal domains. What's more, there are different models of weather depending on the intentions of the modeler. A temperature forecast aims to predict, while analyses of previous temperature recordings may aim to estimate the temperature at a time in the past.

A multitude of scientific models require sequences of weather data, arbitrary in length and sometimes in breadth, as input. These fields include, but are not limited to, hydrology [41, 59], ecology [21, 71], meteorology [46], agriculture [80], and climate impact assessment [67]. It turns out that the deficiencies inherent in ground-based meteorological data observations motivate the need for synthetic weather data for these models. The observational data are limited in their length and spatial coverage, and are often polluted with missing or incomplete values. Stochastic weather generators (SWGs) are infinite time series of synthetic weather data that are built to exhibit statistical similarity to the observed data. SWGs can model a single site or incorporate a more general spatial domain. SWGs for daily weather sequences (as opposed to hourly, monthly, etc.) are the most common due to their relative simplicity and feasibility, as well as the availability of

records collected on a day-to-day basis [82]. SWGs are usually categorized into two approaches: model-based and empirical. Model-based SWGs attempt use a statistical model of the meteorological quantity to produce values which behave like the weather data [57, 59]. Empirical SWGs use techniques that sample data that has already been observed and constructs them according to particular guidelines [42, 58]. SWGs traditionally produce simulations at spatial locations from which the observed data were taken, but there have been recent developments in producing gridded simulations that are spatially consistent across a domain [39,81]. A significant feature of using SWGs is their capacity for uncertainty quantification over the domain under consideration [36,47].

Consistent spatiotemporal simulation constitutes a vast and prolific area of research, with the focus usually on nonstationarity and complex terrain. Perhaps the most prominent question in constructing a geostatistical model is whether the underlying field is univariate or multivariate. There are numerous statistical models for nonstationary univariate fields, with approaches involving periodograms and spectral methods [23], lognormal and moving-window kriging [27], process-convolution approaches [29], piecewise Gaussian processes [37], nonstationary covariance functions [51, 54], nonparametric estimation techniques [65], and weighted mixture state space frameworks [72]. Despite the wealth of tools for univariate modeling, many SWGs require multivariate spatiotemporal simulations. There are fewer statistical models in this regard, but some approaches include linears model of coregionalization [26, 34], cross-covariance functions [38], and dynamic linear models [68]. Precipitation simulation and stochastic interpolation is at the forefront of the literature due to its many challenges, including the discrete-continuous nature of precipitation occurrence and intensity [1,2,7,20,31,66]. Still, modeling temperature has setbacks of its own, especially when considering a complex terrain [51]. The difficulties of incorporating both temperature and precipitation into a unified model makes these kinds of models less common, although it has been done by a few authors, such as Gelfand et. al. [25]. There is also significant interest in the deterministic rather than stochastic interpolation of meteorological variables. Some key contributions include Daly et. al. [10], Hijmans et. al. [30], Hutchinson [32], Legates and Willmott [43], Price et. al. [55], Running et. al. [62], Thornton et. al. [76], and Willmott and

Matsuura [83].

As previously mentioned, SWGs for precipitation, known as stochastic precipitation generators (SPGs), comprise a substantial chunk of the field of SWGs. The stochastic model is often split into a component for binary precipitation occurrence (i.e. there was or was not precipitation on a given day) and a component for precipitation intensity (i.e., it rained this much on a given day). For occurrence, SPGs usually employ a Markov chain to capture the temporal dependence [35]. For intensities, modelers customarily turn to an exponential or gamma distribution, or some amalgamation thereof [59, 70, 84]. More recently, SPGs have played a key role in statistical downscaling [45, 79]. Interest has shifted from local precipitation, that is, precipitation at a single location, to spatiotemporally correlated precipitation across a spatial domain. The troublesome nature of local precipitation, including its variability and intermittency, propagates to the spatial case, making spatiotemporal precipitation simulation a daunting task. Regardless, there is a profusion of approaches to spatiotemporal precipitation modeling and simulation. Hidden Markov models have been used to model occurrence [31] as well as intensity [1, 14]. Other methods include nearest-neighbors resampling [3,8,58], generalized chain-dependent processes [86,87], power transformation to normality [66, 85], artificial neural network methods [11], and copula-based approaches [9]. Current approaches to this problem typically seek the assistance of latent multivariate normals, sometimes including a transformation, to generate the occurrence/intensity values over a spatial domain. This approach was catalyzed by Wilks [78], and evolved in terms of efficiency and sophistication by Brissette et. al. [6] and Thompson et. al. [75]. The modern popularity of Wilks' approach can be attributed by a comparative study of his approach with the resampling and hidden Markov model approaches, which found the one of Wilks to best capture spatial dependence as well as local temporal dependence of precipitation [47]. As with temperature, the simulation of precipitation over the entirety of the domain beyond just the sites which supply observational data is of crucial importance and yet absent in most SPGs.

While the literature for SWGs is extensive, the search for new means of their estimation persists due to the many inherent challenges they induce. Approximate Bayesian Computation

(ABC) is a set of general techniques that approximate the posterior densities of model parameters of interest to an arbitrary level of tolerance. These algorithms yield accurately approximated posteriors while sidestepping the often problematic evaluation of analytic likelihood functions [73]. In this thesis, we argue for the utilization of ABC as a serious toolset in the estimation of SWGs. We first provide a brief exposition of ABC and its most common implementations, and then demonstrate its potential through several case studies. Specifically, we will present models for daily local maximum temperature, daily local precipitation occurrence, and daily precipitation occurrence over a spatial domain, and propose ABC procedures to estimate the parameters of these models. When implementing ABC, one needs to specify a metric in which to compare the observed data with the simulations from the model. These metrics are often nontrivial, and depend very much on the problem in consideration. Our focus will be on identifying appropriate metrics for each of the three aforementioned weather scenarios. We assess the validity of our techniques through numerical case studies of available data from the states of Colorado and Iowa.

Presently, we are unaware of any techniques concerning the application of ABC to SWGs. There has been recent interest in using ABC for hydrological purposes, starting with Nott et. al., who compare generalized likelihood uncertainty estimation (GLUE) with ABC in a case study of rainfail-runoff modeling [50]. Sadegh and Vrugt follow suit with some further exploration of ABC, again with rainfall-runoff applications in mind [63, 64]. Otherwise, the utilization of ABC remains chiefly in the field of biology due to its origins in population genetics research. Nonetheless, we contend that ABC constitutes an excellent toolset not only for the creation of reliable SWGs, but as a promising tool that should be exploited across the many branches of statistics.

Chapter 2

Approximate Bayesian Computation

The methods derived in this thesis belong to a category of techniques collectively known as Approximate Bayesian Computation, or simply ABC. Consequently, a thorough examination of the background, derivation, and implementation of ABC will serve as a necessary prelude to our methods. While the ideas behind ABC are intuitive, and mathematically well-founded, we will analyze them with caution to address their inherent subtleties and pitfalls. We begin our discussion with Bayes's Theorem, a simple but powerful result derived from the concept of conditional probability. We review how this theorem forms the basis of the field of Bayesian inference as an alternative to the classical frequentist approach. From here we will be in position to motivate ABC and discuss some of its variations.

2.1 History of ABC

The central ideas behind ABC were first introduced in a 1984 essay by Donald Rubin, in which he puts forth a thought experiment to illustrate how Bayesian ideas should be interpreted and put into effect [61, 73]. He argued for the emphasis on approximate posterior distributions so that more complex models could be considered. The first algorithm to incorporate ideas that comprise contemporary ABC was proposed by Tavare et. al., independent of the work of Rubin, concerning the chronology of common ancestors of humans based on homologous DNA sequence samples [74]. They describe a rejection algorithm to approximate the likelihood of coalescence times from available sequence data approximately distributed as Gamma random variables. This

rejection scheme quickly infiltrated the field of population genetics as other researchers found modifications and extensions of these ideas, forming the basis of the techniques which are in use today [22, 56, 77]. Beaumont et. al. were the ones to officially dub these techniques "Approximate Bayesian Computation", and from there the name stuck [5].

2.2 Bayes's Theorem

Recall the definition of conditional probability of an event A given some other event B has been observed, denoted $\mathbf{P}[A|B]$:

$$\mathbf{P}[A|B] =_{\text{def}} \frac{\mathbf{P}[A \cap B]}{\mathbf{P}[B]}.$$

In general, conditional probability is not commutative, so that $\mathbf{P}[A|B]$ does not necessarily equal $\mathbf{P}[B|A]$. However, this can be done by including the proper scaling factors to keep $\mathbf{P}[\cdot|\cdot]$ a valid probability. This is where Bayes's Theorem comes to the rescue, which states in that

$$\mathbf{P}[A|B] = \frac{\mathbf{P}[B|A]\,\mathbf{P}[A]}{\mathbf{P}[B]}. \tag{2.1}$$

This provides us with a valid means of reversing the arguments in the conditional probability operator, which is useful if you have an expression for one, but not the other. We can also formulate this in terms of probability density functions:

$$f_{X|Y}(x|y) = \frac{f_{Y|X}(y|x) f_X(x)}{f_Y(y)}. \tag{2.2}$$

In fact, it is easily verified that

$$f_{\mathbf{X}|\mathbf{Y}}(\mathbf{x}|\mathbf{y}) = \frac{f_{\mathbf{Y}|\mathbf{X}}(\mathbf{y}|\mathbf{x}) f_{\mathbf{X}}(\mathbf{x})}{f_{\mathbf{Y}}(\mathbf{y})}, \tag{2.3}$$

which is the extension of (2.1) for joint probability density functions of arbitrary random vectors $\mathbf{X} = (X_1, \ldots, X_n)$ and $\mathbf{Y} = (Y_1, \ldots, Y_m)$. The multivariate case given by (2.3) will be the one in which we are interested for this section, and for the remainder of this thesis. We will see that this simple theorem yields profound implications, paving the way for the Bayesian interpretation of statistics as an alternative to the traditional frequentist approach.

2.3 Bayesian Inference

Let p be a probability of some event of interest. The frequentist defines p as the limit of its frequency in an arbitrarily large number of trials. This approach is the traditional one, and stood alone before the Bayesian paradigm gained traction with statisticians. Bayesian inference diverges from the traditional frequentist approach in that, instead of assuming that p is a true limiting value, it is assumed to contain uncertainty and is assigned a probability distribution representing a degree of belief by the statistician. This is not the only interpretation of Bayesian inference, but will be the one assumed for the rest of this thesis.

With Bayesian inference, one is often interested in the probability distribution of certain parameters $\boldsymbol{\theta}$ given some data $\mathbf{X}$. Let us denote this probability as the posterior distribution, written $f_{\boldsymbol{\Theta}|\mathbf{X}}(\boldsymbol{\theta}|\mathbf{x})$, or simply $f(\boldsymbol{\theta}|\mathbf{x})$ for convenience. By (2.2), we see that

$$\begin{aligned} f(\boldsymbol{\theta}|\mathbf{x}) &= \frac{f_{\mathbf{X}|\boldsymbol{\Theta}}(\mathbf{x}|\boldsymbol{\theta})\, f_{\boldsymbol{\Theta}}(\boldsymbol{\theta})}{f_{\mathbf{X}}(\mathbf{x})} \\ &\propto f_{\mathbf{X}|\boldsymbol{\Theta}}(\mathbf{x}|\boldsymbol{\theta})\, f_{\boldsymbol{\Theta}}(\boldsymbol{\theta}), \end{aligned}$$

where we stow away the normalizing constant $f_{\mathbf{X}}(\mathbf{x})$ since it is not a function of the parameters $\boldsymbol{\theta}$. The right hand factor $f(\mathbf{x}|\boldsymbol{\theta})$ turns out to be the familiar likelihood function L of parameters $\boldsymbol{\theta}$ given $\mathbf{x}$, as $L(\boldsymbol{\theta}|\mathbf{x}) = f_{\mathbf{X}|\boldsymbol{\Theta}}(\mathbf{x}|\boldsymbol{\theta}) = f(\mathbf{x}|\boldsymbol{\theta})$. Moreover, following the convention of the literature, we rewrite $f_{\boldsymbol{\Theta}}(\boldsymbol{\theta})$ as $\pi(\boldsymbol{\theta})$, called the prior density of $\boldsymbol{\theta}$. We are left with

$$f(\boldsymbol{\theta}|\mathbf{x}) \propto L(\boldsymbol{\theta}|\mathbf{x})\, \pi(\boldsymbol{\theta}). \tag{2.4}$$

In English, (2.4) states that the posterior distribution is proportional to the likelihood function times the prior density. This gives us an intuitive interpretation of the posterior distribution as the product of our initial beliefs about the parameters, represented by $\pi(\boldsymbol{\theta})$, and information about the parameters obtained from empirical evidence, represented by $L(\boldsymbol{\theta}|\mathbf{x})$. For our purposes, we will assume independence of the parameters $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_k)$, so that the joint prior density π is the product of the independent marginal densities π_i :

$$\pi(\boldsymbol{\theta}) = \prod_{i=1}^{k} \pi_i(\theta_i).$$

2.4 Approximate Bayesian Computation

Perhaps the most important problem in Bayesian statistics is determining the posterior distribution of the parameters $\boldsymbol{\theta}$ given some context-dependent prior distribution. If the likelihood is known, then one can simply refer to (2.4) to get a closed-form analytical formula directly. Unfortunately, it is rarely the case that a closed-form likelihood solution exists. Moreover, in the case that the closed-form posterior is available, it can be intractable to evaluate as it depends on the entire dataset $\mathbf{x}$, which is frequently large in practice.

To address this problem, Marjoram et. al. [44] developed a set of methods to sample a posterior distribution without the need to evaluate a likelihood function. Suppose that we generate some data $\mathcal{D}$ from a model $\mathcal{M}$ which relies on some parameters $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_n)$, so that $\mathcal{M} = \mathcal{M}(\boldsymbol{\theta})$. Let us start with the simplest case and assume that the data are discrete. If the likelihood $L(\boldsymbol{\theta}|\mathbf{x})$ is known, we can use algorithm 1, which simply accepts candidate parameters with probability exactly equal to its likelihood.

Algorithm 1: Generating samples from $L(\boldsymbol{\theta}|\mathcal{D})$ for discrete $\mathcal{D}$ and $L(\mathcal{D}|\boldsymbol{\theta})$ known

Output: A vector V of N samples from the posterior $L(\boldsymbol{\theta}|\mathcal{D})$.
$V \leftarrow \{\}$
while $|V| < N$ **do**
  generate $\boldsymbol{\theta} \sim \pi(\cdot)$
  append $\boldsymbol{\theta}$ to V with probability $p = L(\mathcal{D}|\boldsymbol{\theta})$.
end
return V

This algorithm is a basic implementation of the acceptance-rejection (A-R) scheme. It is straightforward to show that if $\exists K > 0 : f(\boldsymbol{\theta}|\mathcal{D}) \leq Kg(\boldsymbol{\theta}|\mathcal{D})$ for some function $g(\boldsymbol{\theta}|\mathcal{D})$, then each $\boldsymbol{\theta} \in V$ is a sample from the posterior $f(\boldsymbol{\theta}|\mathcal{D})$ [60]. However, as we mentioned, the likelihood function is often unavailable or computationally troublesome, yet algorithm 1 relies on knowledge of the likelihood in order to generate the acceptance probability. In the case of an unavailable likelihood, we can sidestep its necessity by simply checking if the simulated data $\mathcal{D}'$ matches the generated $\mathcal{D}$ from $\mathcal{M}$; If these data match exactly, then they are certainly a sample from the

posterior, and so we accept them. This gives rise to the following modification of algorithm 1. While attractive for easily simulable models $\mathcal{M}$ and applicable to a number statistical problems,

Algorithm 2: Generating samples from $L(\boldsymbol{\theta}|\mathcal{D})$ for discrete $\mathcal{D}$ and $L(\mathcal{D}|\boldsymbol{\theta})$ unavailable

Input : Model $\mathcal{M}$, data $\mathcal{D}$
Output: A vector V of N samples from the posterior $L(\boldsymbol{\theta}|\mathcal{D})$
$V \leftarrow \{\}$
while $|V| < N$ **do**
  generate $\boldsymbol{\theta} \sim \pi(\cdot)$
  simulate $\mathcal{D}'$ from $\mathcal{M}$ parametrized by $\boldsymbol{\theta}$
  if $\mathcal{D}' = \mathcal{D}$ **then**
    append $\boldsymbol{\theta}$ to V
  end
end
return V

algorithm 2 is often an unrealistic approach to the problem of sampling from the posterior, as the chance of simulating data that exactly match the observations can be extremely low. In fact, if the data are continuous rather than discrete, $\mathbf{P}\left[\mathcal{D}' = \mathcal{D}\right] = 0$, deeming this algorithm practically ineffective. This issue motivates a modification to the algorithm that can produce samples from a continuous, or otherwise problematic, dataset, while retaining an acceptable level of accuracy.

So, we want an algorithm that accepts simulated $\mathcal{D}'$ that are close enough to the observations $\mathcal{D}$, based on some notion of closeness. Translating this to mathematics, we can employ the following method: accept $\mathcal{D}'$ if $\varrho\left(\mathcal{D}, \mathcal{D}'\right) < \varepsilon$, for some proposed dissimilarity metric $\varrho(\cdot, \cdot)$ and some user-defined tolerance $\varepsilon > 0$. This is laid out in more detail in algorithm 3. The obvious drawback of this method is that the samples are no longer strictly from the posterior $f\left(\boldsymbol{\theta}|\mathcal{D}\right)$, but instead from $f\left(\boldsymbol{\theta}\middle|\varrho\left(\mathcal{D}, \mathcal{D}'\right) < \varepsilon\right)$. Furthermore, it is up to the statistician to determine appropriate $\varrho(\cdot, \cdot)$ and ε, which will be highly dependent on the context, and a major component of our impending analyses. An important thing to note is that $\varepsilon = 0$ yields samples from the true posterior, whereas sending $\varepsilon \to \infty$ yields samples from the prior. In a computational perspective, a higher ε will yield more samples, but lose accuracy, while a low ε will increase accuracy, but become more intractable. Hence, the choice of ε represents a tradeoff between accuracy and feasibility, and can be tweaked to yield a reasonable acceptance rate of parameters.

Algorithm 3: Generating samples distributed from $f\left(\boldsymbol{\theta}|\varrho\left(\mathcal{D},\mathcal{D}'\right)<\varepsilon\right)$

Input : Model $\mathcal{M}$, data $\mathcal{D}$, metric $\varrho(\cdot,\cdot)$, tolerance $\varepsilon>0$
Output: A vector V of samples approximately distributed from $f(\boldsymbol{\theta}|\mathcal{D})$
$V \leftarrow \{\}$
while $|V|<N$ **do**
  generate $\boldsymbol{\theta}\sim\pi(\cdot)$
  simulate $\mathcal{D}'$ from $\mathcal{M}$ parametrized by $\boldsymbol{\theta}$
  if $\varrho\left(\mathcal{D},\mathcal{D}'\right)<\varepsilon$ **then**
    append $\boldsymbol{\theta}$ to V
  end
end
return V

We can attempt to gain more efficiency via yet another modification. As the size of $\mathcal{D}$ gets large, the comparison $\varrho\left(\mathcal{D},\mathcal{D}'\right)$ can become inefficient. If we instead consider summary statistics $\mathbf{S}=\mathbf{s}\left(\mathcal{D}\right)$ and $\mathbf{S}'=\mathbf{s}\left(\mathcal{D}'\right)$, we can cut back a significant amount of computation while retaining a reasonable level of accuracy. If $\mathbf{S}$ is sufficient for $\mathcal{D}$, then we have $f(\boldsymbol{\theta}|\mathcal{D})=f(\boldsymbol{\theta}|\mathbf{S})\ \forall\boldsymbol{\theta}\in\boldsymbol{\Theta}$. In light of algorithm 3, we would have $f\left(\boldsymbol{\theta}|\varrho\left(\mathcal{D},\mathcal{D}'\right)<\varepsilon\right)=f\left(\boldsymbol{\theta}|\varrho\left(\mathbf{S},\mathbf{S}'\right)<\varepsilon\right)$. Even if the statistics are not sufficient, if they describe a quality of the data that is important in the context of the problem, then these statistics should still act as acceptable summaries of the data. This gives rise to algorithm 4.

2.5 ABC Using Markov Chain Monte Carlo Techniques

So far, we have pruned our way towards a reasonably efficient ABC algorithm to obtain approximate samples from the posterior distribution. However, algorithm 4 has its shortcomings. For example, if the prior distribution contains large intervals of values from which to sample parameters, the chances of accepting can be extremely low. This effect is compounded heavily as the number of parameters increases. Additionally, the acceptance-rejection algorithm generates independent samples with each step, so each iteration relies on identifying a desirable $\boldsymbol{\theta}$ from $\pi(\cdot)$ without any other information. In other words, when parameters satisfy the $\varrho(\cdot,\cdot)$ criterion, their discovery does not inform future samples from the prior.

Algorithm 4: Generating samples distributed from $f\left(\boldsymbol{\theta}|\varrho\left(\mathbf{s}(\mathcal{D}),\mathbf{s}(\mathcal{D}')\right)<\varepsilon\right)$

Input : Model $\mathcal{M}$, summary function $\mathbf{s}(\cdot)$, metric $\varrho(\cdot,\cdot)$, tolerance $\varepsilon>0$
Output: A vector V of samples approximately distributed from $f(\boldsymbol{\theta}|\mathcal{D})$
$\mathbf{S}\leftarrow\mathbf{s}\left(\mathcal{D}\right)$
$V\leftarrow\{\}$
while $|V|<N$ **do**
  generate $\boldsymbol{\theta}\sim\pi(\cdot)$
  simulate $\mathcal{D}'$ from $\mathcal{M}$ parametrized by $\boldsymbol{\theta}$
  $\mathbf{S}'\leftarrow\mathbf{s}\left(\mathcal{D}'\right)$
  if $\varrho\left(\mathbf{S},\mathbf{S}'\right)<\varepsilon$ **then**
    append $\boldsymbol{\theta}$ to V
  end
end
return V

Thankfully, this issue has been rectified by introduction of the Metropolis-Hastings (M-H) algorithm. M-H is a Markov chain Monte Carlo method used to sample from a probability distribution which can be fine-tuned to yield a high acceptance rate of proposed values [28, 48]. Given target density $\varphi(\boldsymbol{\theta})$, M-H uses the most recent accepted value $\boldsymbol{\theta}$ to generate a candidate value $\boldsymbol{\theta}'$ using some candidate-generating density $q(\boldsymbol{\theta}, \boldsymbol{\theta}')$. Thus, candidate values will be close to the most recent acceptance, and will likely have a higher chance of being accepted themselves than if they were sampled without the influence of the recently accepted $\boldsymbol{\theta}$. The details of the M-H algorithm are laid out in algorithm 5. One essential feature of 5 is the flexibility in choosing $q(\cdot, \cdot)$. In partic-

Algorithm 5: The Metropolis-Hastings algorithm

Input : Arbitrary initial value $\boldsymbol{\theta}_0$, candidate-generating density $q(\cdot, \cdot)$
Output: A vector V of samples approximately distributed from $\varphi(\cdot)$
$V \leftarrow \{\}$
$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_0$
while $|V| < N$ **do**
  $\boldsymbol{\theta}' \sim q(\boldsymbol{\theta}, \cdot)$
  $u \sim \mathcal{U}(0, 1)$
  if $u < \min\left\{\frac{\varphi(\boldsymbol{\theta}')q(\boldsymbol{\theta}', \boldsymbol{\theta})}{\varphi(\boldsymbol{\theta})q(\boldsymbol{\theta}, \boldsymbol{\theta}')}, 1\right\}$ **then**
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}'$
  end
  append $\boldsymbol{\theta}$ to V
end

ular, Chib and Greenberg discuss several strategies for choosing q which have been put forth over the last several decades [15]. Notice that if q is symmetric, $q(\boldsymbol{\theta}, \boldsymbol{\theta}') = q(\boldsymbol{\theta}', \boldsymbol{\theta})$, then our acceptance criterion reduces to checking if $u < \min\left\{\varphi(\boldsymbol{\theta}')/\varphi(\boldsymbol{\theta}), 1\right\}$. If in addition we sample from a uniform density for each paramter, so that $\forall i : \theta_i \sim \mathcal{U}(a_i, b_i)$, then the condition $u < \min\left\{\varphi(\boldsymbol{\theta}')/\varphi(\boldsymbol{\theta}), 1\right\}$ reduces to simply checking that each parameter lies in the prior. That is, accept $\boldsymbol{\theta}' = (\theta_1', \ldots, \theta_n')$ with probability

$$\prod_{i=1}^{n} \mathbf{1}_{[a_i \leq \theta_i' \leq b_i]} \in \{0, 1\}. \tag{2.5}$$

It should be noted that the advantages of algorithm 5 are curbed by some possible drawbacks. If the variance of q is too low, then if the algorithm is currently at an accepted value $\boldsymbol{\theta}$,

there is a high probability that it will jump to a value extremely close to $\boldsymbol{\theta}$. Thus, a low variance will cause a very timid chain which will take unreasonably long to span all of the prior, instead fixating on regions of high acceptance. While the region will be eventually well-explored as the chain's length approaches infinity, this will be a computational nightmare in practice. On the other hand, if the variance is too high, then the generated candidates will in general be far from the most recent accepted value, inducing a low acceptance rate and negating the benefits of using M-H in the first place. Therefore, the choice of q requires a good degree of creativity from the modeler, and will surely depend on some trial-and-error before a useful density is identified. It is common to adjust q, along with our tolerance ε, so that we get a desirable acceptance rate (e.g. accepting 30% of proposed parameters).

Due to the above observations, it is often useful to use a normal density for $q(\cdot, \cdot)$. That is, for $\boldsymbol{\theta}, \boldsymbol{\theta}' \in \mathbb{R}^n$,

$$q(\boldsymbol{\theta}, \boldsymbol{\theta}') = \frac{1}{(2\pi)^{n/2}\sqrt{\det \boldsymbol{\Sigma}}} \exp\left\{ -\frac{1}{2} \left(\boldsymbol{\theta}' - \boldsymbol{\theta}\right)^\top \boldsymbol{\Sigma}^{-1} \left(\boldsymbol{\theta}' - \boldsymbol{\theta}\right) \right\}, \tag{2.6}$$

where the mean of the new candidate $\boldsymbol{\theta}'$ is the most recently accepted value $\boldsymbol{\theta}$. The parameters $\boldsymbol{\theta}$ are often assumed to be pairwise independent, so that the variance matrix becomes $\boldsymbol{\Sigma} = \sigma \mathbf{I}_{k \times k}$ for some vector of variances $\boldsymbol{\sigma}^2 = \left(\sigma_1^2, \ldots, \sigma_k^2\right)$. This choice of q is desirable due to its symmetry and the relative ease of adjusting its variance [49]. Using a normal candidate-generating density also exploits the advantage of using MCMC over the rejection method, since proposed $\boldsymbol{\theta}'$ will most likely be fairly close to the most recently accepted $\boldsymbol{\theta}$. For the MCMC techniques used later in this thesis, we will utilize the candidate-generating density given by equation 2.6 unless otherwise noted. However, we will maintain an arbitrary $q(\cdot, \cdot)$ in this chapter's algorithms for completeness.

We are now in position to revise algorithm 4 using the Metropolis Hasting algorithm described in algorithm 5. We henceforth refer to it as ABC-MCMC, and have laid out in algorithm 6. This will be our default variant of ABC (excepting the case of determining an initial guess), and consequently, we give a proof of its correctness before we proceed.

Algorithm 6: Generating samples distributed from $f\left(\boldsymbol{\theta}|\varrho\left(\mathcal{D},\mathcal{D}'\right)<\varepsilon\right)$ via ABC-MCMC

Input : Observed data $\mathcal{D}$, arbitrary initial value $\boldsymbol{\theta}_0$, candidate-generating density $q(\cdot,\cdot)$, model $\mathcal{M}$, , metric $\varrho(\cdot,\cdot)$, tolerance $\varepsilon>0$, desired number of samples N

Output: A vector V of samples approximately distributed from $f(\boldsymbol{\theta}|\varrho(\mathcal{D},\mathcal{D}')<\varepsilon)$

$V \leftarrow \{\}$

$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_0$

while $|V| < N$ **do**

  generate $\boldsymbol{\theta}' \sim q(\boldsymbol{\theta},\cdot)$

  simulate $\mathcal{D}'$ from $\mathcal{M}$ parametrized by $\boldsymbol{\theta}'$

  if $\varrho\left(\mathcal{D},\mathcal{D}'\right)<\varepsilon$ **then**

    $u \sim \mathcal{U}(0,1)$

    if $u < \min\left\{\frac{\pi(\boldsymbol{\theta}')q(\boldsymbol{\theta}',\boldsymbol{\theta})}{\pi(\boldsymbol{\theta})q(\boldsymbol{\theta},\boldsymbol{\theta}')},1\right\}$ **then**

      $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}'$

    end

  end

  append $\boldsymbol{\theta}$ to V

end

return V

Theorem 1. *Algorithm 5 produces samples from $f(\theta|\mathcal{D})$ as $|V| \to \infty$ and $\varepsilon \to 0$.*

Proof. Let us denote the probability transition of the chain as $r(\theta, \theta')$. Without loss of generality, assume that θ, θ' satisfy

$$\frac{\pi(\theta')q(\theta', \theta)}{\pi(\theta)q(\theta, \theta')} \leq 1.$$

The probability that the chain jumps from θ to θ' is represented by

$$q(\theta, \theta') \times \mathbf{P}\left[\mathcal{D}|\theta\right] \times \min\left\{\frac{\pi(\theta')q(\theta', \theta)}{\pi(\theta)q(\theta, \theta')}, 1\right\}.$$

So, we see that if $\varepsilon = 0$,

$$\begin{aligned}
f(\theta|\mathcal{D})r(\theta, \theta') &= f(\theta|\mathcal{D})q(\theta, \theta')\mathbf{P}\left[\mathcal{D}|\theta'\right] \min\left\{\frac{\pi(\theta')q(\theta', \theta)}{\pi(\theta)q(\theta, \theta')}, 1\right\} \\
&= \frac{\mathbf{P}\left[\mathcal{D}|\theta\right]\pi(\theta)}{\mathbf{P}\left[\mathcal{D}\right]} q(\theta, \theta')\mathbf{P}\left[\mathcal{D}|\theta'\right] \left(\frac{\pi(\theta')q(\theta', \theta)}{\pi(\theta)q(\theta, \theta')}\right) \\
&= \frac{\mathbf{P}\left[\mathcal{D}|\theta\right]}{\mathbf{P}\left[\mathcal{D}\right]} \mathbf{P}\left[\mathcal{D}|\theta'\right] \pi(\theta')q(\theta', \theta) \\
&= \frac{\mathbf{P}\left[\mathcal{D}|\theta'\right]\pi(\theta')}{\mathbf{P}\left[\mathcal{D}\right]} q(\theta', \theta)\mathbf{P}\left[\mathcal{D}|\theta\right] \cdot 1 \\
&= f(\theta'|\mathcal{D})q(\theta', \theta)\mathbf{P}\left[\mathcal{D}|\theta\right] \min\left\{\frac{\pi(\theta)q(\theta, \theta')}{\pi(\theta')q(\theta', \theta)}, 1\right\} \\
&= f(\theta'|\mathcal{D})r(\theta', \theta)
\end{aligned}$$

Here we have utilized the fact that

$$\frac{\pi(\theta')q(\theta', \theta)}{\pi(\theta)q(\theta, \theta')} \leq 1 \implies \frac{\pi(\theta)q(\theta, \theta')}{\pi(\theta')q(\theta', \theta)} = \frac{1}{\frac{\pi(\theta')q(\theta', \theta)}{\pi(\theta)q(\theta, \theta')}} \geq 1.$$

So, the above relationship shows that $f(\theta|\mathcal{D})$ satisfies the detailed-balance equations for $r(\theta, \theta')$. The argument for when

$$\frac{\pi(\theta')q(\theta', \theta)}{\pi(\theta)q(\theta, \theta')} > 1$$

proceeds in a symmetric manner. Therefore, $f(\theta|\mathcal{D})$ is a stationary measure for $r(\theta, \theta')$, and by scaling accordingly, it is the stationary distribution. □

In the case of uniform priors, algorithm 6 will accept a candidate $\boldsymbol{\theta}'$ with probability

$$\mathbf{1}_{\left[\varrho(\mathcal{D}, \mathcal{D}'(\boldsymbol{\theta}')) < \varepsilon\right]} \prod_{i=1}^{n} \mathbf{1}_{[a_i \leq \theta_i' \leq b_i]}. \tag{2.7}$$

This will be the case for most of our analyses in this thesis, and we mention it to reconcile any confusion concerning our implementations included in the appendices. One last remark we should make is that algorithm 6 is extremely sensitive to the initial guess $\boldsymbol{\theta}_0$ due to the relatively small jumping window in comparison to the prior. There are a couple of ways to sidestep this issue. For example, one can perform a grid search over $\boldsymbol{\theta}$ to approximate a minimizing value $\boldsymbol{\theta}_0 \approx \arg\min_{\boldsymbol{\theta}} \varrho(\mathcal{D}, \mathcal{D}'(\boldsymbol{\theta}))$ for our initial guess. Caution must be taken, however, since ϱ is a stochastic function, and therefore a deterministic minimizing value may be difficult to ascertain. Another remedy is to start with algorithm 4 until an accepted parameter has been found, and then proceed with algorithm 6.

2.6 Further Extensions of ABC

Although ABC remains an inchoate collection of techniques, there are numerous versions and enhancements to the implementations discussed above. Much of the validity of ABC relies on the fact that the approximate posterior converges to the true distribution as the number of samples approaches infinity. However, Sisson et. al. point out that even samples as large as 10,000 can produce grossly inaccurate approximations to the posterior, and suggest an alteration based on importance sampling to assuage this inefficiency [69]. Beaumont addresses an inherent bias in Sisson et. al.'s modification, and attenuates this bias by introducing a scheme that iterates through a decreasing sequence of tolerances $\{\varepsilon_1, \ldots, \varepsilon_n\}$ [4]. Sadegh et. al. explore yet another modification that can increase the sampling efficiency and allow for concurrent processing [64]. These variants of ABC among others should always be kept in mind when suspicious of biased posterior samples for the application under consideration. ABC techniques are straightforward, but their attractiveness must be counterbalanced with care and skepticism.

Chapter 3

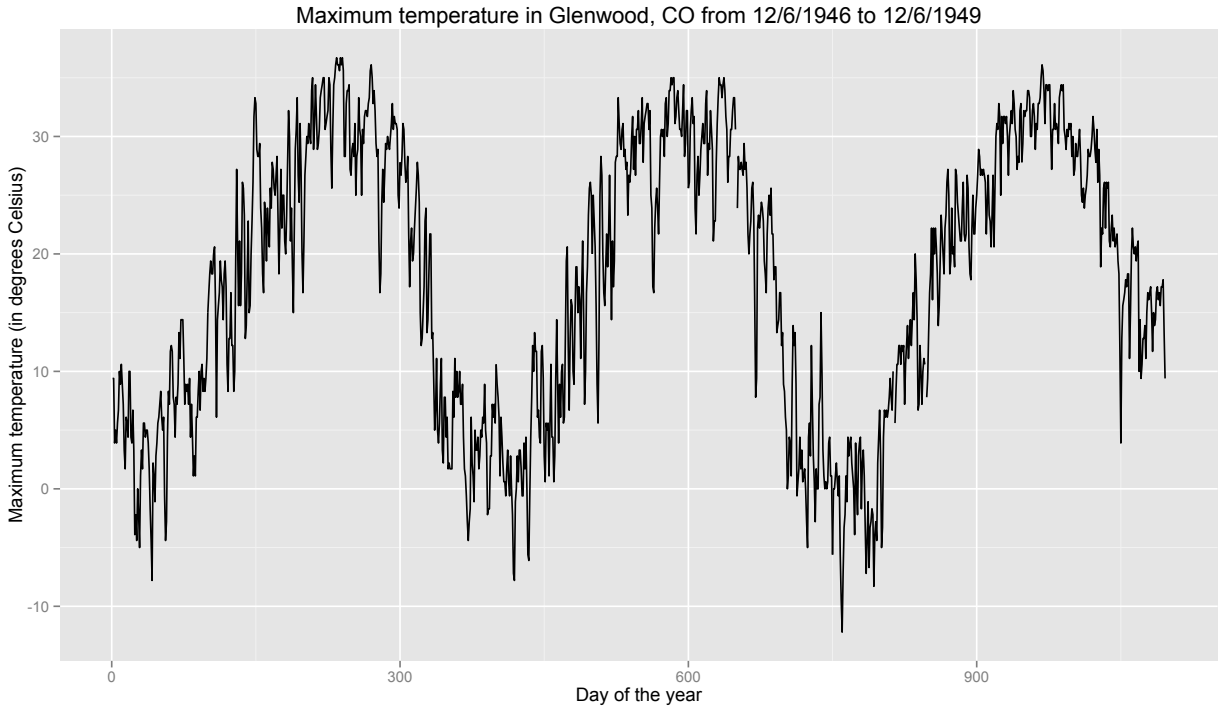
Simulation of Local Daily Maximum Temperature with ABC

3.1 Introduction

Our goal in this chapter is to construct a SWG of daily local temperature using ABC. However, since temperature is a continuous quantity whose value changes throughout the day, we will need to hone in on a particular, precise definition of daily temperature. The minimum and maximum daily temperature at a given location are excellent candidates for several reasons. First, they are both well-defined since they necessarily take on exactly one value for a given day. Moreover, measurements of both minimum and maximum temperatures are readily available for many sites with the help of modern meteorological instruments. In addition, many physical models across multiple disciplines require daily minimum and maximum temperature as an input, such as models of snowmelt runoff [41] and the assessment of climate change [67]. It is natural to suggest that these two measurements of daily temperature should behave similarly, and in fact they have been modeled as a symmetric bivariate process by Kleiber et. al. [40]. So, without loss of generality, we select daily local maximum temperature as the desired quantity to model and simulate in this chapter.

Before we choose our statistical model, we should get some intuition about how daily maximum temperature behaves. To do this, let's first examine a plot of the maximum temperature by day over the course of three years from data collected from Glenwood Springs, Colorado, seen in figure 3.1. This snapshot of temperature data gives the impression that the mean function adheres to a sinusoidal pattern. If we partition our data into the integers 1 through 365, where 1 corre-

Figure 3.1: Daily maximum temperature in Glenwood, Colorado over the span of 3 years.



sponds to January 1 and 365 corresponds to December 31, and take the empirical mean of each day, we can observe the trend, shown in figure 3.2. In fact, looking at the three-year snapshot, we can also convince ourselves that the variance of the temperature also depends on the time of year. If we also look at the empirical standard deviation of each day via figure 3.3, we see that it too follows an oscillatory mean function. These observed characteristics of our data will govern the formulation of our model in the next section.

3.2 Statistical Model

We are interested in modeling the maximum temperature T of a location $\mathbf{s} \in D \subset \mathbb{R}^2$. Our model takes the form

$$T(\mathbf{s}, d) = \mathbf{X}(\mathbf{s}, d)\boldsymbol{\beta}(\mathbf{s}) + W(\mathbf{s}, d; \boldsymbol{\alpha}) \tag{3.1}$$

It is helpful to think of the mean term $\mathbf{X}(\mathbf{s}, d)\boldsymbol{\beta}(\mathbf{s})$ as the local climate, and the stochastic term $W(\mathbf{s}, t; \boldsymbol{\alpha})$ as the local weather. For our most general model, the regressors which comprise our design matrix $\mathbf{X}$ will include a constant term, an autoregressive term for the previous day's maximum temperature, a pair of sinusoidal terms to encapsulate the sinusoidal mean, and a pair of higher-order sinusoidal terms. Thus, for a given day d and location $\mathbf{s}$, we have

$$\mathbf{X}(\mathbf{s}, d) = \left(1, T(\mathbf{s}, d-1), \cos\left(\frac{2\pi d}{365}\right), \sin\left(\frac{2\pi d}{365}\right), \cos\left(\frac{4\pi d}{365}\right), \sin\left(\frac{4\pi d}{365}\right)\right)^{\mathsf{T}}. \tag{3.2}$$

which implies a corresponding vector of parameters $\boldsymbol{\beta} = (\beta_0, \ldots, \beta_5)^{\mathsf{T}}$. Note that $\beta_2 \cos(2\pi d/365) + \beta_3 \sin(2\pi d/365) = A\cos(2\pi(d+\delta)/365)$ for some amplitude A and shift δ, so these covariates represent a first-order sine wave whose period is 365 days. The inclusion of $\beta_4 \cos(4\pi d/365) + \beta_5 \sin(4\pi d/365)$ yields a second first-order sine wave of twice the frequency, and hence their combination is a second-order sine wave with a 365-day period.

Moreover, we model the W term as a normally distributed random variable with a sinusoidal variance. More technically, we assume

$$W(\mathbf{s}, d; \boldsymbol{\alpha}) \sim \mathcal{N}\left(0, \sigma^2(\mathbf{s}, d)\right)$$

Figure 3.2: Daily empirical mean of maximum temperature in Glenwood, Colorado.

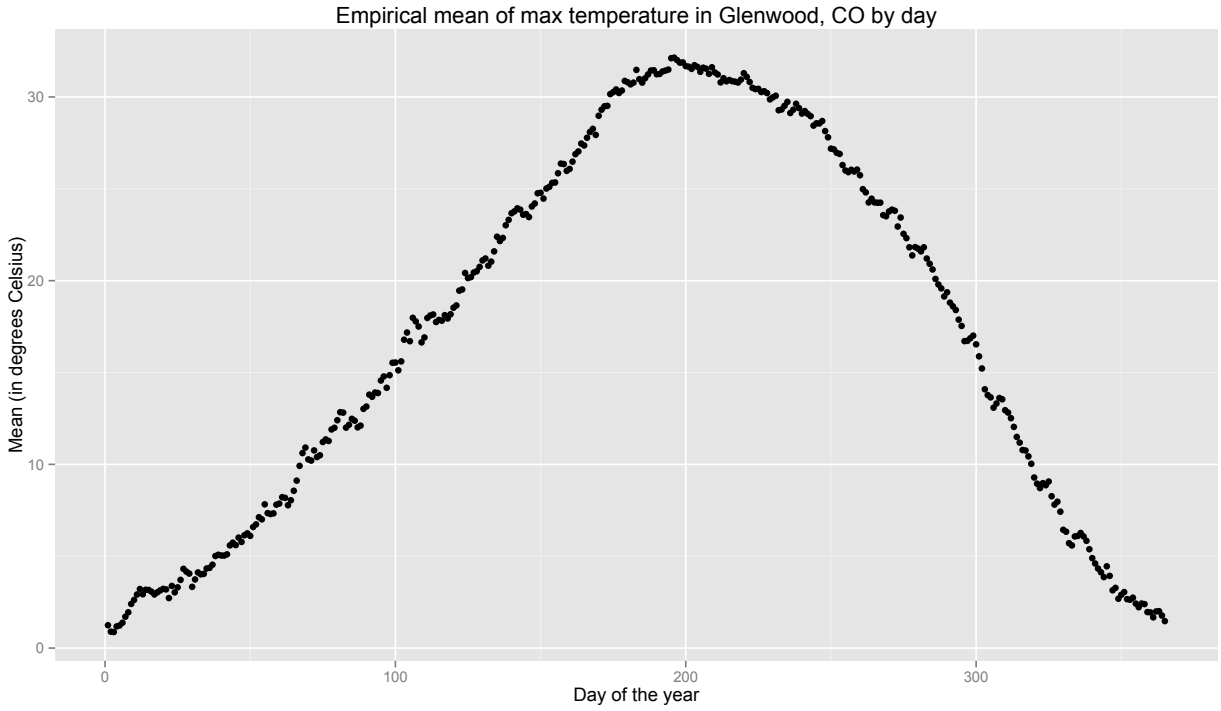
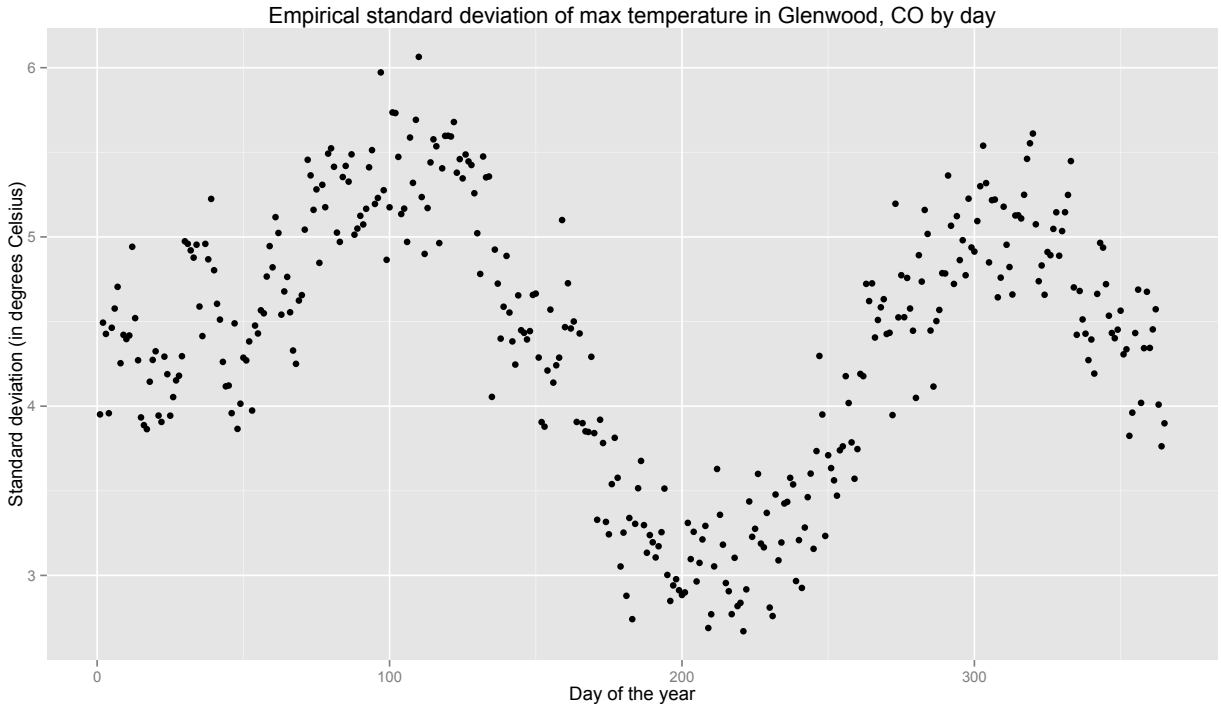


Figure 3.3: Daily empirical standard deviation of maximum temperature in Glenwood, Colorado.



where $\boldsymbol{\alpha} = (\alpha_0, \cdots, \alpha_4)$, and

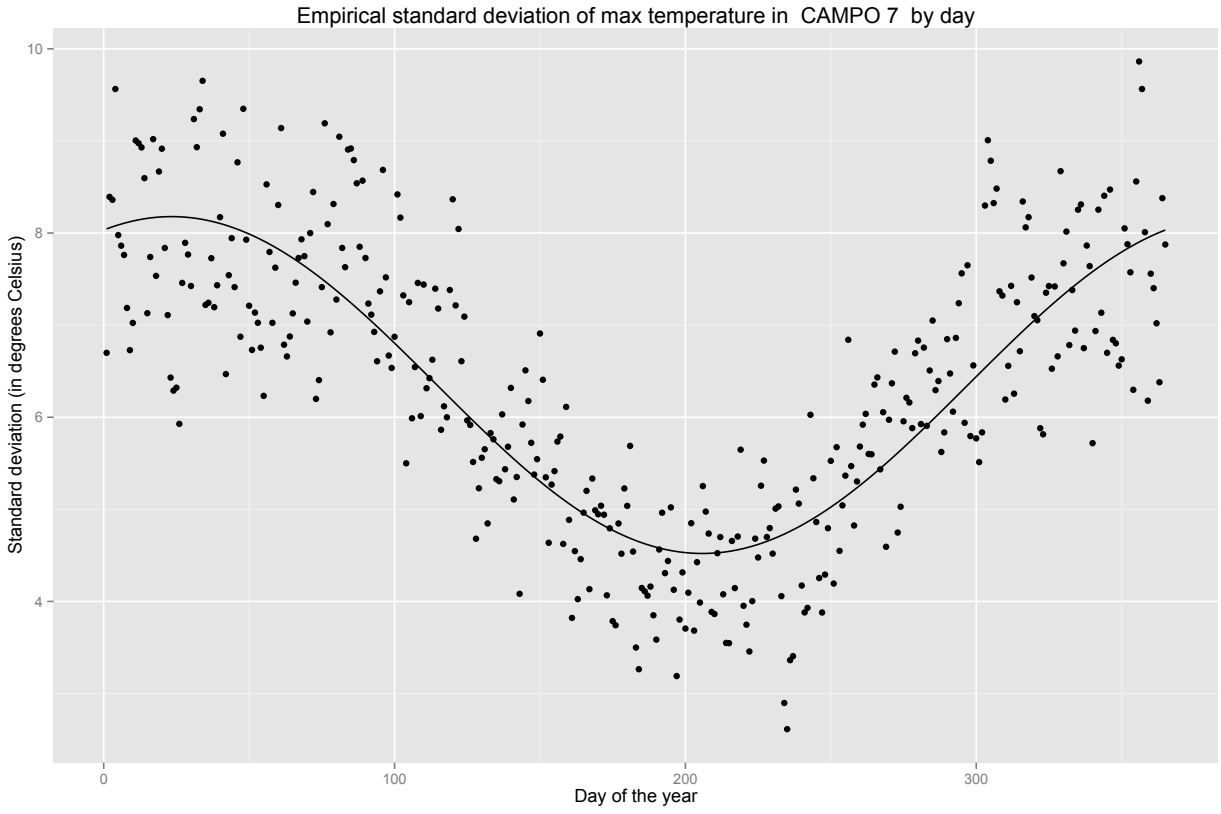
$$\sigma^2(\mathbf{s}, d) = \exp\left\{\alpha_0(\mathbf{s}) + \alpha_1(\mathbf{s})\cos\left(\frac{2\pi d}{365}\right) + \alpha_2(\mathbf{s})\sin\left(\frac{2\pi d}{365}\right) + \alpha_3(\mathbf{s})\cos\left(\frac{4\pi d}{365}\right) + \alpha_4(\mathbf{s})\sin\left(\frac{4\pi d}{365}\right)\right\}.$$

Note that σ^2 takes the form of an exponential to ensure positivity. This detail can be altered as necessary if the exponential does not provide a good enough fit for the data, as long as the transformation preserves the sinusoidal behavior and will guarantee that $\sigma^2(\mathbf{s}, d) \geq 0$.

One immediate concern of this model is that it burdens us with 11 separate parameters of interest, since length$(\boldsymbol{\beta}) = 6$ and length$(\boldsymbol{\alpha}) = 5$. To address this, there are a couple of remedies we can employ that depend on the problem domain. First, not every location $\mathbf{s}$ will require the higher-order harmonic terms present in $\mathbf{X}$ or σ^2, and for those that can get away with it, we can just set $\alpha_3 = \alpha_4 = 0$ and/or $\beta_4 = \beta_5 = 0$, and use a standard sinusoidal approximation to the mean and/or variance. This is in practice a reasonable assumption for many locations. For example, we can plot the data from the Campo 7, Colorado site along with the Maximum Likelihood sinusoidal fit, which is shown in Figure 3.4. A quick visual sanity check should assure the modeler whether or not making this simplification would be sensible.

For the case of Glenwood Springs above, the oscillation is clearly of a higher order than a simple sine wave, so we need to handle the general case of length$(\boldsymbol{\alpha}) = 5$. One common solution, which is the one we will employ repeatedly in this thesis, is to first estimate the mean parameters $\boldsymbol{\beta}$ using a fixed, small variance, and then use these estimated parameters $\widehat{\boldsymbol{\beta}}$ to estimate the stochastic parameters $\boldsymbol{\alpha}$. This is justified by the segregation of $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$, attached to separate terms $\mathbf{X}$ and W, respectively. For the case of estimating temperature, we can first set α_0 to be an arbitrarily small number and then set the rest of the paramters α_i to be zero. This will minimize the influence of the variance on the mean estimators.

Figure 3.4: Daily empirical standard deviation data for the Campo 7, Colorado site along with its MLE sinusoidal fit.



3.3 Metrics for Estimating Temperature

Since our model was constructed to capture the oscillatory nature of both the mean and variance of the temperature, these are the two criteria which motivate the metrics required for our ABC procedure. In what follows, let $\mathbf{s} = \mathbf{s}_\gamma$ be the simulated time series from our model (3.1) parametrized by parameters γ, where $\gamma = \left(\boldsymbol{\alpha}^\mathsf{T}, \boldsymbol{\beta}^\mathsf{T}\right)^\mathsf{T}$. To assess how similar $\mathbf{s}_\gamma$ is to observed data $\mathbf{o}$, we can look at the standardized error of the mean concurrently with the standardized error of the standard deviation. Moreover, we will allow the modeler to penalize these errors separately using weights $c_m > 0$ for the mean error and $c_v > 0$ for the variance error. We can formalize these ideas mathematically with the following equation:

$$\widehat{\gamma}_{\text{ABC}} = \arg\min_{\gamma} \left\{ \sum_{t \in \mathcal{T}} \left(c_m \frac{|\widehat{\mu}\left(\mathbf{o}[t]\right) - \widehat{\mu}\left(\mathbf{s}_\gamma[t]\right)|}{|\widehat{\mu}\left(\mathbf{o}[t]\right)|} + c_v \frac{|\widehat{\sigma}\left(\mathbf{o}[t]\right) - \widehat{\sigma}\left(\mathbf{s}_\gamma[t]\right)|}{|\widehat{\sigma}\left(\mathbf{o}[t]\right)|} \right) \right\} \tag{3.3}$$

where $\mathcal{T}$ is a partition of the months of the year, $\widehat{\mu}(\cdot)$ returns the empirical mean, $\widehat{\sigma}(\cdot)$ returns the empirical standard deviation, and the notation $\mathbf{x}[t]$ represents the time series $\mathbf{x}$ subsetted by the set of time points t. For the remainder of this chapter, we assign equal weight to the mean and variance errors, so that $c_m = c_v = 1$. This indicates that we should define our penalization metric ϱ as

$$\varrho(\mathcal{D}, \mathcal{D}') := \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \left(\frac{|\widehat{\mu}(\mathcal{D}[t]) - \widehat{\mu}(\mathcal{D}'[t])|}{|\widehat{\mu}(\mathcal{D}[t])|} + \frac{|\widehat{\sigma}(\mathcal{D}[t]) - \widehat{\sigma}(\mathcal{D}'[t])|}{|\widehat{\sigma}(\mathcal{D}[t])|} \right) \tag{3.4}$$

where we are scaling by $1/|\mathcal{T}|$ to standardize our values of ϱ making for a more interpretable comparison to ε. Some standard choices for $\mathcal{T}$ are be the usual set of the four common seasons, e.g.

$$\mathcal{T} = \{\text{summer, autumn, winter, spring}\} = \{\{6,7,8\}, \{9,10,11\}, \{12,1,2\}, \{3,4,5\}\}$$

where $1 = \text{January}, 2 = \text{February}, \ldots, 12 = \text{December}$, or simply the set of each month individually, e.g. $\mathcal{T} = \{1, 2, \ldots, 12\}$. The choice of $\mathcal{T}$ amounts to a tradeoff between accuracy via more sets of months in $\mathcal{T}$, and efficiency via less sets in $\mathcal{T}$. In what follows, we elect $\mathcal{T} = \{1, \ldots, 12\}$.

We can now construct our algorithm to get samples from our posterior, shown in Algorithm 7. While this algorithm maintains the form of traditional ABC methods, we note that since ϱ is

a metric involving two statistics, the mean and the standard deviation, one may be tempted to estimate these metrics separately. Indeed, the modeler can let

$$\varrho_\mu(\mathcal{D}, \mathcal{D}') := \sum_{t \in \mathcal{T}} \frac{|\widehat{\mu}(\mathbf{o}[t]) - \widehat{\mu}(\mathbf{s}_\gamma[t])|}{|\widehat{\mu}(\mathbf{o}[t])|},$$

$$\varrho_\sigma(\mathcal{D}, \mathcal{D}') := \sum_{t \in \mathcal{T}} \frac{|\widehat{\sigma}(\mathbf{o}[t]) - \widehat{\sigma}(\mathbf{s}_\gamma[t])|}{|\widehat{\sigma}(\mathbf{o}[t])|},$$

and then simultaneously check if both $\varrho_\mu < \varepsilon_\mu$ and $\varrho_\sigma < \varepsilon_\sigma$ for some user-defined $\varepsilon_\mu, \varepsilon_\sigma > 0$. Thinking of $\varepsilon = \varepsilon_\mu + \varepsilon_\sigma$, it is easy to see that these methods are equivalent; we opt to use the latter due to its modularity and ease of implementation.

3.4 Numerical Results for Glenwood Springs, Colorado

We are ready to apply our model and procedure to real data and demonstrate its efficacy. We will work with data provided by the Global Historical Climatology Network Database (GHCND), which contains daily maximum temperature data for 145 different stations in Colorado, among thousands of other locations across the world [53]. In particular, we choose data from Glenwood Springs, Colorado, which, as seen previously, exhibits high-order oscillatory standard deviation behavior. The GHCND provides a quality flag for each observation; to avoid poor quality observations, we removed any which contained a flag which indicated a corruption. As a result, there is a total of 36,721 days worth of available maximum temperature data. For the sake of simplicity, we remove all leap days from the dataset, so that every available year has exactly 365 days worth of values (including `NA`, i.e., missing data values). These data preparation steps will be enacted in each of the three analyses in Chapters 3 through 5, so we will only mention them here for brevity.

Since we have omitted linear drift as a covariate to our model, our assumption is that the mean and standard deviation should behave similarly for each year of data. Thus, since it is computationally inefficient to use all 36,721 days of data, we isolate a sequence of 20 years of consecutive records, yielding 365*20 = 7,300 days for our observation vector $\mathbf{o}$. While most of this subsequence of data is available and unsullied, there are still days with incomplete data which we

Algorithm 7: ABC-MCMC for daily local maximum temperature parameter estimation

Input : Observed data $\mathcal{D}$, arbitrary initial value $\mathbf{x}_0$, candidate-generating density $q(\cdot,\cdot)$, tolerance $\varepsilon > 0$, desired number of samples N

Output: A vector V of samples approximately distributed from target distribution $f(\boldsymbol{\theta}|\mathcal{D})$

Define $\mathcal{M}$ using (3.1)

Define ϱ as follows:

$$\varrho(\mathcal{D},\mathcal{D}') := \frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}}\left(\frac{|\widehat{\mu}(\mathcal{D}[t]) - \widehat{\mu}(\mathcal{D}'[t])|}{|\widehat{\mu}(\mathcal{D}[t])|} + \frac{|\widehat{\sigma}(\mathcal{D}[t]) - \widehat{\sigma}(\mathcal{D}'[t])|}{|\widehat{\sigma}(\mathcal{D}[t])|}\right)$$

$V \leftarrow \{\}$
$\mathbf{x} \leftarrow \mathbf{x}_0$
while $|V| < N$ **do**
  generate $\boldsymbol{\theta}_{\text{candidate}} \sim q(\boldsymbol{\theta},\cdot)$
  simulate $\mathcal{D}'$ from $\mathcal{M}$ parametrized by $\boldsymbol{\theta}_{\text{candidate}}$
  if $\varrho(\mathcal{D},\mathcal{D}') < \varepsilon$ **then**
    $u \sim \mathcal{U}(0,1)$
    if $u < \min\left\{\frac{\pi(\boldsymbol{\theta}_{\text{candidate}})q(\boldsymbol{\theta}_{\text{candidate}},\boldsymbol{\theta})}{\pi(\boldsymbol{\theta})q(\boldsymbol{\theta},\boldsymbol{\theta}_{\text{candidate}})}, 1\right\}$ **then**
      $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_{\text{candidate}}$
    end
  end
  append $\boldsymbol{\theta}$ to V
end

regard as missing. To compensate, we mask each simulated vector $\mathbf{s}$ with the missing values in $\mathbf{o}$, so that they possess missing data on the same days. We will perform this data masking throughout the rest of the analyses in this thesis. In further pursuit of tractability, we set $\beta_4 = \beta_5 = 0$, yielding a standard sinusoidal wave for our mean function.

We show the results after 500 jumps of the chain, implying about 1,500 samples of $(\boldsymbol{\alpha}^\mathsf{T}, \boldsymbol{\beta}^\mathsf{T})^\mathsf{T}$. Figure 3.5 shows the approximate posterior densities of $\boldsymbol{\beta}$ along with the priors, and figure 3.6 shows the densities of $\boldsymbol{\alpha}$. Our priors are uniform whose mean was chosen via a grid search through the parameters, with the exception of the $\mathcal{U}(0,1)$ prior for the autoregressive parameter β_1 which necessarily lies in $(0,1)$. It is quickly evident that the posterior certainty varies for each parameter; the auto-regressive parameter β_1 has a much narrower density than the sine coefficient β_3. Nonetheless, each parameter seems to be captured within a well-defined posterior density as we hoped. Another interesting feature is that each of the α_i coefficients possesses a similar size and shape of posterior density.

Let us examine some more summaries of the data to get a better assessment of our parameters. First, we compute the empirical mean of each posterior for the mean parameters and call it our ABC estimator for β_i, i.e. $\widehat{\beta}_{i\text{ABC-MCMC}} =_{\text{def}} \widehat{\mu}_{\beta_i}$, and do the same for the posterior of the standard deviation parameters, $\widehat{\alpha}_{i\text{ABC-MCMC}} =_{\text{def}} \widehat{\mu}_{\alpha_i}$. Next, we create 3 independent simulations of the maximum temperature parametrized by our estimators. We can plot the mean for each month for the observed data and these three simulations, which is shown in Figure 3.7. The analogous plot for the monthly standard deviation is shown in Figure 3.8. The simulated means are a reasonable representation of the observed mean, although they do systematically underestimate the curve during the summer and fall months. This is almost certainly due to the absence of the higher-order sine terms in our model since the observed empirical means do not exactly fit a regular sine wave. The modeler can seek the power of computing resources such as a supercomputer if a more accurate estimation of the mean were essential. The simulated standard deviations also follow the trend of the observed standard deviations quite closely. To get an even more comprehensive picture of the accuracy of our posteriors, we can create a boxplot of the values from each density

Figure 3.5: Prior and approximate posterior distributions for $\boldsymbol{\beta}$

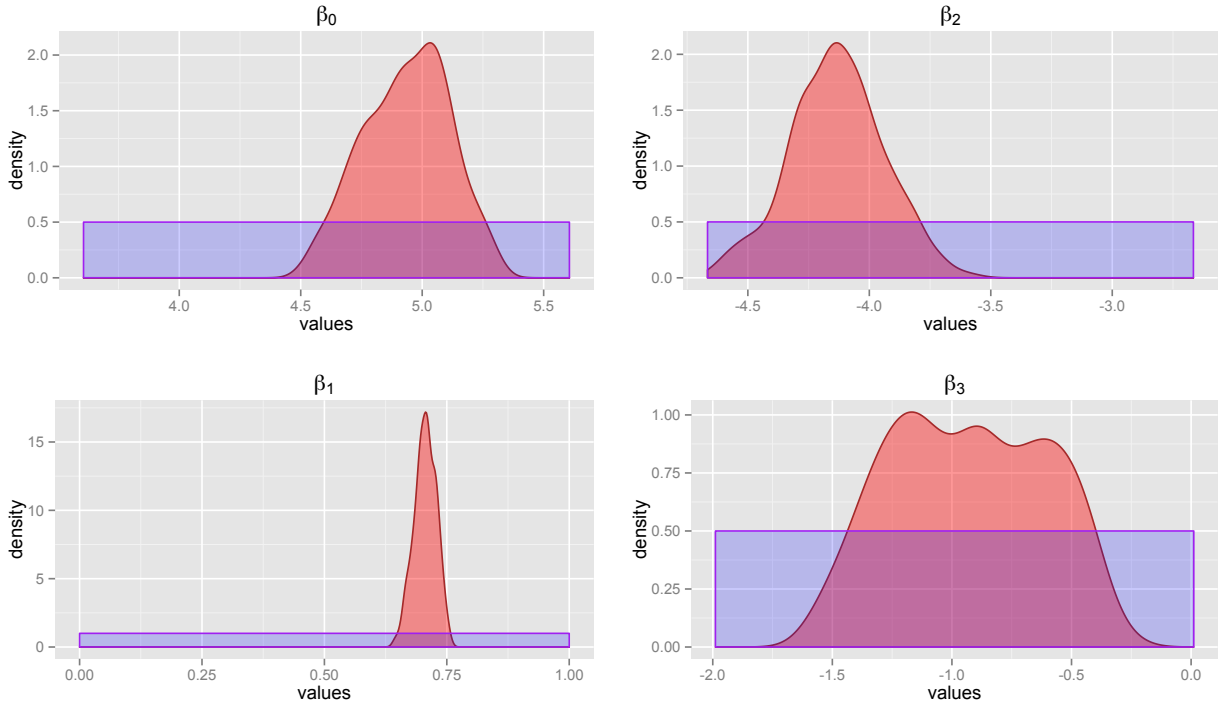
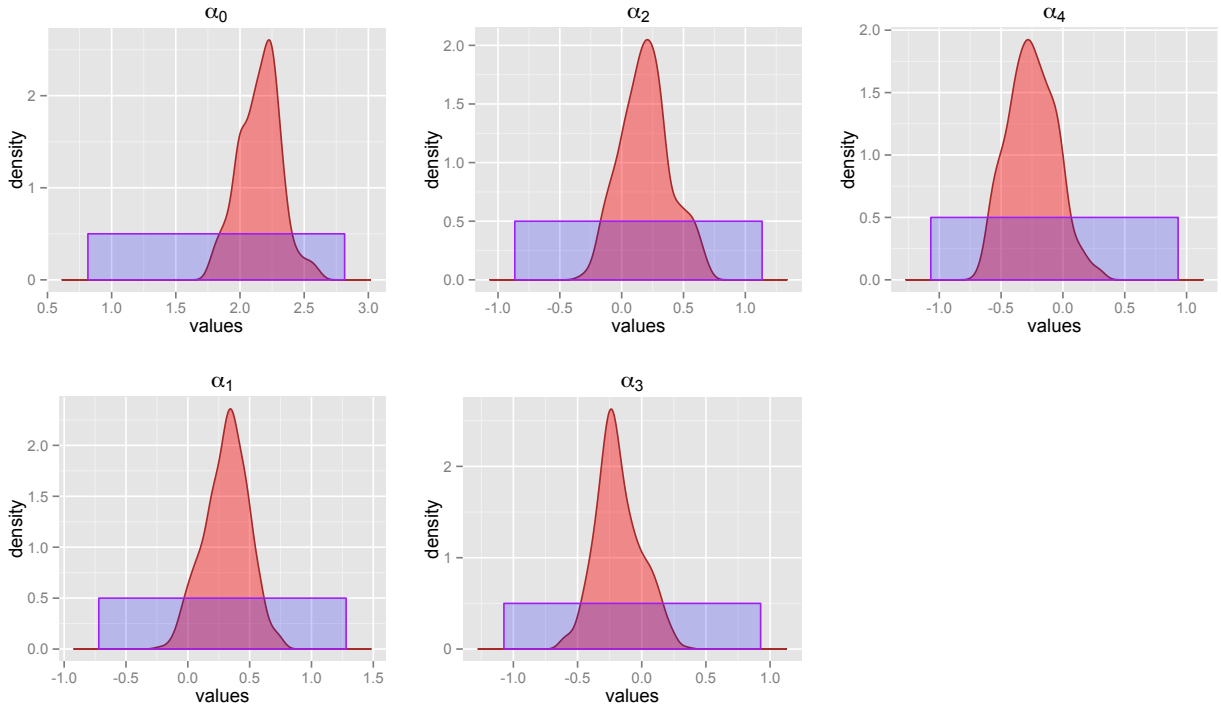


Figure 3.6: Prior and approximate posterior distributions for $\boldsymbol{\alpha}$



and plot them over the values for the observations, which is shown for $\boldsymbol{\beta}$ in Figure 3.9 and for $\boldsymbol{\alpha}$ in Figure 3.10. It turns out that there is strong and consistent agreement of the boxplot means with the observed means for almost every month.

Nevertheless, our model has limitations, some of which can be seen when comparing the hot and cold spell counts of the observations and simulations. Hot/cold spells are a difficult phenomenon to capture, partly because there are many different ways to define a hot/cold spell. One natural definition of a hot spell, for instance, is a series of consecutive days d_i where $T_{\max}(d_i) > \mu(d_i)$ for each i, where $\mu(d_i)$ is the mean function of temperature for day d_i. Note that $\mu(d_i)$ is a value that must somehow be estimated, and this introduces some uncertainty in our hot spell definition. Another natural spell definition is a set of days d_i that are greater than some specified threshold value τ regardless of the day of the year. This can be considered a form of absolute hot spell as opposed to the previous one, which is a more relative assessment. We investigate some hot and cold spell figures based on the latter definition, seen in Figures 3.11 through 3.16. It appears that this particular realization matches the cold spell statistics very well, whereas it fails to match the hot spell statistics as closely, probably due to the discrepancies in the mean function for the autumn months.

To conclude our examination of our model's efficacy, we can display a sample SWG realization from the posterior parameters along with the corresponding temperature data. This is shown in figure 3.17. The masking of missing data into the simulation is evident in this plot, seen by the long sequences of days without recorded maximum temperature values. By visual inspection, these time series are very similar and exhibit similar mean and standard deviation patterns, a final sanity check that our model is producing sound simulations in comparison to the observed data.

3.5 Discussion

Our numerical results for daily maximum temperature simulation reassure us that ABC can provide a dependable methodology in constructing SWGs. Still, there is certainly room for future work. We made the decision to set $\beta_4 = \beta_5 = 0$, but the inclusion of higher-order sinusoidal

Figure 3.7: Monthly mean maximum temperature in degrees Celsius for observed data as well as for 3 simulations parametrized by the empirical posterior means

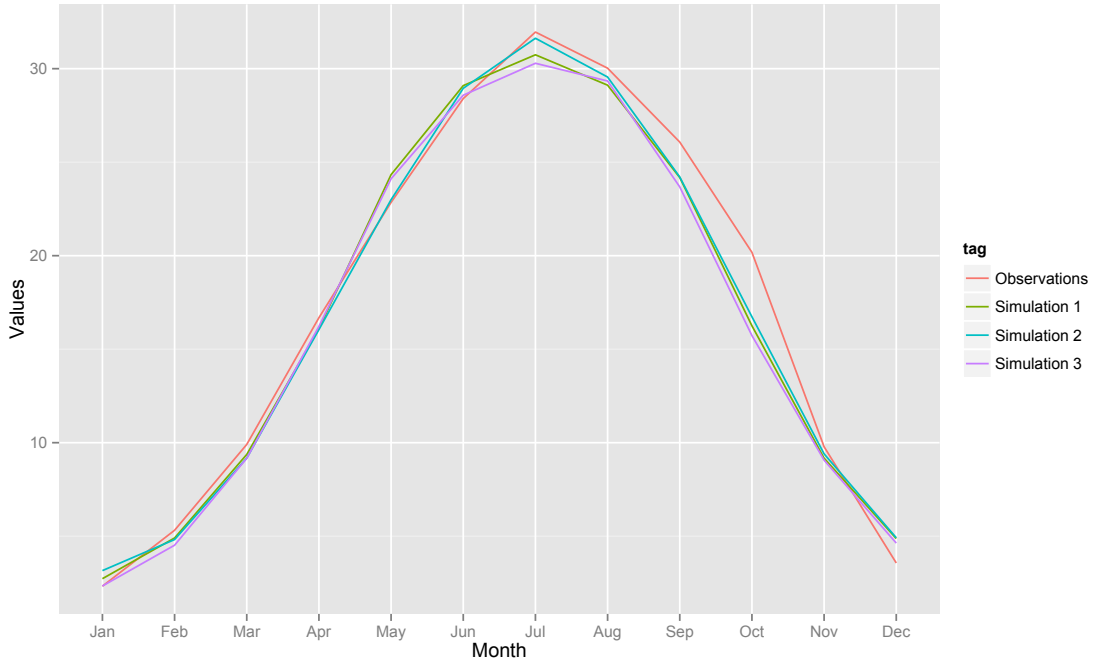


Figure 3.8: Monthly standard deviation of maximum temperature in degrees Celsius for observed data as well as for 3 simulations parametrized by the empirical posterior means

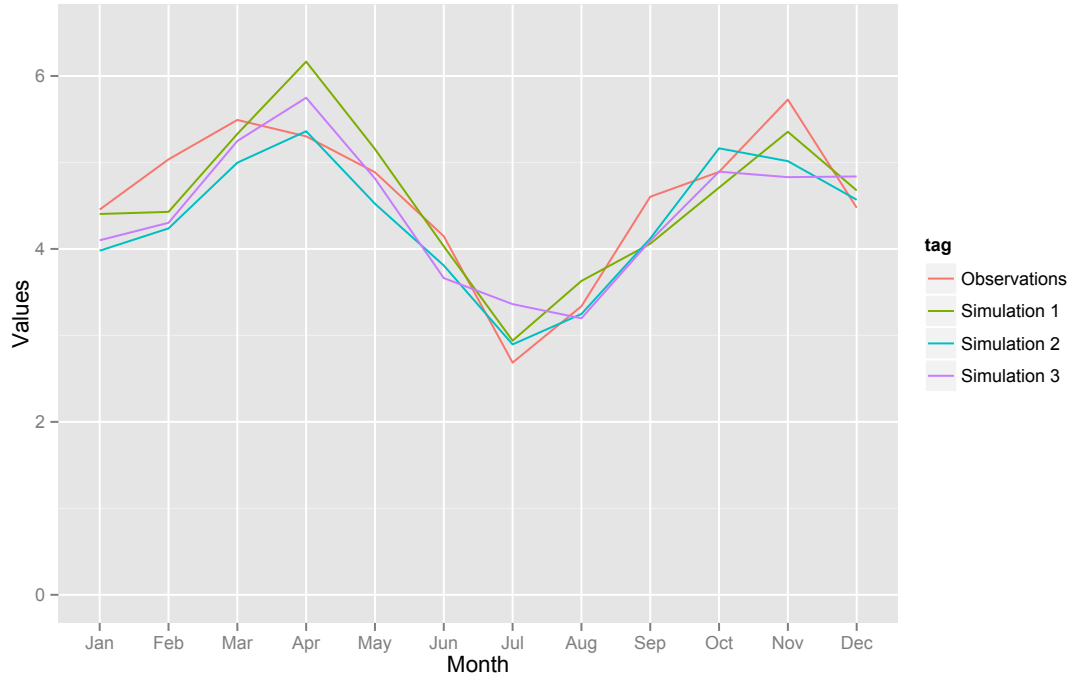


Figure 3.9: Boxplot of simulated average monthly maximum temperature values in degrees Celsius for β as well as those for observed values

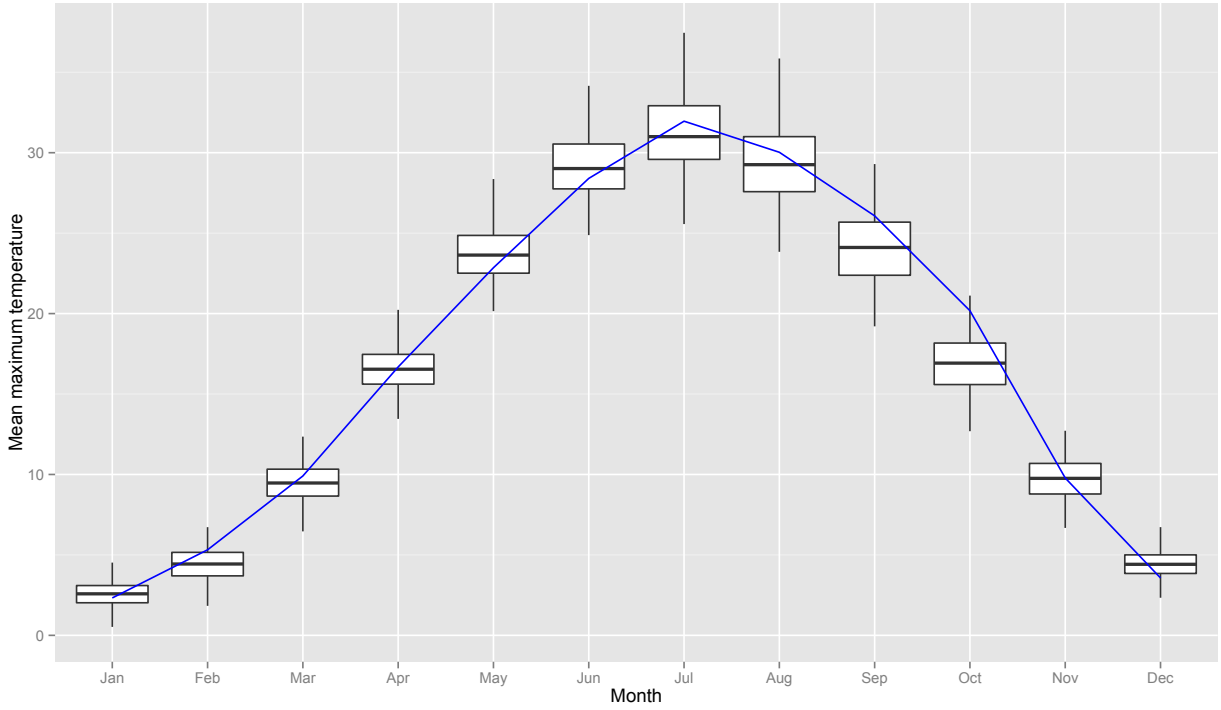


Figure 3.10: Boxplot of simulated average monthly maximum temperature standard deviations in degrees Celsius for $\boldsymbol{\alpha}$ as well as observed values

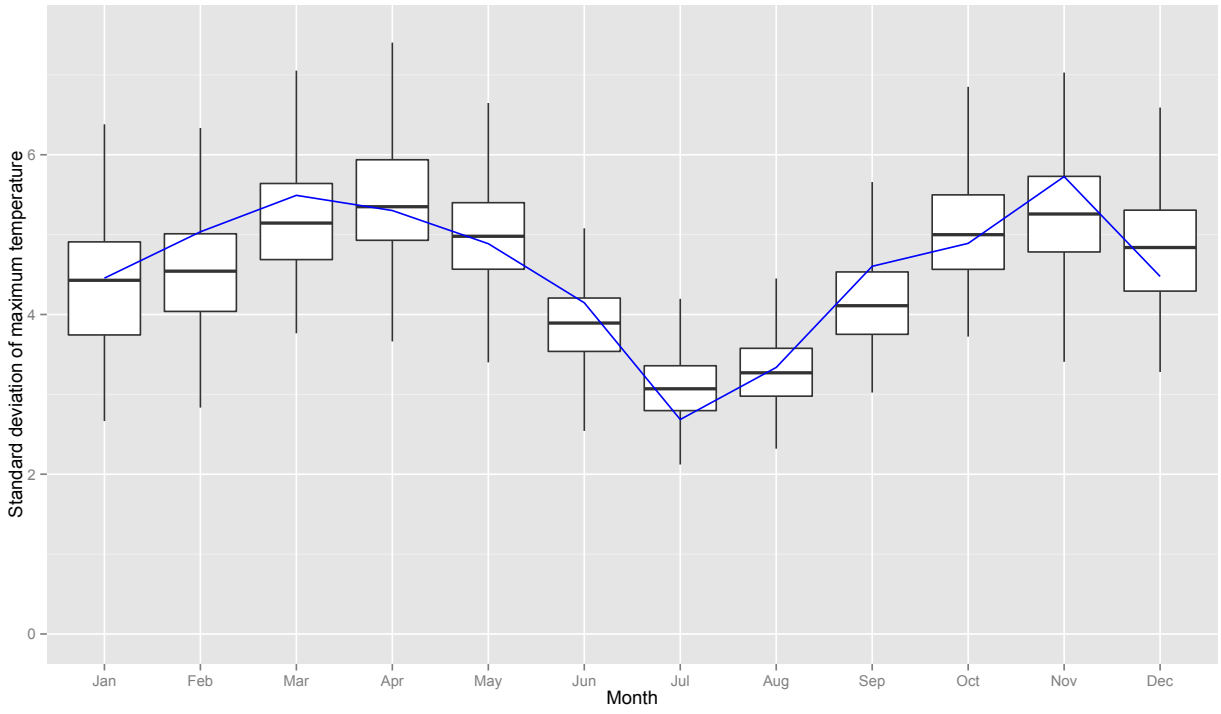


Figure 3.11: Counts of hot spells over 30 degrees Celsius for observations and simulated data for Glenwood Springs, CO.

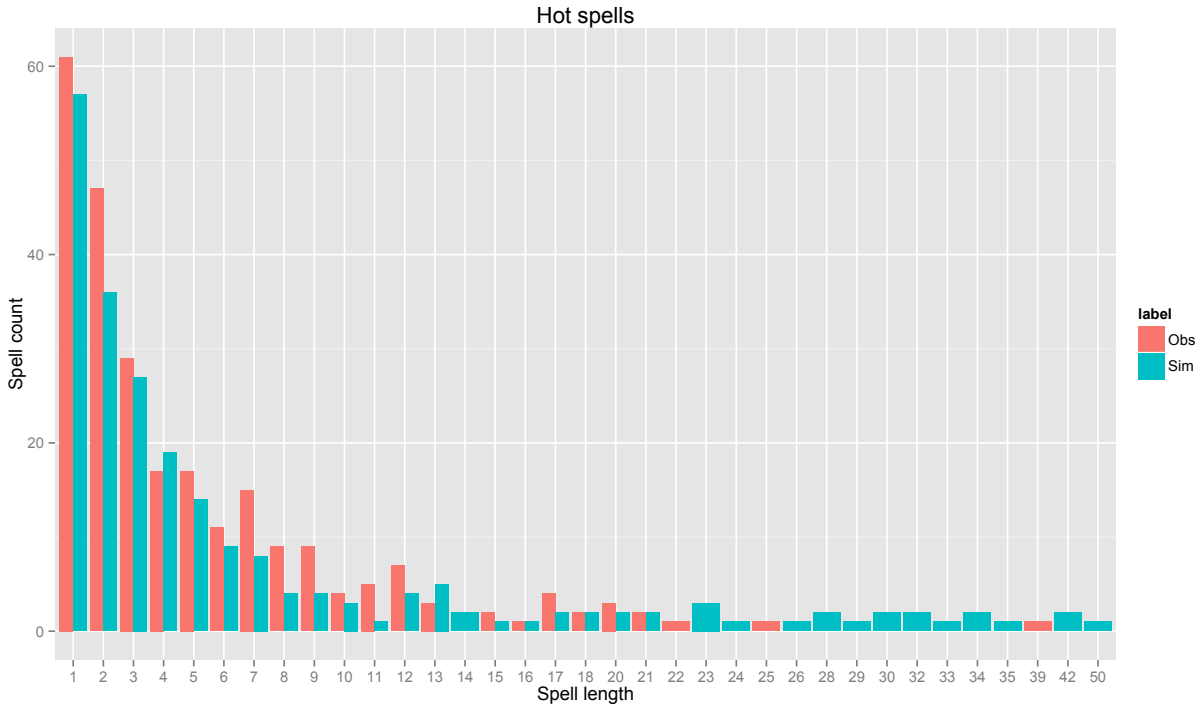


Figure 3.12: Counts of hot spells over 35 degrees Celsius for observations and simulated data for Glenwood Springs, CO.

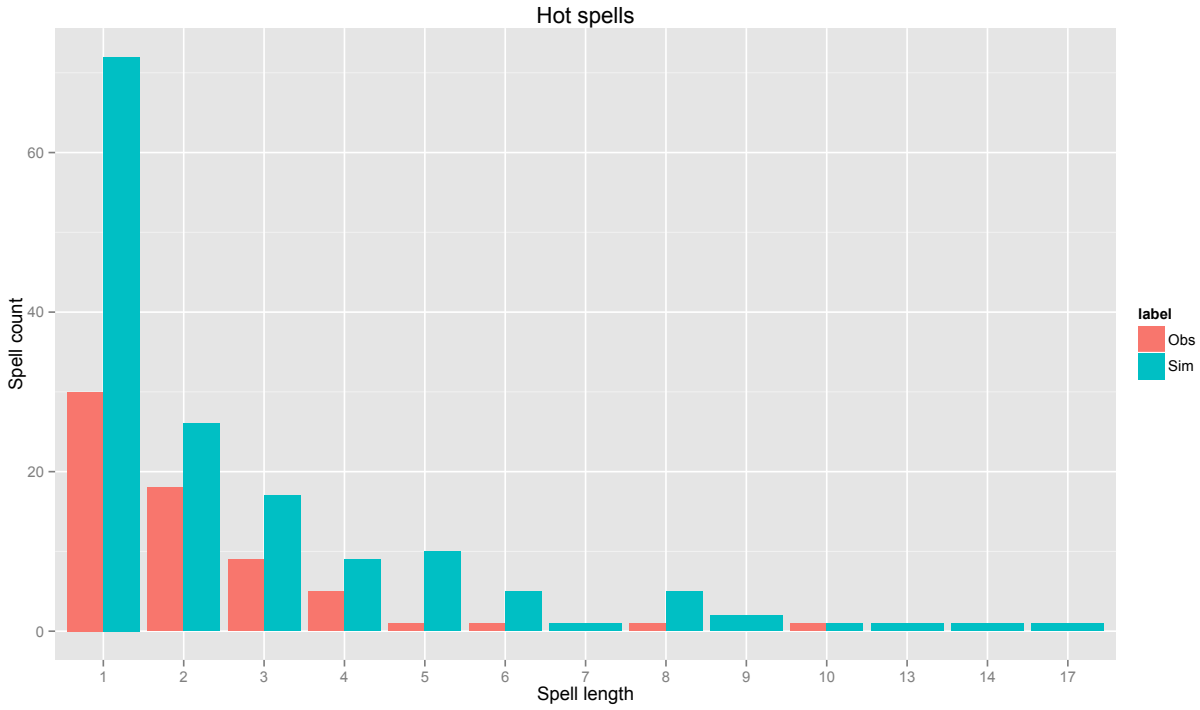


Figure 3.13: Counts of hot spells over 40 degrees Celsius for observations and simulated data for Glenwood Springs, CO.

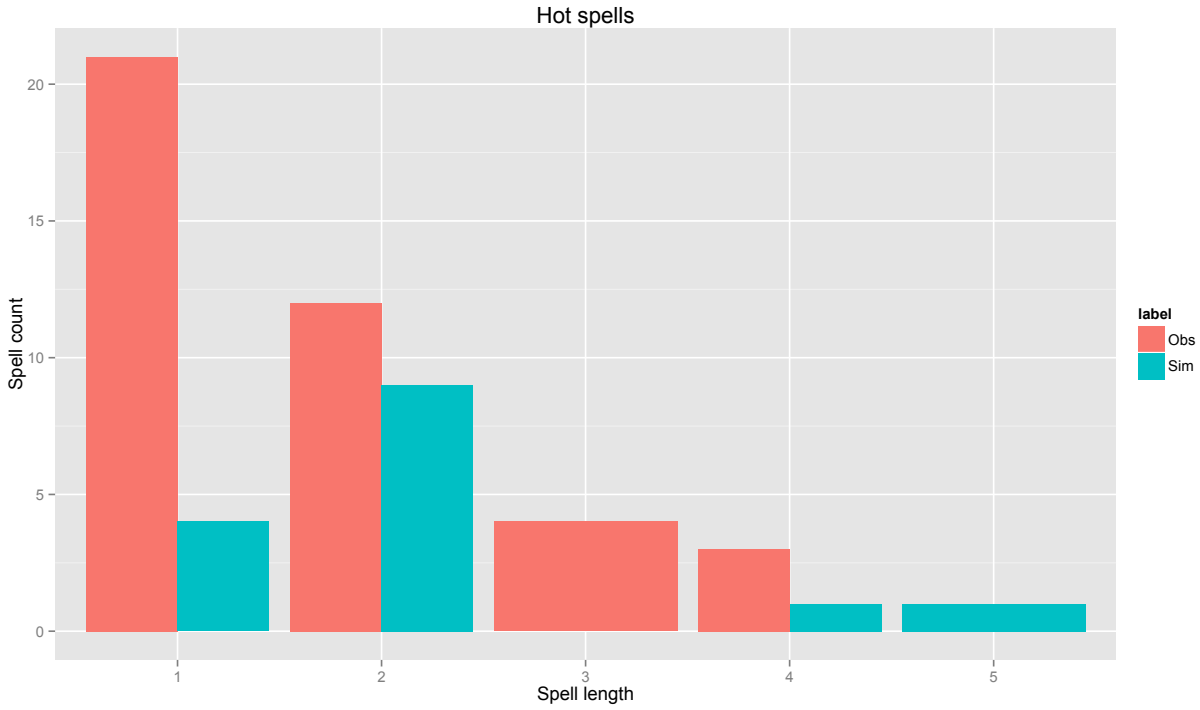


Figure 3.14: Counts of cold spells below 5 degrees Celsius for observations and simulated data for Glenwood Springs, CO.

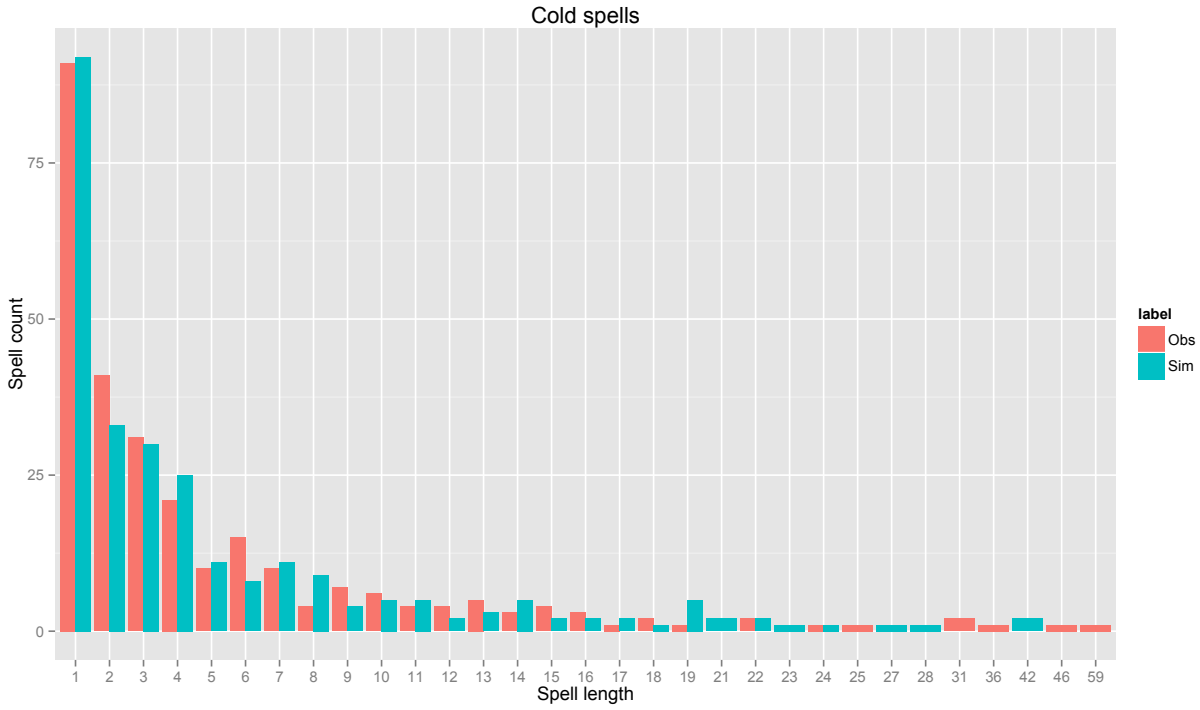


Figure 3.15: Counts of cold spells below 0 degrees Celsius for observations and simulated data for Glenwood Springs, CO.

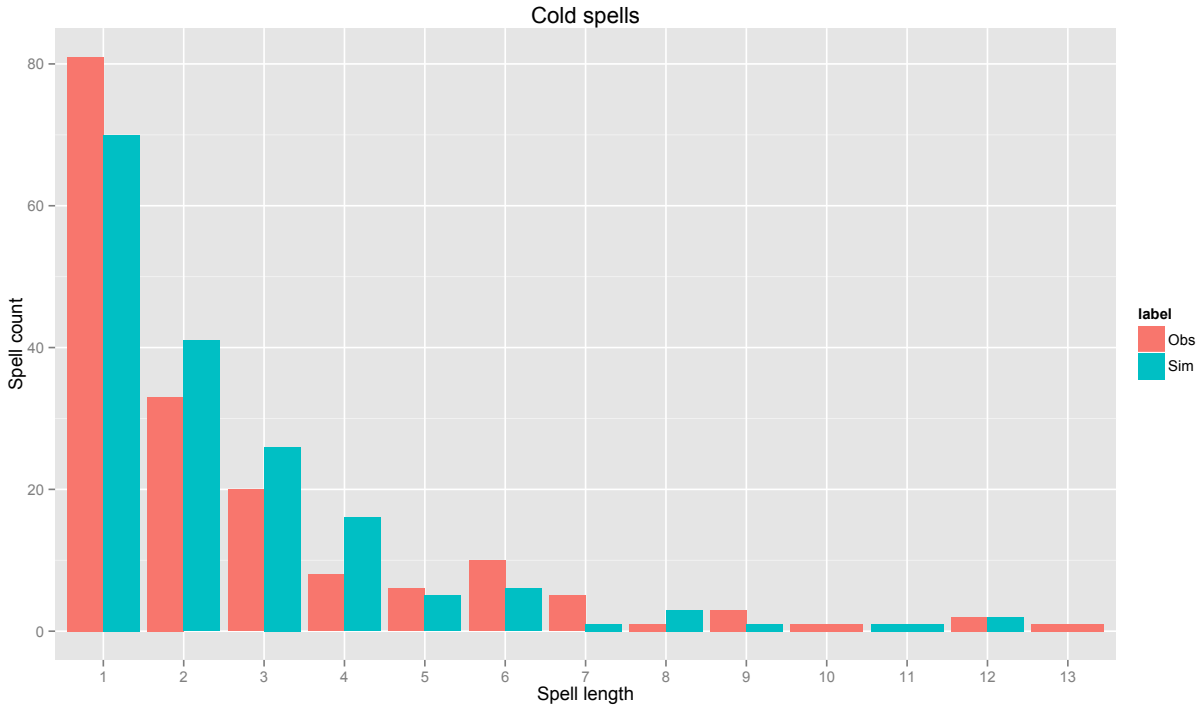


Figure 3.16: Counts of cold spells below -5 degrees Celsius for observations and simulated data for Glenwood Springs, CO.

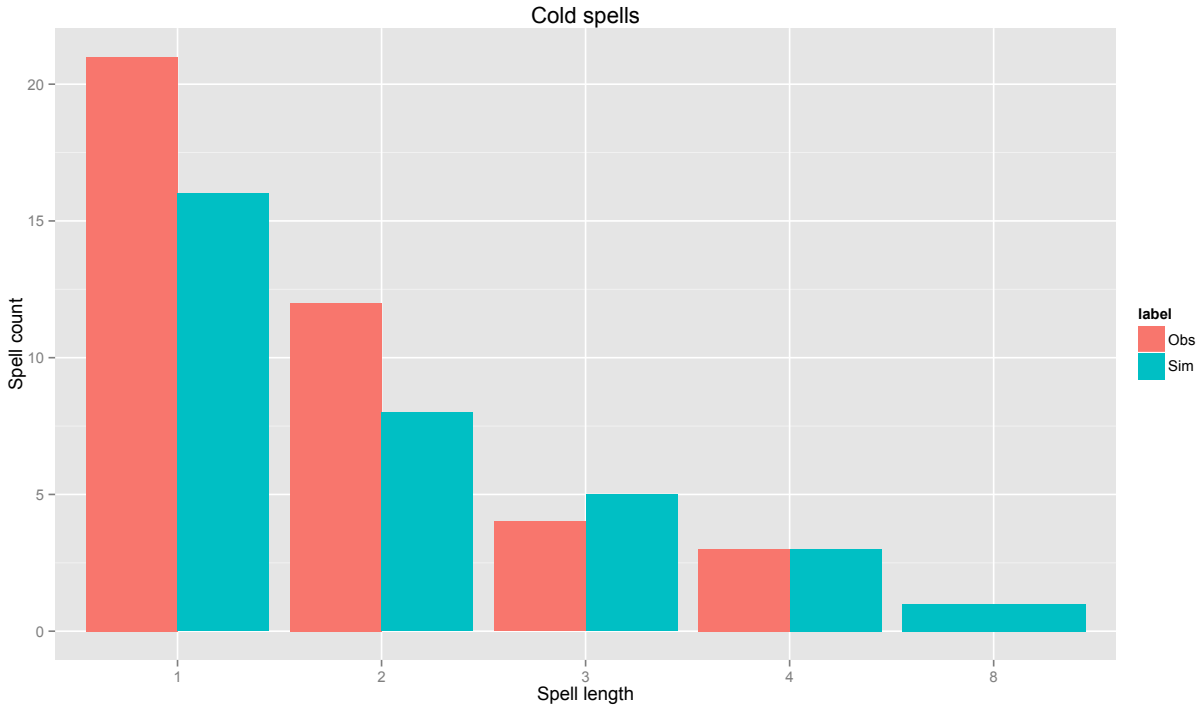
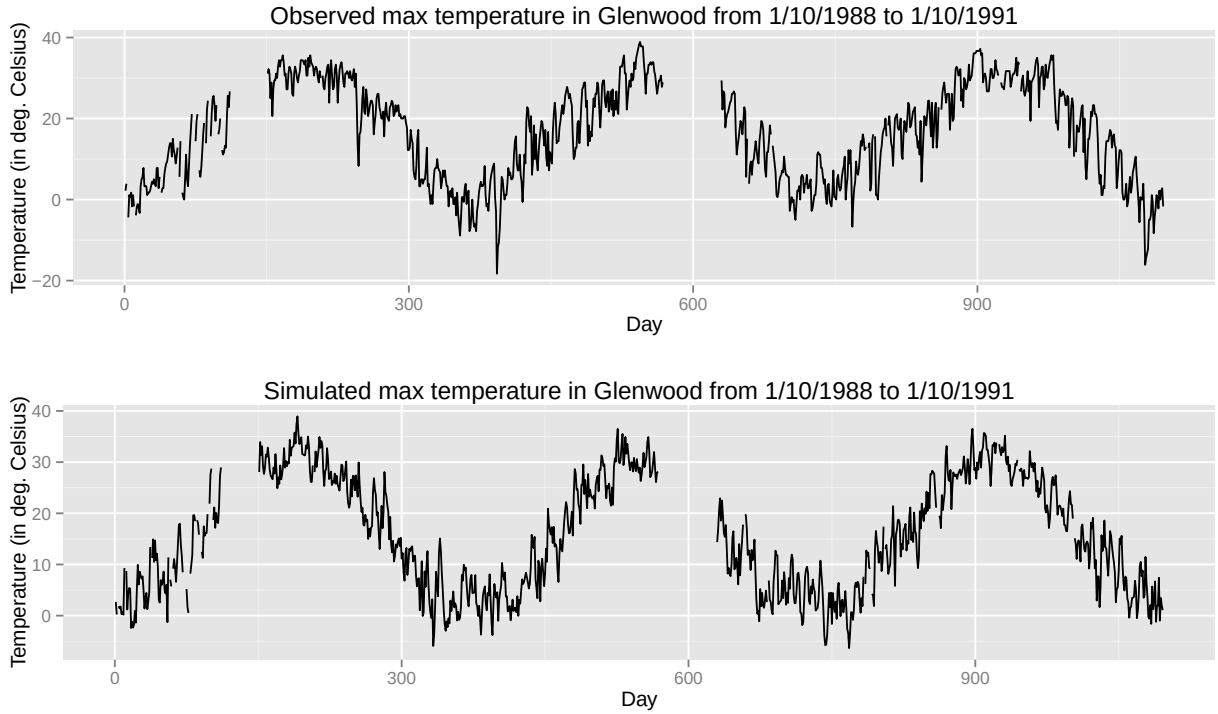


Figure 3.17: Realization of our SWG for daily maximum temperature for Glenwood Springs from January 10, 1988 to January 10, 1991. Note the masking of missing data into our simulations.



covariates would increase the accuracy. In this case, the burden of 11 parameters would coerce the need for an efficient means of performing algorithm 7, such as using a supercomputer, if available, and gathering a very large number of samples (say $> 100,000$). This would likely amend the inconsistencies in some of the spell counts seen in figures 3.11 - 3.16. If these spells are of utmost importance to the model, another option is to modify ϱ to penalize against them in a way that fits the model's needs.

An attractive feature of 7 is its flexibility, and by redefining our model $\mathcal{M}$, penalization criterion ϱ, prior π, and candidate-generating density q, we can create many variations of our implementation to suit the needs of different scenarios. Other possible extensions of these results include modeling daily minimum temperature, simultaneously modeling maximum and minimum temperature as a bivariate process, and generalizing our model over a spatial domain.

At this point, we shift our focus from temperature to precipitation, as SPGs comprise the more sizable portion of the world of SWGs. We will see that the case of precipitation is more sophisticated in comparison, but by applying the same principles of this chapter we can march onward with confidence. The ultimate goal of this thesis is to construct an ABC method for spatiotemporal precipitation occurrence due to its prominence in the literature, but before we arrive there, we will need to investigate the case of single-site precipitation occurrence.

Chapter 4

Simulation of Local Daily Precipitation Occurrence with ABC

4.1 Introduction

In the previous chapter, we demonstrated the effectiveness of ABC as a means of creating a SWG for daily maximum temperature values at a given geographic location. This gives us confidence that we can utilize the same methodology to investigate SWGs for other geospatial quantities, and from this point forward, precipitation will be our main focus. As discussed in the introductory chapter, the need for reliable precipitation models remains a significant one and an extraordinary task. This is mainly because of the discrete-continuous nature of occurrence and intensities, and the difficulties are magnified when incorporating a spatial dependence across a domain. Consequently, this thesis will resist venturing into the realm of modeling intensities, and instead concentrate solely on modeling precipitation occurrence. We will still attempt to create spatiotemporal SPGs across a domain in Chapter 5. For now, we begin our journey with the application of ABC to an SPG for daily local precipitation occurrence.

4.2 Statistical Model

The goal of our model is to capture annual sinusoidal behavior in observed daily precipitation occurrence and produce realistic simulations with matching characteristics. Given location $\mathbf{s} \in D$ and day d, either it will have rained at $\mathbf{s}$ on d, or it will not have rained. Thus, if $Y(\mathbf{s}, d)$ is the occurrence of rain at $\mathbf{s}$ on d, then $Y(\mathbf{s}, d) \sim \text{Bernoulli}(p)$ for some $0 < p < 1$. To keep things

general, we assume that p depends on the location $\mathbf{s}$, and by assumption, p also depends on d. Thus, we have $Y(\mathbf{s}, d) \sim \text{Bernoulli}\,(p(\mathbf{s}, d))$.

The nature of precipitation occurrence as a Bernoulli random variable suggests the use of a generalized linear model in our analysis. We adopt the following approach specified by Chandler et. al. [13] and demonstrated by Kleiber et. al [39]. Generalized linear models are of the form

$$g\left(\mathbf{E}\left[\mathbf{Y}\right]\right) = \mathbf{X}\boldsymbol{\beta} \tag{4.1}$$

for some monotonic link function $g(\cdot)$. Because precipitation occurrence is binary, we will employ logistic regression which is tailored to model random quantities whose outcomes are boolean-valued, i.e., live in the set $\{0, 1\} = \{\text{False}, \text{True}\}$. There are several popular choices of the link function in a logistic regression, including logit, probit, and the complimentary log log function [19]. We elect probit regression for our model, so that the link function $g(\cdot) = \Phi^{-1}(\cdot)$, the inverse of the cdf of the standard normal distribution. Our preference of the probit over the others results from our extension into a spatial domain, which will be explained in the next chapter.

Gabriel and Newmann showed that precipitation occurrence can be successfully modelled using a first-order Markov chain [24], which has been a feature of precipitation occurrence models ever since. Thus, we will follow suit and include the previous day's occurrence as a regressor to our generalized linear model. For other covariates, we elect the two pairs of sinusoidal regressors present in our temperature design matrix described by 3.2. Hence, our model becomes

$$\begin{aligned}\mathbf{E}\left[Y(\mathbf{s}, d)\right] = \Phi\Bigg(\beta_0 + \beta_1 Y(\mathbf{s}, d-1) + \beta_2 \cos\left(\frac{2\pi d}{365}\right) + \beta_3 \sin\left(\frac{2\pi d}{365}\right) \\ + \beta_4 \cos\left(\frac{4\pi d}{365}\right) + \beta_5 \sin\left(\frac{4\pi d}{365}\right)\Bigg)\end{aligned} \tag{4.2}$$

where $\Phi(\cdot)$ is the cumulative distribution function of the standard Normal random variable. For convenience, if $\mathbf{Y} = (Y(\mathbf{s}, 1), \ldots, Y(\mathbf{s}, T))$ represents the random vector of precipitation occurrence for days 1 through T, then we may also write this as

$$\mathbf{E}\left[\mathbf{Y}(\mathbf{s}, d)\right] = \Phi\left(\mathbf{X}\left(\mathbf{s}, d\right)\boldsymbol{\beta}\left(\mathbf{s}\right)\right) \tag{4.3}$$

where

$$\boldsymbol{\beta}(\mathbf{s}) = \begin{pmatrix} \beta_0(\mathbf{s}) \\ \beta_1(\mathbf{s}) \\ \beta_2(\mathbf{s}) \\ \beta_3(\mathbf{s}) \\ \beta_4(\mathbf{s}) \\ \beta_5(\mathbf{s}) \end{pmatrix} \quad \text{and} \quad \mathbf{X} = \begin{pmatrix} 1 & Y_0 & \cos\left(\frac{2\pi}{365}\right) & \sin\left(\frac{2\pi}{365}\right) & \cos\left(\frac{4\pi}{365}\right) & \sin\left(\frac{4\pi}{365}\right) \\ 1 & Y(\mathbf{s},1) & \cos\left(\frac{2\pi\cdot 2}{365}\right) & \sin\left(\frac{2\pi\cdot 2}{365}\right) & \cos\left(\frac{4\pi\cdot 2}{365}\right) & \sin\left(\frac{4\pi\cdot 2}{365}\right) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & Y(\mathbf{s},T-1) & \cos\left(\frac{2\pi\cdot T}{365}\right) & \sin\left(\frac{2\pi\cdot T}{365}\right) & \cos\left(\frac{4\pi\cdot T}{365}\right) & \sin\left(\frac{4\pi\cdot T}{365}\right) \end{pmatrix}.$$

Furthermore, the probability mass function of a Bernoulli$(\Phi(\mathbf{X}\boldsymbol{\beta}))$ random variable is

$$p(y) = \mathsf{P}\left[Y(\mathbf{s},d) = y\right] = \left[\Phi(\mathbf{X}\boldsymbol{\beta})\right]^{y}\left[1-\Phi(\mathbf{X}\boldsymbol{\beta})\right]^{1-y}\mathbf{1}_{[y\in\{0,1\}]}. \tag{4.4}$$

Note the inclusion of Y_0 as the $(1,2)$ entry of our design matrix. This corresponds to an arbitrary initial value of the sequence of occurrences $\mathbf{Y}$, which we usually take to be zero. It is important to note that, as presented, the choice of using a probit regression to model local precipitation occurrence can be seamlessly replaced by a regression based on an alternative logistic link function if the modeler so desired. It is seen that (4.2) includes a pair of higher-order sinusoidal covariates, but analogous to (3.1), one can set $\beta_4 = \beta_5 = 0$ if the need for fewer parameters is present. However, for many locations, the higher-order terms are necessary, and sometimes still do not adequately represent the empirical probability curve of precipitation occurrence throughout the year. An initial scan of the empirical probability of rain for the observed data should suffice in deciding which sinusoidal covariates are appropriate.

4.3 Metrics for Estimating Precipitation Occurrence

Precipitation occurrence values for a number of consecutive days will be a binary sequence of 0's and 1's, and consequently, it can be difficult to construct a statistic that can accurately compare two different sequences of this sort of data. One approach stems from the observation that the nature of precipitation, which our model (4.2) captures, is that rainy days and dry days tend to

be grouped in sequences of values of the same outcome. For example, here is a randomly chosen vector of 40 days of precipitation occurrence data for Iowa Falls, Iowa:

$$0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,1\,1\,0\,0\,0\,0\,1\,1\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,1\,1\,1\,0\,0\,1\,0$$

It is evident that dry days in general lead to the next day being dry, and wet days will often spawn a successive wet day. There are not many instances of 0's and 1's alternating in concession. Moreover, wet spells and dry spells, loosely defined as extended sequences of either precipitation or no precipitation, have strong implications and consequences in many of the fields which require SPGs for their models. An unusually long dry spell will lead to droughts, whereas an unusually long wet spell will lead to floods, among other issues. Because of these factors, we choose to construct our metric to penalize against these criteria, namely, that the number of wet and dry spells of various lengths in the observations and simulations should match closely. Let ζ be a set of numbers that correspond to spell count lengths. Define $\varphi(\mathbf{x};\zeta)$ to be the function that computes the number of wet spells of length in ζ present in the time series $\mathbf{x} = (x_1, \ldots, x_T)$, and define $\psi(\mathbf{x};\zeta)$ to be the function that computes the number of dry spells of length in ζ present in $\mathbf{x}$:

$$\varphi(\mathbf{x};\zeta) = \#\left\{\left(x_j, x_{j+1}, \ldots, x_{j+z}, x_{j+z+1}\right) : x_j = x_{j+z+1} = 0 \,\&\, x_{j+1} = \ldots = x_{j+z} = 1 \,\&\, z \in \zeta\right\} \quad (4.5)$$

$$\psi(\mathbf{x};\zeta) = \#\left\{\left(x_j, x_{j+1}, \ldots, x_{j+z}, x_{j+z+1}\right) : x_j = x_{j+z+1} = 1 \,\&\, x_{j+1} = \ldots = x_{j+z} = 0 \,\&\, z \in \zeta\right\} \quad (4.6)$$

where $\#\{S\}$ represents the cardinality of set S. Our estimator $\widehat{\boldsymbol{\beta}}$ should adhere to the following criterion:

$$\widehat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left\{ \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \left(\frac{c_w}{|\mathcal{Z}_1|} \sum_{\zeta_1 \in \mathcal{Z}_1} \frac{\left|\varphi(\mathbf{s}_{\boldsymbol{\beta}}[t];\zeta_1) - \varphi(\mathbf{o}[t];\zeta_1)\right|}{\max\left\{\varphi(\mathbf{o}[t];\zeta_1), 1\right\}} + \frac{c_h}{|\mathcal{Z}_2|} \sum_{\zeta_2 \in \mathcal{Z}_2} \frac{\left|\psi(\mathbf{s}_{\boldsymbol{\beta}}[t];\zeta_2) - \psi(\mathbf{o}[t];\zeta_2)\right|}{\max\left\{\psi(\mathbf{o}[t];\zeta_2), 1\right\}} \right) \right\} \quad (4.7)$$

Here, $\mathcal{T}$ is a partition of the months of the year, $\mathcal{Z}_1$ is a set of sets of wet spell lengths (e.g. $\{\{1,2,3\},\{4,5,6\},\{7,8,9\}\}$), $\mathcal{Z}_2$ is a set of sets of dry spell lengths, and the notation $\mathbf{x}[t]$ represents the time series $\mathbf{x}$ subsetted by the set of time points t. We would like to scale by the number of dry/wet spells in the observations, but it is possible that the particular isolated sequence $\mathbf{o}[t]$

does not have any of the given amount ζ_i, so we divide by $\max\{\varphi(\mathbf{o}[t];\zeta_i),1\}$ to ensure there is no division by zero. Furthermore, we allow $\mathcal{Z}_1$ and $\mathcal{Z}_2$ to be sets of sets to retain generality as well as allow for a faster algorithmic runtime in practice.

With this formulation of our estimator $\widehat{\boldsymbol{\beta}}$, we are in position to formalize our metric to estimate local precipitation occurrence for a site $\mathbf{s}$:

$$\begin{aligned}\varrho(\mathcal{D},\mathcal{D}') := \frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}}\Bigg(&\frac{1}{|\mathcal{Z}_1|}\sum_{\zeta_1\in\mathcal{Z}_1}\frac{|\varphi(\mathcal{D}[t];\zeta_1)-\varphi(\mathcal{D}'[t];\zeta_1)|}{\max\{\varphi(\mathcal{D}[t];\zeta_1),1\}}\\&+\frac{1}{|\mathcal{Z}_2|}\sum_{\zeta_2\in\mathcal{Z}_2}\frac{|\psi(\mathcal{D}[t];\zeta_2)-\psi(\mathcal{D}'[t];\zeta_2)|}{\max\{\psi(\mathcal{D}[t];\zeta_2),1\}}\Bigg)\end{aligned} \tag{4.8}$$

One immediate quality of our precipitation metric is its similarity to the metric used in algorithm 8. Indeed, this penalization based on relative error of a certain function of the data will also be used in the next chapter concerning precipitation occurrence over a correlated spatial domain. While our methods are defined using specific, contrived functions, it should be noted that this procedure can be abstracted as needed, and provides an intuitive and effective method of comparing two datasets. For example, if the modeler were more interested in the total number of rainy days per month, one could replace our functions φ and ψ with a function that counts the number of days where precipitation occurred, and then use the relative error per month as the metric for the ABC criterion.

4.4 An Analytic Solution of the True Posterior

While ABC bypasses the need to evaluate a likelihood function, we can actually deduce a closed-form expression of the likelihood for our model. This will in turn provide an expression of the true posterior given our prior distribution. We will use this to check our approximations from the ABC algorithm and show that ABC actually does surprisingly well in representing the true posterior. In what follows, let $\mathbf{o} = (o_1, \cdots, o_T)$ be a vector of observed precipitation data. Also, for the sake of simplicity, we write $\mathsf{P}[o_1|\boldsymbol{\theta}] = p_0$, the constant that corresponds to the probability of rain on the very first day. This constant must be assigned an arbitrary value since our model needs the previous day's occurrence to compute this probability; we usually choose $p_0 = 0.5$.

Algorithm 8: ABC-MCMC for daily local precipitation occurrence parameter estimation

Input : Observed data $\mathcal{D}$, arbitrary initial value $\boldsymbol{\beta}_0$, candidate-generating density $q(\cdot,\cdot)$, tolerance $\varepsilon > 0$, desired number of samples N

Output: A vector V of samples approximately distributed from target distribution $f(\boldsymbol{\beta}|\mathcal{D})$

Define $\mathcal{M}$ using (4.2)

Define ϱ as follows:

$$\varrho(\mathcal{D},\mathcal{D}') := \frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}}\left(\frac{1}{|\mathcal{Z}_1|}\sum_{\zeta_1\in\mathcal{Z}_1}\frac{|\varphi(\mathcal{D}[t];\zeta_1)-\varphi(\mathcal{D}'[t];\zeta_1)|}{\max\{\varphi(\mathcal{D}[t];\zeta_1),1\}} + \frac{1}{|\mathcal{Z}_2|}\sum_{\zeta_2\in\mathcal{Z}_2}\frac{|\psi(\mathcal{D}[t];\zeta_2)-\psi(\mathcal{D}'[t];\zeta_2)|}{\max\{\psi(\mathcal{D}[t];\zeta_2),1\}}\right)$$

$V \leftarrow \{\}$
$\boldsymbol{\beta} \leftarrow \boldsymbol{\beta}_0$
while $|V| < N$ **do**
  generate $\boldsymbol{\beta}_{\text{candidate}} \sim q(\boldsymbol{\beta},\cdot)$
  simulate $\mathcal{D}'$ from $\mathcal{M}$ parametrized by $\boldsymbol{\beta}_{\text{candidate}}$
  if $\varrho(\mathcal{D},\mathcal{D}') < \varepsilon$ **then**
    $u \sim \mathcal{U}(0,1)$
    if $u < \min\left\{\frac{\pi(\boldsymbol{\beta}_{\text{candidate}})q(\boldsymbol{\beta}_{\text{candidate}},\boldsymbol{\beta})}{\pi(\boldsymbol{\beta})q(\boldsymbol{\beta},\boldsymbol{\beta}_{\text{candidate}})},1\right\}$ **then**
      $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta}_{\text{candidate}}$
    end
  end
  append $\boldsymbol{\beta}$ to V
end

First, we seek the aid of the multiplicative rule of probability, which states that, for some events $\{A_1, \cdots, A_n\}$,

$$\begin{aligned}\mathbf{P}\left[\bigcap_{i=1}^{n} A_i\right] &= \mathbf{P}\left[A_1\right]\mathbf{P}\left[A_2|A_1\right]\mathbf{P}\left[A_3|A_1 \cap A_2\right] \times \cdots \times \mathbf{P}\left[A_n \middle| \bigcap_{i=1}^{n-1} A_i\right] \\ &= \mathbf{P}\left[A_1\right]\prod_{i=2}^{n}\mathbf{P}\left[A_i \middle| \bigcap_{j=1}^{i-1} A_j, \boldsymbol{\theta}\right].\end{aligned}$$

In terms of our likelihood function $\mathbf{P}\left[\mathbf{o}|\boldsymbol{\theta}\right] = L\left(\boldsymbol{\theta}|\mathbf{o}\right)$, we see that

$$\mathbf{P}\left[\mathbf{o}|\boldsymbol{\theta}\right] = \mathbf{P}\left[o_1|\boldsymbol{\theta}\right]\prod_{d=2}^{T}\mathbf{P}\left[o_i \middle| \bigcap_{j=1}^{d-1} o_j, \boldsymbol{\theta}\right] \tag{4.9}$$

However, by the Markov property of our precipitation chain, each occurrence of precipitaiton only depends on the precipitation of the previous day. Thus, this leaves

$$\mathbf{P}\left[\mathbf{o}|\boldsymbol{\theta}\right] = \mathbf{P}\left[o_1|\boldsymbol{\theta}\right]\prod_{d=2}^{T}\mathbf{P}\left[o_d|o_{d-1}, \boldsymbol{\theta}\right]$$

Next we note that by 2.4,

$$\log f\left(\boldsymbol{\theta}|\mathbf{o}\right) \propto \log\left(L(\boldsymbol{\theta}|\mathbf{o})\pi(\boldsymbol{\theta})\right) \implies \log f\left(\boldsymbol{\theta}|\mathbf{o}\right) = \log L(\boldsymbol{\theta}|\mathbf{o}) + \log \pi(\boldsymbol{\theta}) + C_1 \tag{4.10}$$

for some constant $C_1 \in \mathbb{R}$. Isolating the term with the log of the likelihood, we find that

$$\begin{aligned}\log L(\boldsymbol{\theta}|\mathbf{o}) &= \log\left(\mathbf{P}\left[\mathbf{o}|\boldsymbol{\theta}\right]\right) \\ &= \log\left(\mathbf{P}\left[o_1|\boldsymbol{\theta}\right]\prod_{d=2}^{T}\mathbf{P}\left[o_d|o_{d-1}, \boldsymbol{\theta}\right]\right) \\ &= \log\left(\mathbf{P}\left[o_1|\boldsymbol{\theta}\right]\right) + \sum_{d=2}^{T}\log\left(\mathbf{P}\left[o_d|o_{d-1}, \boldsymbol{\theta}\right]\right) \\ &= \log p_0 + \sum_{d=2}^{T}\log\left(\Phi\left(\boldsymbol{\theta}^\top \mathbf{X}\right)^{o_d}\left[1 - \Phi\left(\boldsymbol{\theta}^\top \mathbf{X}\right)\right]^{1-o_d}\right) \\ &= \sum_{d=2}^{T}\left\{o_d \log\left(\Phi\left(\boldsymbol{\theta}^\top \mathbf{X}\right)\right) + (1 - o_d)\log\left(1 - \Phi\left(\boldsymbol{\theta}^\top \mathbf{X}\right)\right)\right\} + \log p_0 \\ &= \sum_{d=2}^{T}\left\{\mathbb{1}_{[o_d=1]} \log\left(\Phi\left(\boldsymbol{\theta}^\top \mathbf{X}\right)\right) + \mathbb{1}_{[o_d=0]}\log\left(1 - \Phi\left(\boldsymbol{\theta}^\top \mathbf{X}\right)\right)\right\} + \log p_0\end{aligned}$$

Plugging this into (4.10) yields

$$\log f(\boldsymbol{\theta}|\mathbf{o}) \propto \sum_{d=2}^{T}\left\{\mathbb{1}_{[o_d=1]} \log\left(\Phi\left(\boldsymbol{\theta}^\top \mathbf{X}\right)\right) + \mathbb{1}_{[o_d=0]}\log\left(1 - \Phi\left(\boldsymbol{\theta}^\top \mathbf{X}\right)\right)\right\} + \log \pi(\boldsymbol{\theta}) + C \tag{4.11}$$

where $C = C_1 + \log p_0$. Moreover, since we are using uniform, independent priors, we can simplify the expression for $\log \pi(\boldsymbol{\theta})$:

$$\begin{aligned}\log \pi(\boldsymbol{\theta}) &= \log \prod_{i=1}^{k} \pi_i(\theta_i) \\ &= \sum_{i=1}^{k} \log\left(\frac{1}{b_i - a_i}\right) \\ &= -\sum_{i=1}^{k} \log(b_i - a_i)\end{aligned}$$

Therefore, in our case, we arrive at the following relationship:

$$\log f(\boldsymbol{\theta}|\mathbf{o}) = \sum_{d=2}^{T} \left\{ \mathbb{1}_{[o_d=1]} \log\left(\Phi\left(\boldsymbol{\theta}^\top \mathbf{X}\right)\right) + \mathbb{1}_{[o_d=0]} \log\left(1 - \Phi\left(\boldsymbol{\theta}^\top \mathbf{X}\right)\right) \right\} - \sum_{i=1}^{k} \log(b_i - a_i) + C.$$

Keeping in mind that C is simply a normalizing constant that causes $f(\boldsymbol{\theta}|\mathbf{o})$ to integrate to unity, we have identified a closed-form expression for our likelihood function in the case of local precipitation occurrence. Furthermore, its tractability will allow us assess the validity of the approximate densities given by our ABC algorithm.

4.5 Numerical Results for Bonny Dam, Colorado

We shift our attention to Bonny Dam, Colorado, for which there is plenty of precipitation data supplied by the GHCND. Moreover, we will see that the data is adequately modeled only with the inclusion of higher-order sinusoidal covariates. We set our priors to be uniform, where the intercept prior ranges between -1.5 and 1.5, the second (autoregressive) prior ranges from 0 to 1, and the remaining priors range from -0.5 to 0.5. Moreover, we must define our sets of wet and dry spell counts for our ϱ function. We opt for

$$\mathcal{Z}_1 = \{\{1\}, \{2,3\}, \{4,5\}, \{6,7\}, \{8,9\}, \{10,11\}, \{12,13,\dots\}\}$$

for our wet spells and

$$\begin{aligned}\mathcal{Z}_2 = \{ &\{1,2,3,4\}, \{5,6,7,8\}, \{9,10,11,12\}, \{13,14,15,16\}, \{17,18,19,20\}, \{21,22,23,24\}, \\ &\{25,26,27,28,29\}, \{30,31,\dots\} \}\end{aligned}$$

to strike a balance of feasibility and comprehensiveness. To start, we gather about 2,300 samples (after omitting the burn-in) from the posterior distribution using the Metropolis-Hastings algorithm. For our ABC algorithm, we set $\varepsilon = 1.09$, and obtain $\boldsymbol{\beta}_0$ from a grid-search based on our ϱ criterion.

After collecting 10,000 ABC samples (after omitting the burn-in), we can examine the overlap of the priors, true posteriors, and ABC-approximated posteriors, shown in figure 4.1. It appears that the ABC posteriors replicate the true posteriors adequately and exhibit high certainty within the uniform priors. To further assess their accuracy, we can compute the mean functions parametrized by our parameter estimates, using the empirical mean of each posterior density, and compare them to the empirical probability of rain based on the observed data, as seen in figure 4.2. Indeed, both the true posterior and ABC mean functions present a credible description of the data's trend. As for a more explicit look at the behavior of our SPG, we can also inspect a plot of the empirical probabilities of precipitation for the observations and simulated values, shown in figure 4.3. The probabilities in fact line nicely although the ABC probabilities exhibit slightly less variability overall. It should be noted that the empirical probability of rain was not a feature of our metric ϱ, but we still get matching empirical probabilities in the observations and simulations. In other words, we got the probability trends to match "for free", which is one of the strongest arguments in favor of using ABC for our estimation. Another statistic to peruse is the monthly empirical standard deviation of the observations, ABC posterior, and true posterior, shown in figure 4.4. These statistics again match well despite the absence of standard deviation comparisons in the penalization criterion within ϱ.

Finally, we take a look at the wet and dry spell counts for the observations and ABC simulation, displayed in figures 4.5 and 4.6 respectively. Because we are particularly interested in high wet and dry spells, we also inspect the log of these counts so we can get a clear visualization of their behavior in the tail. As desired, these spell counts match to a very high degree across the entire range of count lengths, almost to an uncanny extent. The observed and simulated wet spell counts coincidentally end at the length of eight, a feature we did not include in our metric.

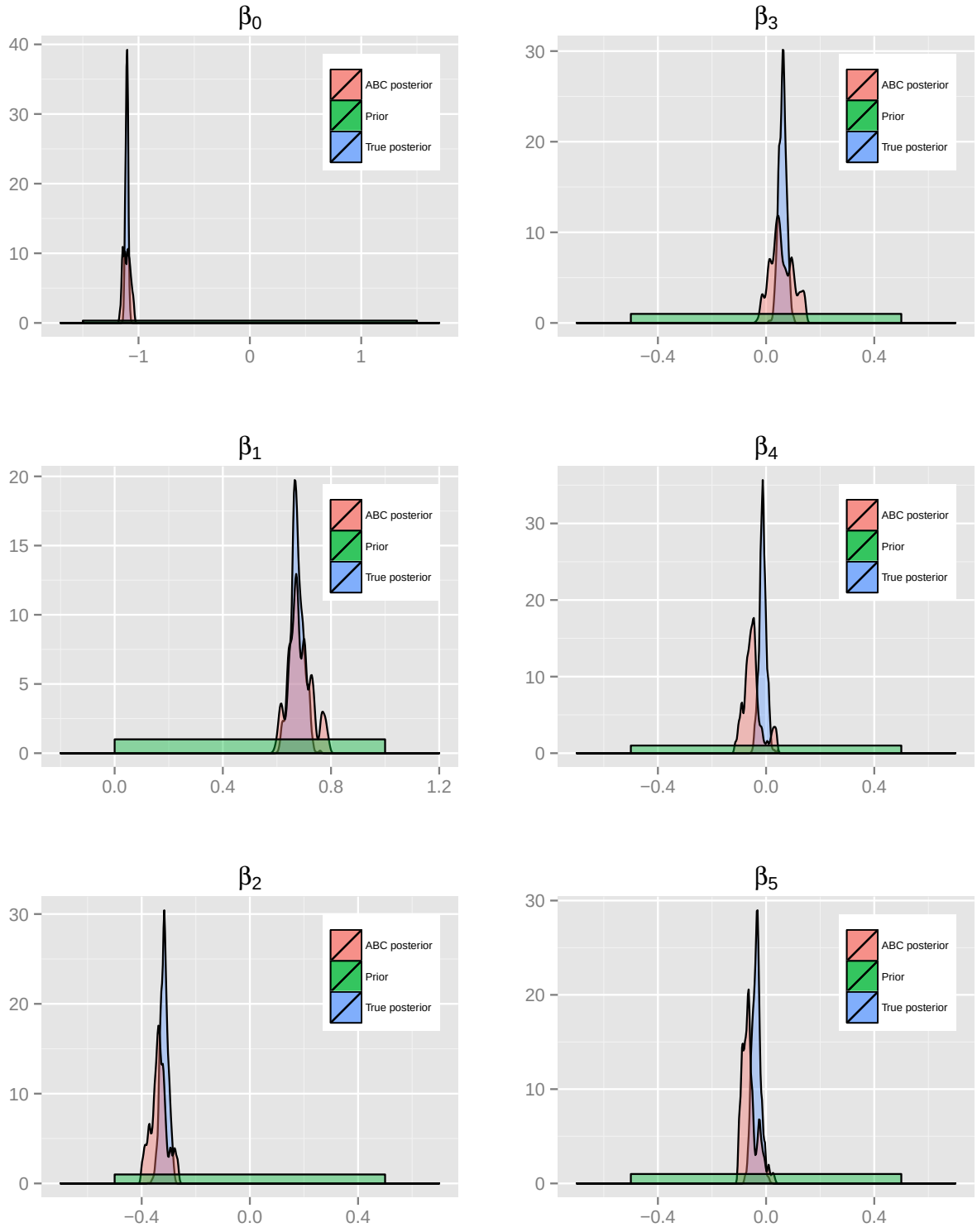


Figure 4.1: The prior, true posterior (via MCMC sampling), and approximate posterior densities for each mean parameter β_i

Figure 4.2: The mean function of precipitation occurrence for the ABC and true posterior estimates as well as the observed empirical probability of precipitation for Bonny Dam, Colorado.

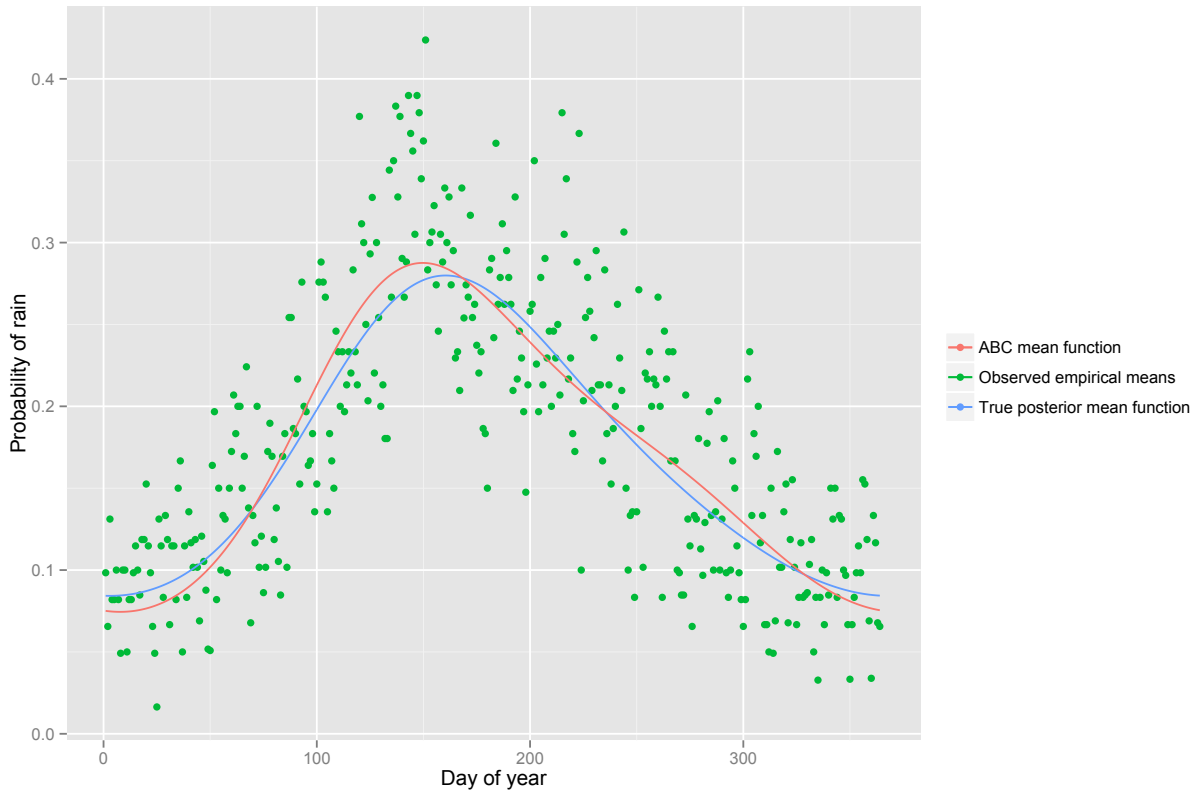


Figure 4.3: The empirical probability of precipitation for the observations and ABC simulation for Bonny Dam, Colorado.

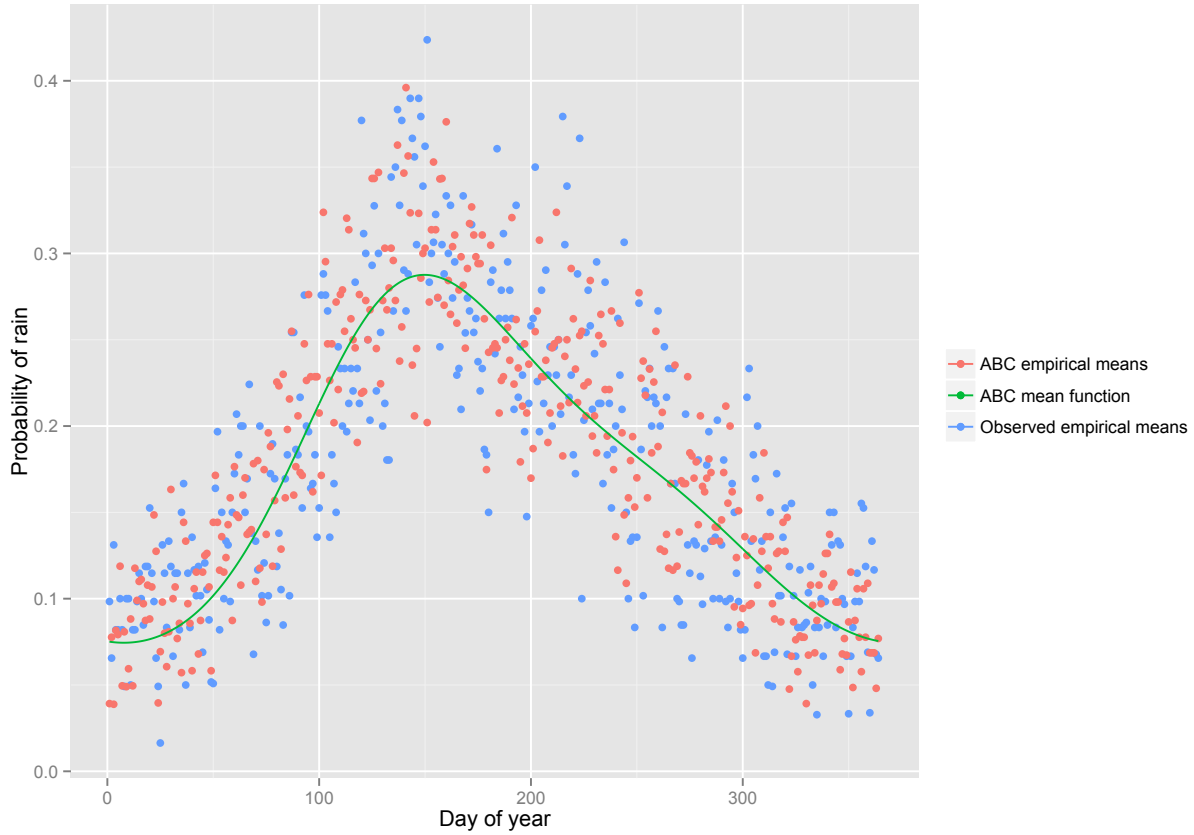
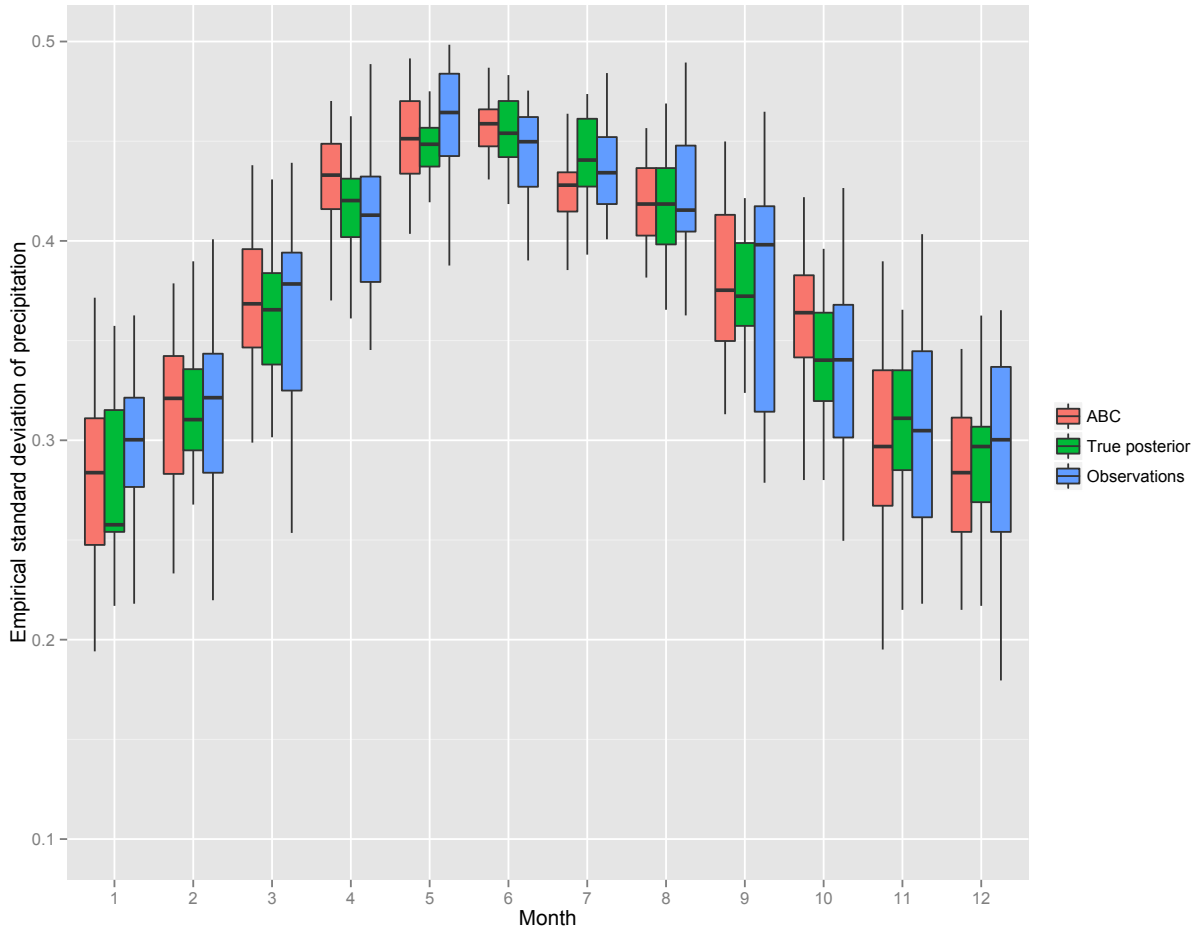


Figure 4.4: The empirical standard deviation of precipitation (in mm) for the ABC simulation, true posterior simulation, and observations for Bonny Dam, Colorado.



Overall, the consistency of our model with the true precipitation occurrence process seems to be well-testified by these summaries.

Discussion

It turns out that algorithm 8 successfully characterizes the precipitation of many different locations across Colorado, given that their empirical probabilities sufficiently adhere to the higher-order sinusoidal covariates given in (4.2). Indeed, looking at each of our summary plots, it is difficult to find much to complain about concerning the efficacy of algorithm 8. In fact, the model succeeds for each location in Colorado that we tried, to the extent that the precipitation data does not strongly disobey (4.2). Regardless, there are a handful of improvements and extensions to be investigated. The most prominent deficiency is the absence of precipitation intensities given precipitation occurrence, which is a key component of many SPGs. Since intensities are usually modeled with an exponential or gamma distribution, there is high credibly in the hypothesis that ABC could be used to estimate the parameters present in the corresponding models. Furthermore, since our model contains just 6 parameters, we can try adding another set of higher-order sinusoidal covariates, namely $\cos(8\pi d/365)$ and $\sin(8\pi d/365)$, to attempt an even better fit, especially for locations whose precipitation behavior is erratic.

With favorable results for the cases of local daily maximum temperature simulation and local daily precipitation occurrence simulation, we seek to extend our techniques into a general spatiotemporal domain. Since spatiotemporal temperature simulation has been tackled extensively, whereas spatiotemporal precipitation simulation tends to be more problematic, we will focus only on the case of precipitation. However, at this point, the prospect of using ABC to produce local simulations for most meteorological variables is convincing. We hope that our results will pave the way for the integration of ABC in the quest for better SWGs as a promising and mighty set of tools.

Figure 4.5: Wet spell counts and the logarithm of wet spell counts by spell length for the observations and ABC simulation.

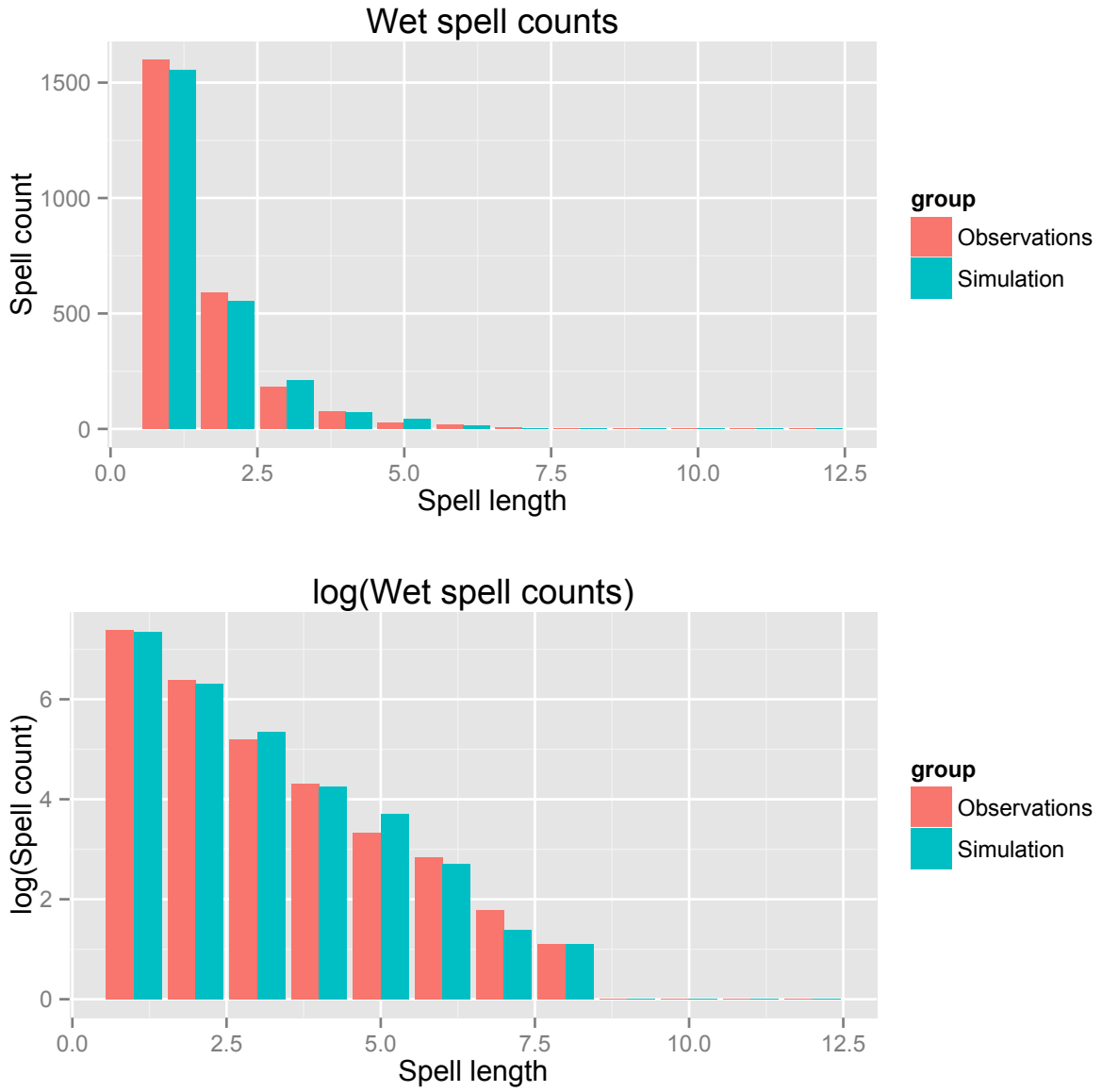
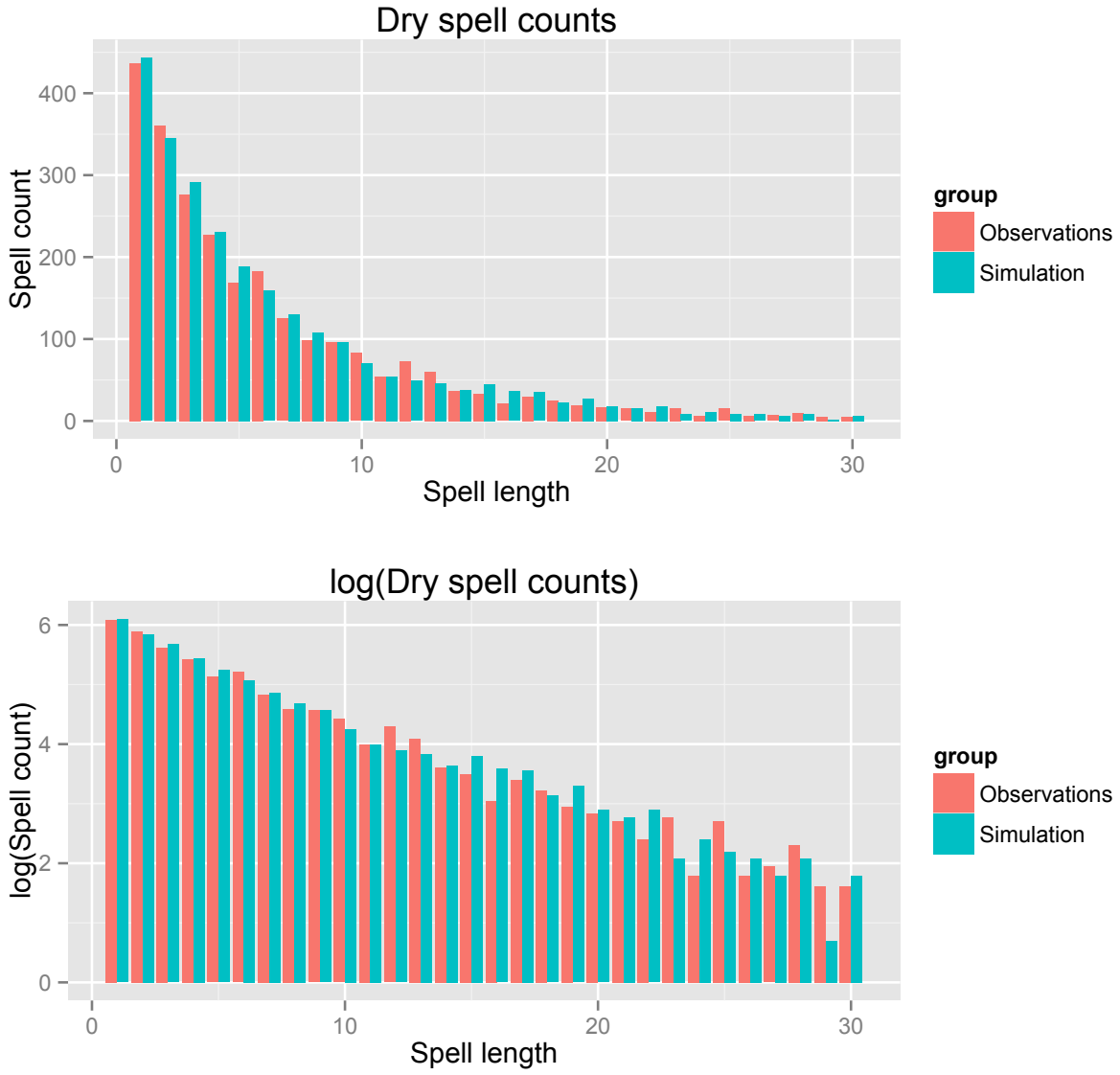


Figure 4.6: Dry spell counts and the logarithm of dry spell counts by spell length for the observations and ABC simulation.



Chapter 5

Simulation of Daily Precipitation Occurrence across a Spatial Domain with ABC

5.1 Introduction

At this point, we have succeeded in constructing ABC methods for stochastic weather generation of local daily maximum temperature and precipitation occurrence, two geospatial quantities with contrasting behavior. Our next objective is to extend our methodology to estimate SPGs for a correlated spatial domain. The generalization of a stochastic process into a spatiotemporal one requires care and sophistication, so we retain our focus on precipitation occurrence without intensity to keep our exploration grounded and viable.

When it comes to describing and modeling spatial phenomena, stochastic processes play a prominent role. In particular, Gaussian process (GP) theory has proven to be incredibly effective in spatial modeling, and consequently, we will employ GPs in our aspiration for an effectual SWG for spatially correlated precipitation occurrence. For a spatial domain $\mathcal{S} \subset \mathbb{R}^d$, a stochastic process $Z(\mathbf{s})$ is a Gaussian process if all finite-dimensional distributions are distributed as multivariate normal random variables:

$$\begin{pmatrix} Z(\mathbf{s}_1) \\ \vdots \\ Z(\mathbf{s}_n) \end{pmatrix} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \quad \forall\, \mathbf{s}_1, \ldots, \mathbf{s}_n \in \mathcal{S}. \tag{5.1}$$

One reason GPs are so attractive is that their entire probability distribution is determined by its mean $\boldsymbol{\mu} \in \mathbb{R}^n$ and covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{n \times n}$, so that very general spatial behavior can be described with relatively few parameters. We can reconcile the advantages of GP theory with the

binary nature of precipitation occurrence by using a latent GP for our precipitation occurrence process introduced by Wilks (1998) [78]. Moreover, we will show that our spatially extended model reduces to a close approximation of (4.2) at a single location, and can be made to reduce exactly using a straightforward modification. Using similar principles as in the previous two chapters, we will construct metrics that allow ABC estimation of both the mean and covariance parameters, and evaluate the results against observed precipitation data across the state of Iowa.

5.2 Statistical Model

While Gaussian process theory is thriving in the field of spatial analysis, there is less to be said about the simulation of a binary equivalent to a GP. That is, while GPs by definition are real-valued, it is more difficult to coerce its values to be in the discrete set $\{0, 1\}$. However, we can use the technique of Kleiber et. al. which allows for the simulation spatial binary values using a Gaussian process [39]. First, simulate $W(\mathbf{s}, d) \sim \text{GP}(\mu, C(\cdot)) = \text{GP}(\mathbf{X}\boldsymbol{\beta}, C(\cdot))$. Then, define $Y(\mathbf{s}, d) = \mathbf{1}_{[W(\mathbf{s},d)\geq 0]}$, i.e., $Y = 1$ at a location if $W \geq 0$ at that location, and $Y = 0$ otherwise. Observe that at a single location $\mathbf{s}$ and day d, a Gaussian Process reduces to a normal random variable of the form $W = W(\mathbf{s}, d) = \mu(\mathbf{s}, d) + Z + \varepsilon$, with stochastic component $Z \sim \mathcal{N}(0, 1)$ and nugget effect $\varepsilon \sim \mathcal{N}(0, \tau^2)$, a component of small-scale variability. We then see that

$$
\begin{aligned}
\mathbf{P}\left[Y(\mathbf{s}, d) = 1\right] &= \mathbf{P}\left[W(\mathbf{s}, d) \geq 0\right] \\
&= \mathbf{P}\left[\mu(\mathbf{s}, d) + Z(\mathbf{s}, d) + \varepsilon \geq 0\right], \\
&= \mathbf{P}\left[Z(\mathbf{s}, d) + \varepsilon \geq -\mu(\mathbf{s}, d)\right] \\
&= 1 - \mathbf{P}\left[Z(\mathbf{s}, d) + \varepsilon \leq -\mu(\mathbf{s}, d)\right] \\
&= \mathbf{P}\left[Z(\mathbf{s}, d) + \varepsilon \leq \mu(\mathbf{s}, d)\right] \\
&= \mathbf{P}\left[Z(\mathbf{s}, d) + \varepsilon \leq \mathbf{X}(\mathbf{s}, d)\boldsymbol{\beta}(\mathbf{s})\right] \\
&= \mathbf{P}\left[\mathcal{N}(0, 1) \leq \frac{\mathbf{X}(\mathbf{s}, d)\boldsymbol{\beta}(\mathbf{s})}{\sqrt{1 + \tau^2}}\right] \\
&= \Phi\left(\frac{\mathbf{X}(\mathbf{s}, d)\boldsymbol{\beta}(\mathbf{s})}{\sqrt{1 + \tau^2}}\right).
\end{aligned}
$$

Note that we used the definition of a Gaussian process Z, namely, that $Z(\mathbf{s}) \sim \mathcal{N}(0,1)$ at any given location $\mathbf{s} \in \mathcal{S}$. Since it is common in practice that $\tau^2 \ll 1$, we have $\sqrt{1+\tau^2} \approx 1$, and so

$$\mathbf{P}\left[Y(\mathbf{s},d)=1\right] \approx \Phi\left(\mathbf{X}(\mathbf{s},d)\boldsymbol{\beta}(\mathbf{s})\right)$$

which is approximately our generalized linear model for a single location given in (4.3). That is, our thresholded Gaussian process produces essentially the same precipitation occurrence behavior for individual locations that our methods in the previous chapter produced. For models which must exactly recover the reduction to single locations, one can instead threshold based on

$$Y(\mathbf{s},d)=1 \quad \text{if} \quad \sqrt{1+\tau^2}\,(Z+\varepsilon)+\boldsymbol{\mu}(\mathbf{s},d) \geq 0$$

which yields $\mathbf{P}\left[Y(\mathbf{s},d)=1\right] = \Phi(\mathbf{X}(\mathbf{s},d)\boldsymbol{\beta}(\mathbf{s}))$ as desired. It should be noted that the true value of τ^2 cannot be known, but must be estimated, and so this equality in practice will always be approximate. Nonetheless, this procedure allows us to use the features of a Gaussian process for the simulation of correlated precipitation occurrence over a spatial domain.

With this analysis, we are ready to formalize our extension of model (4.2) to a general spatial domain. Given a spatial domian $\mathcal{S} \subset \mathbb{R}^2$ we wish to simulate a stochastic process Y of daily precipitation occurrence over $\mathcal{S}$. Following our discussion above, for each $\mathbf{s} \in \mathcal{S}$ and $d \in \mathbb{Z}^+$, define

$$\begin{aligned} W(\mathbf{s},d) &\sim \mathrm{GP}\left(\mu\left(\mathbf{s},d;\boldsymbol{\beta}\right), C\left(\mathbf{h},d;\boldsymbol{\alpha}\right) + \tau^2\mathbf{1}_{[\mathbf{h}=\mathbf{0}]}(\mathbf{h})\right) \\ Y(\mathbf{s},d) &= \mathbf{1}_{[W(\mathbf{s},d)\geq 0]} \end{aligned} \tag{5.2}$$

where $C(\mathbf{h},d) = \mathrm{Cov}\left[\mathbf{s},\mathbf{s}+\mathbf{h}\right]$ for any location $\mathbf{s} \in \mathcal{S}$, day d, and shift vector $\mathbf{h}$, and $\mathbf{1}_{[A]}$ is the indicator function which is 1 if A is a true proposition and 0 otherwise. In our newfound jargon, $W \in \mathbb{R}$ is a latent Gaussian Process with mean μ and covariance function C, equipped with an assumed nugget effect τ^2. $Y \in \{0,1\}$ is our thresholded process induced by the latent process W. As in the case for local precipitation occurrence, we assume a mean with sinusoidal and autore-

gressive covariates, namely

$$\begin{aligned}\mu(\mathbf{s}, d; \boldsymbol{\beta}) = \beta_0 + \beta_1 Y(\mathbf{s}, d-1) + \beta_2 \cos\left(\frac{2\pi d}{365}\right) + \beta_3 \sin\left(\frac{2\pi d}{365}\right) \\ + \beta_4 \cos\left(\frac{4\pi d}{365}\right) + \beta_5 \sin\left(\frac{4\pi d}{365}\right),\end{aligned} \tag{5.3}$$

Furthermore, following Kleiber et. al. [40], we will assume an isotropic exponential covariance with a time-dependent range, causing C to take the form

$$C(\mathbf{h}, d; \boldsymbol{\alpha}) = \exp\left(-\frac{\|\mathbf{h}\|}{A(t)}\right).$$

Because we surmise that the spatial correlation of precipitation occurrence oscillates throughout the year, we can include sinusoidal terms within $A(t)$, and because $A(t)$ is required to be positive, we will exponentiate our sine terms, so that

$$A(t) = \exp\left\{\alpha_0 + \alpha_1 \cos\left(\frac{2\pi d}{365}\right) + \alpha_2 \sin\left(\frac{2\pi d}{365}\right) + \alpha_3 \cos\left(\frac{4\pi d}{365}\right) + \alpha_4 \sin\left(\frac{4\pi d}{365}\right)\right\}.$$

However, between μ, $A(t)$ and τ, our model currently boasts 12 parameters, which turns out to be too computationally expensive for our simulations. Hence, we set $\alpha_3 = \alpha_4 = 0$ so that

$$C(\mathbf{h}, d; \boldsymbol{\alpha}) = \exp\left(-\frac{\|\mathbf{h}\|}{\exp\left(\alpha_0 + \alpha_1 \cos\left(\frac{2\pi d}{365}\right) + \alpha_2 \sin\left(\frac{2\pi d}{365}\right)\right)}\right) \tag{5.4}$$

which we will hope to be tractable and still yield adequate approximations to the observed data.

Before we propose our ABC metrics for the case of spatial precipitation occurrence, we will need to discuss some prerequisite information concerning our data analysis and simulation methods. In particular, we explore the estimation of spatial dependence with variograms, and discuss the elements of using Cholesky decompositions for streamlined Gaussian process simulation.

5.3 Spatial Estimation with Variograms

In upgrading to a spatial domain for precipitation occurrence, our main goal is to understand the spatial dependence of precipitation occurrence over S if we wish to produce realistic simulations across its entirety. This presents a great challenge due to intrinsic uncertainties in and

obstreperous qualities of spatial covariance and dependence. One function that assesses the degree of spatial dependence of a stochastic process is known as the theoretical variogram, defined as

$$\gamma(\mathbf{u}) = \frac{\text{Var}\left[Y(\mathbf{x}+\mathbf{u}) - Y(\mathbf{x})\right]}{2}$$

[16]. This function is particularly popular in the estimation of paramters for geostatistical models [18]. The variogram can be seen as a measure of the variance of a stochastic process Y as a function of the location and magnitude of the shift vector $\mathbf{u}$. Intuitively, γ should be small for small distances and large for large distances, since spatial dependence should be strong for nearby locations and week for distant ones. At the moment, we do not have a closed-form solution for $\text{Var}\left[Y(\mathbf{s}_1+\mathbf{u}) - Y(\mathbf{s})\right]$ for our model, so we turn to estimation. For the case of a constant mean function $\mu(\mathbf{x}) = \mu$, a nonparametric estimator is the empirical variogram is given by

$$\widehat{\gamma(\mathbf{u})} = \frac{1}{2\left|N(\mathbf{u})\right|}\sum_{i=1}^{n}\sum_{j\neq i}\left[y(\mathbf{x}_i) - y(\mathbf{x}_j)\right]^2, \tag{5.5}$$

where $N(\mathbf{u})$ is a user-defined neighborhood about $\mathbf{u}$, and $|N(\mathbf{u})|$ denotes the number of distinct elements $\mathbf{u}_j \in N(\mathbf{u})$. This "binning" of variogram values aims to smooth out the curve to prevent noisy point clouds whose structure is difficult to analyze. The choice of the bins is arbitrary, but the rule of thumb is to include at least 30 pairs of observations per bin (i.e. $|N(\mathbf{u})| > 30$). Moreover, for $|\mathbf{u}| \gg 0$, so few pairs are used in estimating $\widehat{\gamma}(\mathbf{u})$ that it is unreliable, so the modeler will often restrict distances so that $\mathbf{u} \leq \frac{1}{2}\max\left\|\mathbf{s}_i - \mathbf{s}_j\right\|$. Despite being an ad-hoc statistic, the empirical variogram has been employed successfully for many statistical analyses. One fallback of the binned variogram is its sensitivity to outliers; to address this, Cressie and Hawkins identified the following more robust version,

$$\widehat{\gamma_{\text{Robust}}}(\mathbf{u}) = \frac{1}{0.914 + \frac{0.988}{|N(\mathbf{u})|}}\left[\frac{1}{|N(\mathbf{u})|}\sum_{\left\|\mathbf{s}_i-\mathbf{s}_j\right\|\in N(\mathbf{u})}\left(y(\mathbf{s}_i) - y(\mathbf{s}_j)\right)\right]^4$$

which is widely used as a safer alternative [17]. Both empirical variograms can be found in most spatial statistical software packages, including our choice of the `fields` package for `R`. Because

our model (5.2) assumes a time-dependent spatial correlation, our empirical variogram will depend on the location $\mathbf{s}$ as well as the day of the year d, so that $\widehat{\gamma(\mathbf{u})} = \widehat{\gamma(\mathbf{u}, d)}$.

Because our empirical variograms are assumed to have intra-annual variability, we will need to estimate its behavior at different instances of time throughout the year. This will require several trade offs. We want to choose enough days in the year to get an accurate account of its behavior, while still maintaining tractability for ABC. This inspires us to construct an monthly aggregate variogram $\widehat{\Gamma}$ as a function of month m as well as $\mathbf{u}$, defined as

$$\widehat{\Gamma}(m, \mathbf{u}) = \frac{1}{|\mathcal{Y}|} \sum_{y \in \mathcal{Y}} \frac{1}{2\omega} \sum_{d=\delta_{m,y}-\omega}^{\delta_{m,y}+\omega} \widehat{\gamma}(\mathbf{u}, d) \tag{5.6}$$

where $\mathcal{Y}$ is the set of sample years, $\delta_{m,y}$ is the sample center day of the month m in year y, and ω is the forwards and backwards offset for binning. So, for each month m, we loop through all of our sample years and select a sample center day for that month, $\delta_{m,\cdot}$, which is arbitrarily chosen, but should be consistent. Next, we compute a sort of "clustered variogram" for δ_m, meaning that we look at a few days before and after this day to smooth the variogram data from an otherwise noisy set of points. This is represented by $\delta_m \pm \omega$; we set $\omega = 5$ which yields a group of 10 consecutive days for each month. Finally, we divide by the total number of aggregated variograms, i.e., $2\omega|\mathcal{Y}|$, to ensure that the values of $\widehat{\Gamma}$ are on the same scale as $\widehat{\gamma}$.

5.4 Gaussian Process Simulation

Our model (5.2) relies on a latent Gaussian process, and while there is an air of pleasant simplicity about the definition of a GP, the act of its simulation will require more care than its straightforward definition may suggest. The simplest way to simulate a GP is to explicitly decompose it as a weighted sum of random variables which adheres to the definition of a GP. Alternatively, if $Z(\mathbf{s})$ is assumed to be a GP with mean zero and exponential covariance function, there are various techniques which can produce the desired result, including Cholesky decomposition, random coins, turning bands, and circulant embedding. Since C is time-dependent, we will need to generate a new covariance matrix for each day of the year, and so our implementation must be

sufficiently clever to avoid unnecessarily lengthy computations.

Perhaps the most straightforward means of GP simulation involves the famous Cholesky factorization (or decomposition) of a matrix, widely used across many disciplines of mathematics. If $\mathbf{\Sigma}$ is a positive definite matrix, then its Cholesky decomposition is defined to be

$$\mathbf{\Sigma} = \mathbf{L}\mathbf{L}^* \tag{5.7}$$

where $\mathbf{L}$ is a lower triangular matrix, and $\mathbf{L}^*$ denotes the adjoint, or conjugate transpose, of $\mathbf{L}$. If $\mathbf{A}$ is real-valued, then $\mathbf{L}$ is also real-valued, so that $\mathbf{L}^* = \mathbf{L}^\mathsf{T}$. A property of covariance matrices is that they are necessarily nonnegative definite, so the Cholesky decomposition will not work for every choice of covariance function, but in the case of an exponential covariance, this hypothesis is satisfied. Hence, the method proceeds as follows. If $Z(s)$ is a GP with covariance $C(\cdot,\cdot)$ which induces a real covariance matrix $\mathbf{\Sigma}$, to simulate Z at locations $\mathbf{s}_1, \ldots, \mathbf{s}_n \in \mathcal{S}$, take

$$\mathbf{L}\boldsymbol{\varepsilon} = \mathbf{L}\begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{pmatrix}, \quad \varepsilon_1, \ldots, \varepsilon_n \sim \mathcal{N}(0,1).$$

Thus, this demonstrates an exact simulation, since $\mathbf{L}\varepsilon$ is a multivariate normal with

$$\mathsf{E}\left[\mathbf{L}\boldsymbol{\varepsilon}\right] = \mathbf{L}\mathsf{E}\left[\boldsymbol{\varepsilon}\right] = \mathbf{L}\mathbf{0} = \mathbf{0}$$

and

$$\begin{aligned} \operatorname{Var}\left[\mathbf{L}\boldsymbol{\varepsilon}\right] &= \operatorname{Cov}\left[\mathbf{L}\boldsymbol{\varepsilon}, \mathbf{L}\boldsymbol{\varepsilon}\right] \\ &= \mathbf{L}\operatorname{Cov}\left[\boldsymbol{\varepsilon}, \boldsymbol{\varepsilon}\right]\mathbf{L}^* \\ &= \mathbf{L}\mathbf{I}\mathbf{L}^\mathsf{T} \\ &= \mathbf{\Sigma} \end{aligned}$$

For our purposes, given a vector of candidate parameters $\boldsymbol{\alpha}$, we can construct our covariance matrix $\mathbf{\Sigma}(\boldsymbol{\alpha})$ and then compute 365 Cholesky decompositions, that is, one for each day of the year. With this list of Cholesky factors, we can reuse them throughout the duration of our simulations, which will save a lot of computation since the day count will be on the order of thousands, or even tens of thousands, of days, as well as thousands of iterations to produce candidate parameters $\boldsymbol{\alpha}$.

Once the covariance matrices are accounted for, the simulation of a GP more or less follows the machinery laid out by equations (5.2), (5.3), and (5.4). Algorithm 9 makes our approach more explicit while still trying to maintain generality for the sake of extensibility or modification. For example, one might insist on using circulant embedding which outperforms Cholesky decomposition significantly at the cost of a more complicated GP simulation algorithm. Nonetheless, algorithm 9 suffices for our purposes, and we will assume its implementation when identifying our ABC procedures in the following section.

5.5 Metrics for Estimating Spatiotemporal Precipitation Occurrence

We seek to apply the ABC methodology of our previous chapters, which examined the simulation of daily local maximum temperature and precipitation occurrence, to a generalized spatiotemporal framework which incorporates multiple locations over the span of an arbitrary number of years. Unsurprisingly, our new setup has accumulated new difficulties, so we need to address our intentions and assumptions with care and precision. First and foremost, we must discuss our strategy in the estimation of our collection of parameters $\boldsymbol{\theta} = \left(\boldsymbol{\beta}^{\mathsf{T}}, \boldsymbol{\alpha}^{\mathsf{T}}, \tau^2\right)^{\mathsf{T}}$, which amounts to a formidable 11 parameters of interest.

In general, it is sensible to assume that each β_i can also vary across locations, so that $\boldsymbol{\beta} = \boldsymbol{\beta}(\mathbf{s})$. This becomes crucial when $\mathcal{S}$ has highly variable orography and general circuluation patterns [12, 33, 39, 52]. While highly accurate, this has quickly become infeasible, since there will be $6n + 4$ parameters to estimate, where n is the number of locations in question, often leading to well over 100 parameters. One possible remedy is to regress each β_i across the domain based on available geographical quantities like longitude, latitude, and elevation, as well as their interactions. However, even this will lead to an unruly amount of paramters, since for example, the model

$$\begin{aligned}\mathbf{E}\left[\beta_i\right] &= \gamma_0 + \gamma_1 \text{Longitude} + \gamma_2 \text{Latitude} + \gamma_3 \text{Elevation} + \gamma_4 \text{Longitude Latitude} \\ &\quad + \gamma_5 \text{Longitude Elevation} + \gamma_6 \text{Latitude Elevation} + \gamma_7 \text{Longitude Latitude Elevation}, \\ &\quad i = 0, \ldots, 5\end{aligned} \tag{5.8}$$

Algorithm 9: Simulating an exponential Gaussian process using Cholesky decomposition

Input : Mean function μ parametrized by $\boldsymbol{\beta}$, covariance function C parametrized by $\boldsymbol{\alpha}$
Output: Matrix $\mathbf{W} \in \mathbb{R}^{n \times T}$, where $w_{i,j}$ represents the realization of a thresholded GP at location $\mathbf{s}_i$ on day j.

choleskies $\leftarrow \{\}$ # List of cholesky factors
$A(t) = \exp\left(a_0 + a_1 \cos\left(\frac{2\pi t}{365}\right) + a_2 \sin\left(\frac{2\pi t}{365}\right)\right)$
for $i = 1, \ldots, 365$ **do**
  $\boldsymbol{\Sigma} \leftarrow \left(\exp\left\{\frac{-\|\mathbf{s}_i - \mathbf{s}_j\|}{A(t)}\right\}\right)_{i,j=1}^{n}$
  Decompose $\boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}^*$
  choleskies[i] $\leftarrow \mathbf{L}$
end
$\mathbf{W}[1] \leftarrow \mathbf{0}_{n \times 1}$ # Arbitraty set all initial values to 0 for day 1
for $d = 2, \ldots, T$ **do**
  for $\mathbf{s}_i \in \mathcal{S}$ **do**
    $\mu_i \leftarrow \mathbf{X}(\mathbf{s}_i, d)\boldsymbol{\beta}(\mathbf{s}_i)$
  end
  $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}_{n \times 1}, \mathbf{I}_{n \times n})$
  $d' \leftarrow d$ **mod** 365
  $\mathbf{Z} \leftarrow$ choleskies[d'] $\boldsymbol{\varepsilon}$
  $\boldsymbol{\Omega} \leftarrow \mathbf{Z} + \boldsymbol{\mu} + \mathcal{N}(\mathbf{0}_{n \times 1}, \tau^2 \mathbf{I}_{n \times n})$
  for $\mathbf{s}_i \in \mathcal{S}$ **do**
    $w_{i,d} \leftarrow \mathbf{1}_{[\omega_{i,d} \geq 0]}$
  end
end

already encumbers us with $6 \times 8 = 48$ mean parameters and 52 total. Of course, one can conduct a model fitting evaluation to identify the most significant covariances, but yet again we are almost certainly left with more parameters than can be handled by ABC.

Instead, we restrict our focus on the special case of a spatially constant mean function, a simplification that describes a rather limited subset of spatial domains. For example, examinations of empirical probabilities of precipitation occurrence over the course of a year for various locations in Colorado dissuades us from even considering this simplification over its domain. Nonetheless, some geographical areas do adhere to this assumption, such as the state of Iowa, using the same examination of empirical precipitation probabilities across the state. We will resort to one more mitigator, which is to first estimate $\boldsymbol{\beta}$ using some form of ABC, and then use these estimated $\boldsymbol{\beta}$ parameters to estimate $\boldsymbol{\alpha}$ and τ using a different embodiment of ABC. While perhaps not ideal, it can be justified by the modularity of $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ in our model. Furthermore, we will illustrate that this approach can yield appealing results and should be viewed as a solid foundation to the application of ABC to the spatiotemporal simulation of precipitation occurrence.

To estimate $\boldsymbol{\beta}$, we will need to come up with a statistic that measures the accuracy of the mean function over the entire domain. Under the assumption that the terrain does not vary greatly and that the precipitation behaves similarly at each location $\mathbf{s}_i \in \mathcal{S}$, then we can consider a sort of aggregation of our metric (4.8) from the previous chapter. That is we can compute a standardized sum of the wet and dry spell errors over all of our locations in $\mathcal{S}$, so that

$$\widehat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \left(\frac{1}{|\mathcal{Z}_1|} \sum_{\zeta_1 \in \mathcal{Z}_1} \frac{|\varphi(\mathbf{s}_i[t]; \zeta_1) - \varphi(\mathbf{o}_i[t]; \zeta_1)|}{\max\{\varphi(\mathbf{o}_i[t]; \zeta_1), 1\}} + \frac{1}{|\mathcal{Z}_2|} \sum_{\zeta_2 \in \mathcal{Z}_2} \frac{|\psi(\mathbf{s}_i[t]; \zeta_2) - \psi(\mathbf{o}_i[t]; \zeta_2)|}{\max\{\psi(\mathbf{o}_i[t]; \zeta_2), 1\}} \right) \right\}. \tag{5.9}$$

where, as before, φ counts wet spells as defined in (4.5), ψ counts dry spells as defined in (4.6), $\mathcal{T}$ is a partition of the months of the year, $\mathcal{Z}_1$ is a set of sets of wet spell lengths, $\mathcal{Z}_2$ is a set of sets of dry spell lengths, and the notation $\mathbf{x}[t]$ represents the time series $\mathbf{x}$ subsetted by the set of time points t. We need to tread lightly with this metric since such an aggregation will lose much of the single-site behavior, but in principle it should favor values of $\boldsymbol{\beta}$ which represent $\mathcal{S}$ well as a whole

while distrusting values of $\boldsymbol{\beta}$ that skew the count at any given site.

After obtaining the estimate $\widehat{\boldsymbol{\beta}}$ of our mean function, we can use it to estimate the remaining parameters, $\boldsymbol{\gamma} = \left(\boldsymbol{\alpha}^{\mathsf{T}}, \tau^2\right)^{\mathsf{T}}$. These parameters are intrinsic to the spatial covariance of the domain $\mathcal{S}$, and consequently, they must be estimated via assessing this covariance. Here we can use our aggregate monthly variogram $\widehat{\Gamma}$ as defined in (5.6), since it computes a smoothed monthly error of the binned variograms for the observed and simulated data. Therefore, in the spirit of our metrics (3.3) and (4.8), we define our ABC estimator $\widehat{\boldsymbol{\gamma}}_{\text{ABC}}$ by

$$\widehat{\boldsymbol{\gamma}}_{\text{ABC}} = \arg\min_{\boldsymbol{\gamma}} \frac{1}{|\mathcal{M}|} \sum_{m \in \mathcal{M}} \frac{\left|\widehat{\Gamma}(m|\mathbf{o}) - \widehat{\Gamma}(m|\mathbf{s}_{\boldsymbol{\alpha}})\right|}{\widehat{\Gamma}(m|\mathbf{o})}, \tag{5.10}$$

where $\mathcal{M}$ is the desired set of months to use in the penalization. A straightforward choice is to let $\mathcal{M} = \{1, \ldots, 12\}$, simply the set of months in the year, which will be our choice. The combination of these metrics will constitute our ABC algorithm for spatial precipitation occurrence, which is further elucidated in algorithm 10.

5.6 Numerical Results for the State of Iowa

For the spatiotemporal case of precipitation simulation, we depart from the state of Colorado in favor of the state of Iowa due to its homogeneous terrain. Once again, we will use data provided by the GHCND which contains 42,705 days of data from 22 different spatial locations across Iowa. For the ABC-estimation of the mean function parameters $\boldsymbol{\beta}$, we choose a 20 year sequence of days for tractability, and reuse our wet and dry spell sets $\mathcal{Z}_1$ and $\mathcal{Z}_2$ from the local precipitation case in Chapter 4. For our tolerance ε_1, we first consider the error used for a single location, which was roughly 1.09, increase it slightly due to the variability throughout the domain, and then multiply it by the number of locations $n = 22$, so that $\varepsilon_1 = 1.15 \times 22 = 25.3$, Moreover, to assist our algorithm as much as possible, we conduct a grid search through the prior space to get a minimizing value of our ϱ function, which we assign to be our initial guess $\boldsymbol{\beta}_0$. Using uniform priors leads to sequences of accepted values that tend to stray, and although they are expected

Algorithm 10: ABC-MCMC for daily spatiotemporal precipitation occurrence parameter estimation

Input : Observed data $\mathcal{D}$, initial values $\boldsymbol{\beta}_0, \boldsymbol{\gamma}_0$, candidate-generating densities $q_1(\cdot,\cdot), q_2(\cdot,\cdot)$, tolerances $\varepsilon_1, \varepsilon_2 > 0$, desired number of samples N_1 for $\boldsymbol{\beta}$ and N_2 for $\boldsymbol{\gamma}$

Output: A vector V of samples approximately distributed from target distribution $f(\boldsymbol{\theta}|\mathcal{D})$

Define $\mathcal{M}_1$ using (5.3)

Define $\mathcal{M}_2$ using (5.2) Define $\varrho_{\boldsymbol{\beta}}$ as follows:

$$\varrho_{\boldsymbol{\beta}}\left(\mathcal{D},\mathcal{D}'\right) := \frac{1}{|\mathcal{S}|}\sum_{i=1}^{m}\frac{1}{|\mathcal{T}|}\sum_{t\in\mathcal{T}}\left(\frac{1}{|\mathcal{Z}_1|}\sum_{\zeta_1\in\mathcal{Z}_1}\frac{|\varphi(\mathbf{s}_i[t];\zeta_1)-\varphi(\mathbf{o}_i[t];\zeta_1)|}{\max\{\varphi(\mathbf{o}_i[t];\zeta_1),1\}} + \frac{1}{|\mathcal{Z}_2|}\sum_{\zeta_2\in\mathcal{Z}_2}\frac{|\psi(\mathbf{s}_i[t];\zeta_2)-\psi(\mathbf{o}_i[t];\zeta_2)|}{\max\{\psi(\mathbf{o}_i[t];\zeta_2),1\}}\right)$$

Define $\varrho_{\boldsymbol{\gamma}}$ as follows:

$$\varrho_{\boldsymbol{\gamma}}\left(\mathcal{D},\mathcal{D}'\right) := \frac{1}{|\mathcal{M}|}\sum_{m\in\mathcal{M}}\frac{\left|\widehat{\Gamma}(m,\mathcal{D})-\widehat{\Gamma}(m,\mathcal{D}')\right|}{\left|\widehat{\Gamma}(m,\mathcal{D})\right|}$$

$V_{\boldsymbol{\beta}} \leftarrow \{\}$
$\boldsymbol{\beta} \leftarrow \boldsymbol{\beta}_0$
while $|V_{\boldsymbol{\beta}}| < N_1$ **do**
  generate $\boldsymbol{\beta}_{\text{candidate}} \sim q_1(\boldsymbol{\beta},\cdot)$
  simulate $\mathcal{D}'(\boldsymbol{\beta}_{\text{candidate}})$ from $\mathcal{M}_1$
  if $\varrho_{\boldsymbol{\beta}}\left(\mathcal{D},\mathcal{D}'(\boldsymbol{\beta}_{\textbf{candidate}})\right) < \varepsilon_1$ **then**
    $u \sim \mathcal{U}(0,1)$
    if $u < \min\left\{\frac{\pi(\boldsymbol{\beta}_{\textbf{candidate}})q_1(\boldsymbol{\beta}_{\textbf{candidate}},\boldsymbol{\beta})}{\pi(\boldsymbol{\beta})q_1(\boldsymbol{\beta},\boldsymbol{\beta}_{\textbf{candidate}})},1\right\}$ **then**
      $\boldsymbol{\beta} \leftarrow \boldsymbol{\beta}_{\text{candidate}}$
    end
  end
  append $\boldsymbol{\beta}$ to $V_{\boldsymbol{\beta}}$
end
$V_{\boldsymbol{\gamma}} \leftarrow \{\}$
$\boldsymbol{\gamma} \leftarrow \boldsymbol{\gamma}_0$
while $|V_{\boldsymbol{\gamma}}| < N_2$ **do**
  generate $\boldsymbol{\beta} \sim V_{\boldsymbol{\beta}}$
  generate $\boldsymbol{\gamma}_{\text{candidate}} \sim q_2(\boldsymbol{\gamma},\cdot)$
  simulate $\mathcal{D}'(\boldsymbol{\beta},\boldsymbol{\gamma}_{\text{candidate}})$ from $\mathcal{M}_2$
  if $\varrho_{\boldsymbol{\gamma}}(\mathcal{D},\mathcal{D}'(\boldsymbol{\beta},\boldsymbol{\gamma}_{\textbf{candidate}})) < \varepsilon_2$ **then**
    $u \sim \mathcal{U}(0,1)$
    if $u < \min\left\{\frac{\pi(\boldsymbol{\gamma}_{\textbf{candidate}})q_2(\boldsymbol{\gamma}_{\textbf{candidate}},\boldsymbol{\gamma})}{\pi(\boldsymbol{\gamma})q_2(\boldsymbol{\gamma},\boldsymbol{\gamma}_{\textbf{candidate}})},1\right\}$ **then**
      $\boldsymbol{\gamma} \leftarrow \boldsymbol{\gamma}_{\text{candidate}}$
    end
  end
  append $\boldsymbol{\gamma}$ to $V_{\boldsymbol{\gamma}}$
end

to approximate the posterior asymptotically, do not return in a reasonable about of time for this situation; this is unsurprising behavior when computing such a high-level aggregation of data values in our metric (5.9). So, to achieve tractability, we instead elect normally distributed priors, centered around our initial guess. More explicitly, our ith prior π_i is distributed as a $\mathcal{N}(\boldsymbol{\beta}_{0,i}, 0.1)$ random variable.

Note that our ABC-MCMC algorithm as laid out in Chapter 2 allows for any valid probability distribution, but the inclusion of our grid search value as the mean of our priors may lead to questions of circular analysis. A couple of justifying remarks can be made in this regard. First, the idea of a prior distribution as a representation of belief creates an inherent partiality in the methodology, and so any error introduced from this double dipping can be incorporated into the already present influence. In addition, SWGs aim merely to replicate the statistical characteristics of a dataset, and do not attempt to make claims about the truth of the underlying parameters. That is, the resulting posteriors are not professing statistical strength, but instead attempt to provide satisfactory results for the application of the simulated data, which is much more subjective. Nonetheless, there are a couple of remedies which can be implemented if desired or necessary. The concerned modeler can instead center their priors based on less biased values, such as centering around -1 for the intercept β_0, about 0.5 for the auto-regressive covariate, and about 0 for the sinusoidal regressors, and adjust the variance as necessary to acquire a sufficient amount of samples. Another remedy is to return to the uniform priors and implement the variant provided by Sisson et. al. that alleviates the bias in the samples [69]. Nonetheless, skeptical ourselves, we shifted our priors around to see how they affected the posteriors, and it turns out that the posteriors are quite robust and did not change much at all.

Figure 5.1 shows the ABC posteriors after about 1,600 samples and discarding the burn-in. We see that there is a considerable amount of certainty in the posteriors, and this is confirmed by examining a plot of the expected probability of precipitation against the 22 empirical probabilities, which is shown in figure 5.2. Since Iowa was chosen due to its mostly unchanging terrain, the behavior of the domain as a whole is able to be replicated quite well. These results allow us to

proceed with our procedure 10 and estimate γ. Again, we use a 20-year subset of the data, and use a grid search to get our initial guess γ_0. We choose $\varepsilon_2 = 1.0$, and uniform priors. After the burn-in, we obtain 800 samples, whose densities are displayed in figure 5.3. We immediately observe the increase in uncertainty for these parameters. Examining the aggregate variogram $\widehat{\Gamma}(m)$ for each month shown in figure 5.4 gives some illumination: while $\widehat{\Gamma}$ is estimated closely for many months, we see consistent over- and underestimation for several months, including March, September, and December. One possible source of this error is the exclusion of the higher-order sinusoidal covariates $\cos(4\pi d/365)$ and $\sin(4\pi d/365)$ (whose exclusion results from their hindrance of ABC with very few parameter jumps in the acceptances). Perhaps 20 years is insufficient to capture the covariance, which is plausible given the limitation of binary-valued data. Regardless, the covariances seem to be replicated for the most part, and if nothing else demonstrates the potential of algorithm 10 with proper modifications.

As a final glimpse of the tenability of the SPG given by algorithm 10, we can create a simulation at our 22 locations in Iowa using the empirical means of each posterior for 12 days ranging from mid- to late-January, and plot them in sequence. These plots are shown in figure 5.5. There are several features which are compatible with the observed data and our general assumptions. It appears that no precipitation is more common across the domain, which conforms to our observations of precipitation behavior. There is also evidence of precipitation on a day at a site often leading to precipitation on the following day at the same site. Moreover, the spatial dependence is palpable with clusters of precipitation occurrence when it exists, which is compatible with the real-world clustered behavior of precipitation.

5.7 Discussion

Using a latent Gaussian process allows for correlated binary-valued simulations of precipitation occurrence, and ABC allows us to adequately estimate the inherent statistical parameters. Our case study of Iowa leads to many avenues of exploration in the application of ABC to spatiotemporal SPGs. Future work includes transgressing the assumption of a constant mean function, so

Figure 5.1: The prior and ABC posterior densities for each parameter β_i via (5.3).

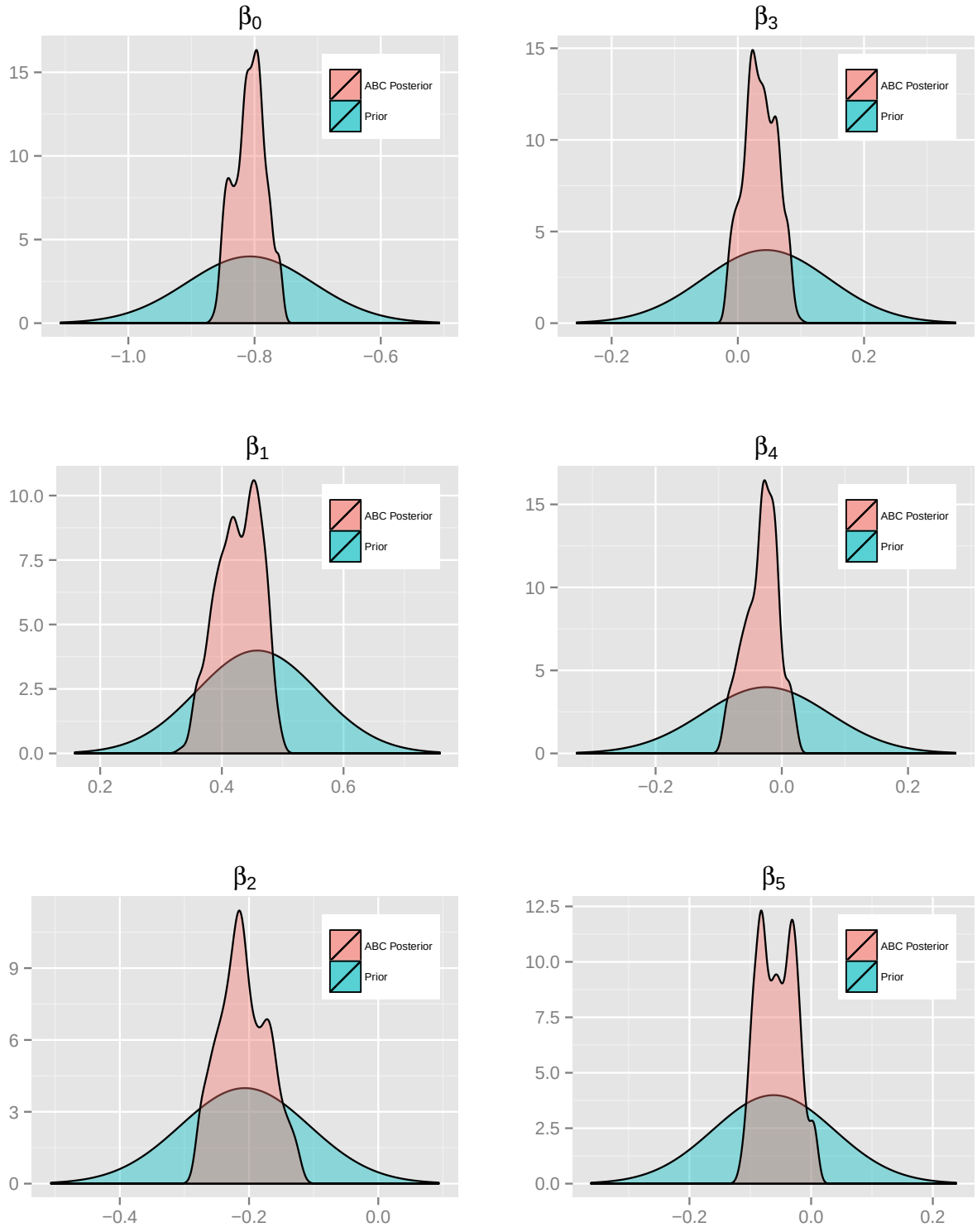


Figure 5.2: The empirical probability of precipitation by day averaged over each month and over all of the 22 locations in Iowa (represented by the box plots), as well as the mean daily probability of precipitation given by our $\beta_{\text{ABC-MCMC}}$ estimate (represented by the solid blue line).

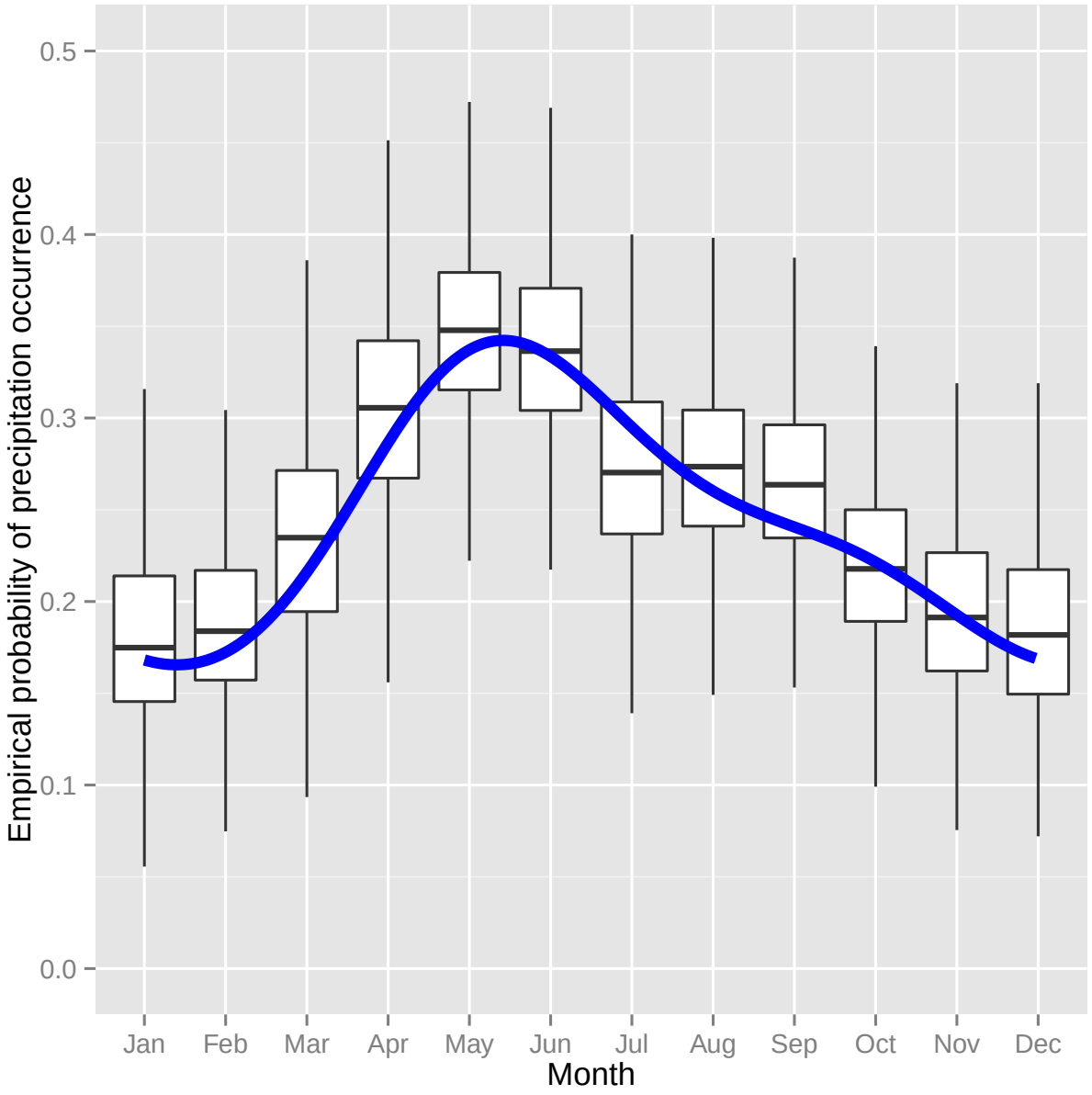


Figure 5.3: The prior and ABC posterior densities for parameters α_i from (5.4), as well as the nugget effect τ^2. Note that the density represents the square root of the nugget effect, $\tau = \sqrt{\tau^2}$, an artifact of our method of implementation.

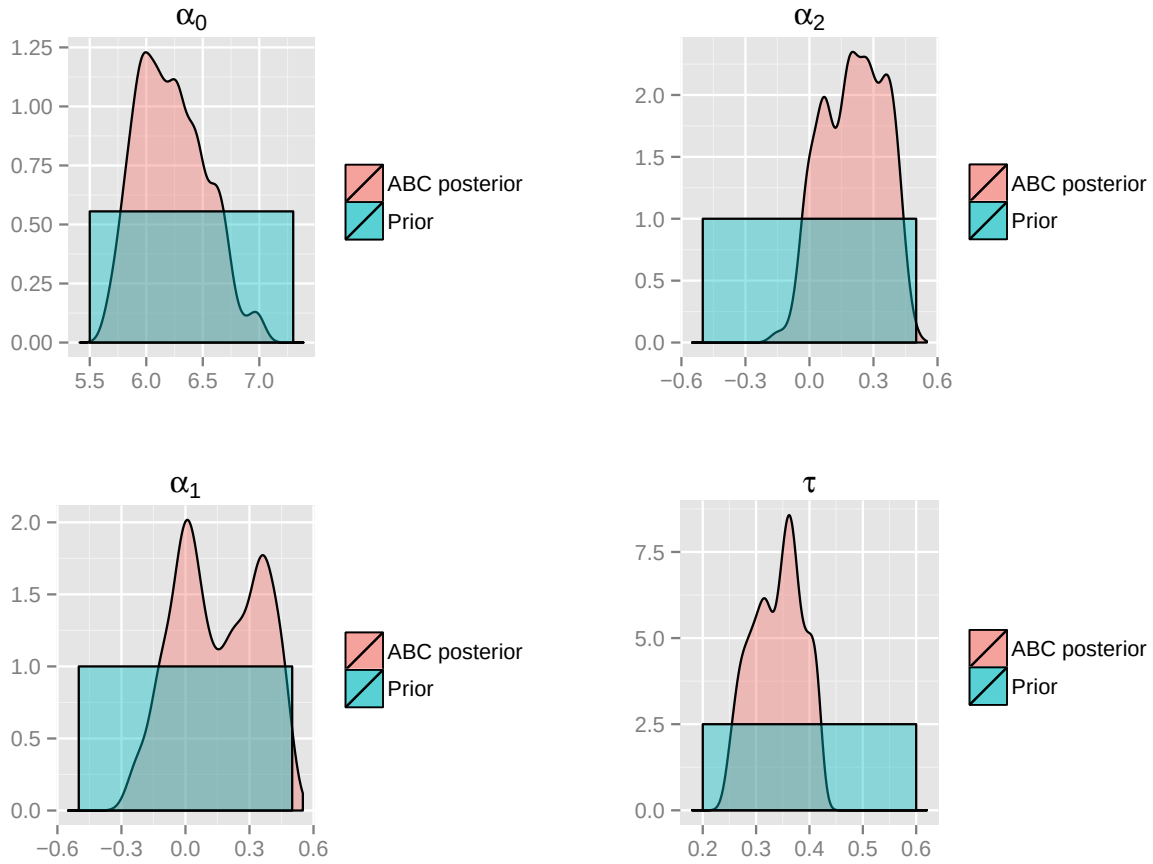


Figure 5.4: The aggregate variogram $\widehat{\Gamma}(m)$ of our simulated thresholded Gaussian process for each month for the state of Iowa (represented by the box plots), as well as the observed aggregate variogram for the state of Iowa (represented by the solid blue lines).

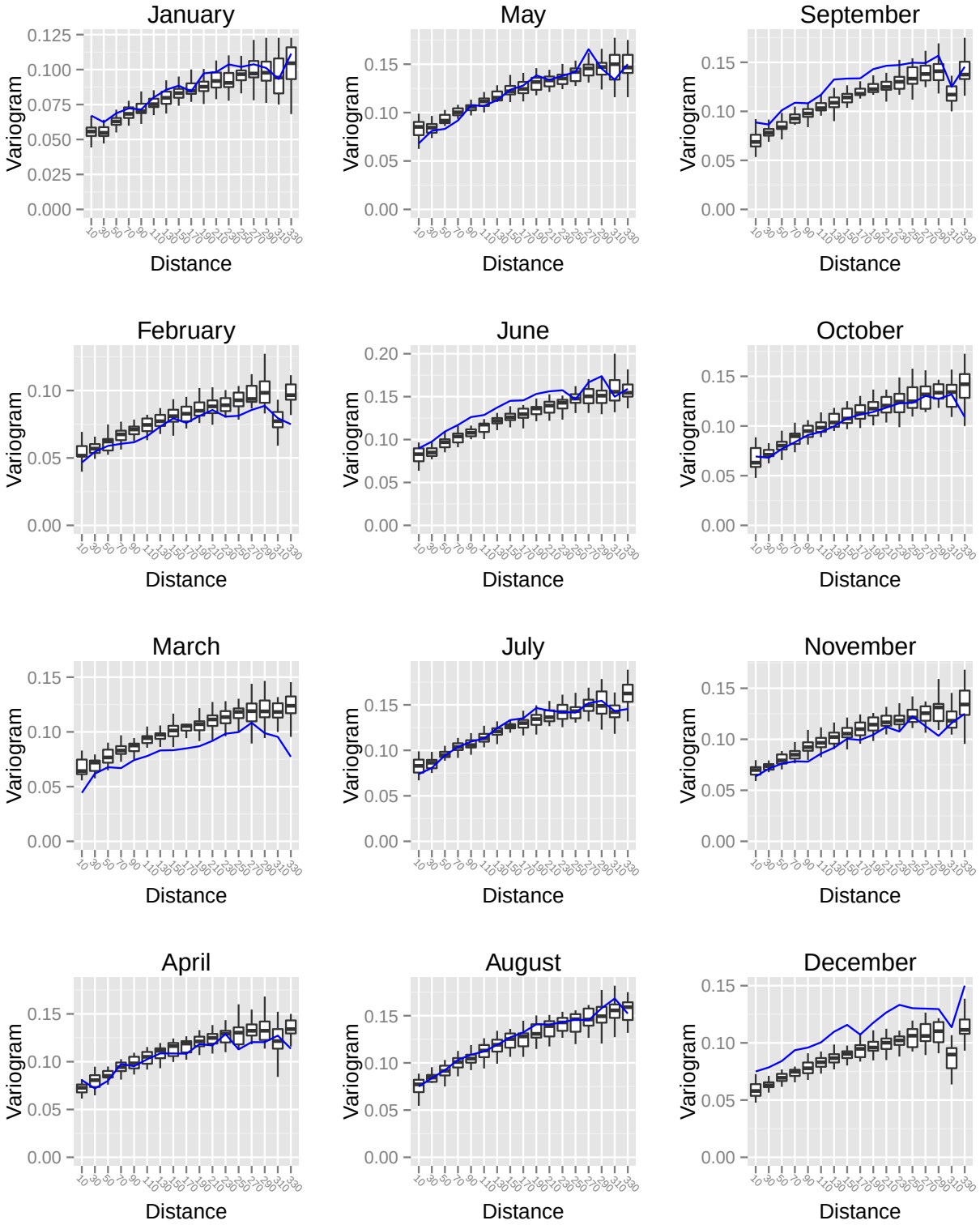
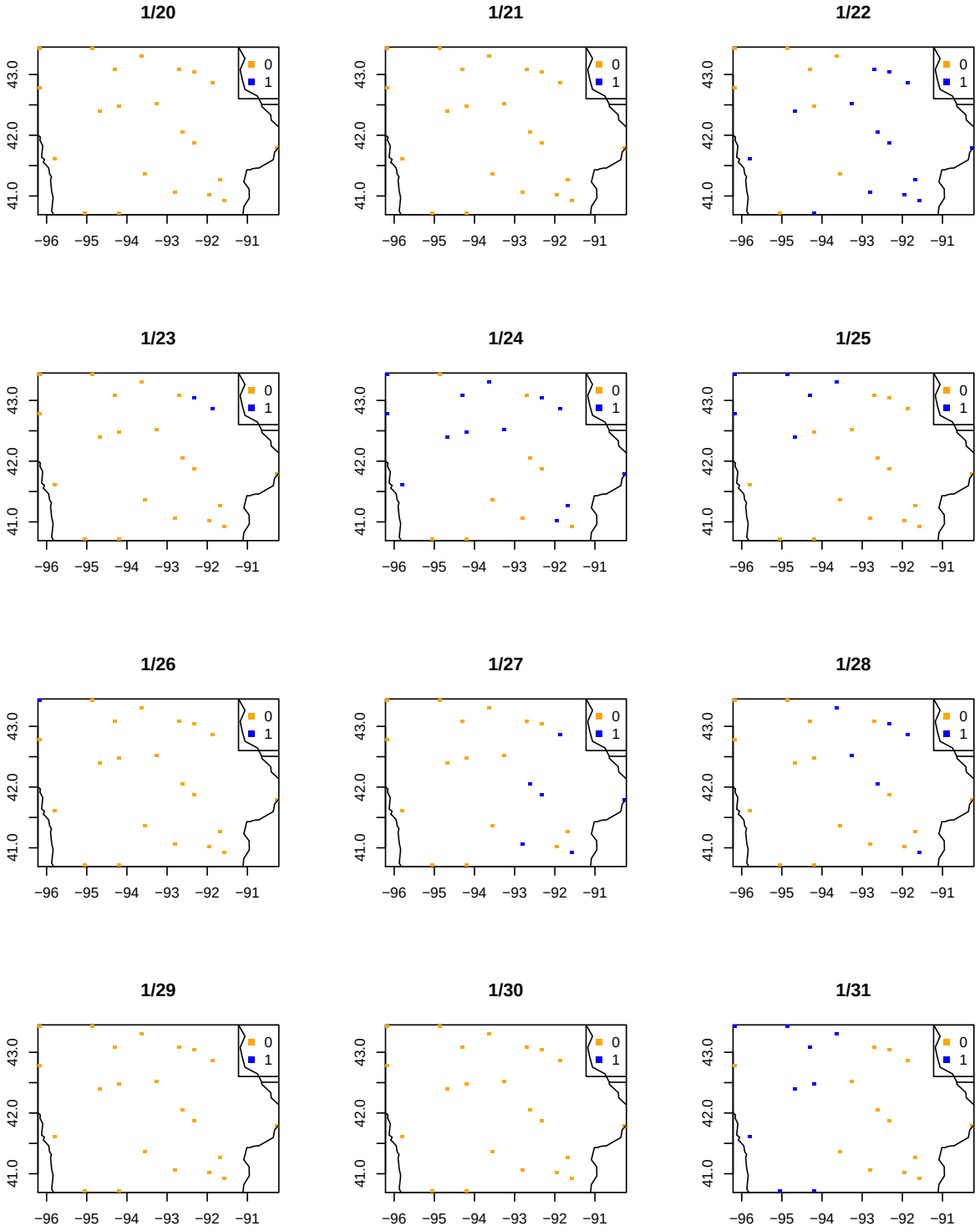


Figure 5.5: Sample daily precipitation occurrence simulation over Iowa from January 20 through January 31.



that $\boldsymbol{\beta} = \boldsymbol{\beta}(\mathbf{s})$. This will present a challenge in how to estimate the mean parameters, although there may be a feasible solution in the regression approach given by 5.8. Modelers can experiment with different covariance functions, such as the celebrated Matern covariance function, and assess their utility and implications. As in the local precipitation occurrence case, a natural expansion to our model would be the integration of precipitation intensity, which is a very desirable feature of an SPG that carries its own set of parameters to be estimated. Algorithm 9 can be optimized by more clever methods of Gaussian process simulation, which can lead to more acceptances, and in turn may allow for an increase in model parameters.

The study of spatiotemporal precipitation occurrence simulation concludes our examination of stochastic weather generation using ABC. The theme of this thesis has been the perpetual battle of precision and pragmatism, a theme familiar in the practice of statistics and applied mathematics. Ultimately, ABC methods have proven to be well suited to this kind of trade-off through their flexibile yet sturdy nature. We persisted with the classical ABC-MCMC given by algorithm 6, but it should be reiterated that much of the bias and inaccuracy we have witnessed can be reduced using modified versions of the algorithm. In addition, there is a complementary static-dynamic balance rooted in the union of ABC and SWGs. For example, one detail that has been glossed over is the fact that with posterior distributions for our parameters, not only can we generate statistically consistent simulations, but we can change the way we extract our samples from these posteriors to alter our data values. This can lead to synthetic weather that is similar to, but significantly different from, observed weather patterns, and this can be beneficial in many models that seek to illustrate the effect of these parameter changes. We hope these musings convince the reader that the application of ABC to SWGs is something we should continue to explore with optimism.

Bibliography

[1] P. Ailliot, C. Thompson, and P. Thomson. Space-time modelling of precipitation by using a hidden markov model and censored gaussian distributions. Journal of the Royal Statistical Society: Series C (Applied Statistics), 58(3), 2009.

[2] D. J. Allcroft and C. A. Glasbey. A latent gaussian markov random-field model for spatiotemporal rainfall disaggregation. Journal of the Royal Statistical Society: Series C (Applied Statistics), 52, 2003.

[3] S. Apipattanavis, G. Podestá, B. Rajagopalan, and R. W. Katz. A semiparametric multivariate and multisite weather generator. Water Resources Research, 43, 2007.

[4] Mark A. Beaumont. Adaptive approximate bayesian computation. Biometrika, 2009.

[5] Mark A. Beaumont, Wenyang Zhang, and David J. Balding. Approximate bayesian computation in population genetics. Genetics, 2002.

[6] F. P. Brissette, M. Khalili, and R. Leconte. Efficient stochastic generation of multi-site synthetic precipitation data. Journal of Hydrology, 345, 2007.

[7] P. E. Brown, P. J. Diggle, M. E. Lord, and Young P. C. Space-time calibration of radar rainfall data. Journal of the Royal Statistical Society: Series C, 50, 2001.

[8] T. A. Buishand and T. Brandsma. Multisite simulation of daily precipitation and temperature in the rhine basin by nearest-neighbor resampling. Water Resources Research, 37(11), 2001.

[9] A. Bárdossy and G. G. S. Pegram. Copula based multisite model for daily precipitation simulation. Hydrology and Earth System Sciences, 13, 2009.

[10] Daly C., R. P. Neilson, and D. L. Phillips. A statistical-topographic model for mapping climatological precipitation over mountainous terrain. Journal of Applied Meteorology, 33, 1994.

[11] A. J. Cannon. Probabilistic multisite precipitation downscaling by an expanded bernoulli-gama density network. Journal of Hydrometeorology, 9, 2008.

[12] R. E. Chandler. On the use of generalized lineaer models for interpreting climate variability. Environmetrics, 16, 2005.

[13] Richard E. Chandler and Howard S. Wheater. Analysis of rainfall variability using generalized linear models: A case study from the west of ireland. Water Resources Research, 38(10), 2002.

[14] S. P. Charles, B. C. Bates, and J. P. Hughes. A spatiotemporal model for downscaling precipitation occurrence and amounts. Journal of Geophysical Research, 104, 1999.

[15] Siddhartha Chib and Edward Greenberg. Understanding the metropolis-hastings algorithm. The American Statistican, 49(4), 1995.

[16] Noel Cressie. Statistics for Spatial Data. Wiley Interscience, 1993.

[17] Noel Cressie and Douglas M. Hawkins. Robust estimation of the variogram: I. Mathematical Geology, 12(2), 1980.

[18] P. J. Diggle and J. A. Tawn. Model-based geostatistics. Applied Statistics, 47(3), 1998.

[19] Annette Dobson and Adrian Barnett. An Introduction to Generalized Linear Models. Taylor & Francis Group, 2008.

[20] M. Durban and C. A. Glasbey. Weather modelling using a multivariate latent gaussian model. Agricultural and Forest Meterology, 109, 2001.

[21] A.D. Friend, A.K. Stevens, R.G. Knox, and M.G.R. Cannell. A process-based, terrestrial biosphere model of ecosystem dynamics. Ecological Modeling, 1997.

[22] Y.X. Fu and W.H. Li. Estimating the age of the common ancestor of a sample of dna sequences. Molecular Biology and Evolution, 14, 1997.

[23] M. Fuentes. Spectral methods for nonstationary spatial processes. Biometrika, 89, 2002.

[24] K.R. Gabriel and J. Neumann. A markov chain model for daily rainfall occurrence at tel aviv. Quarterly Journal of the Royal Meteorological Society, 88(375), 1962.

[25] A. E. Gelfand, S. Banerjee, and D. Gamerman. Spatial process modelling for univariate and multivariate dynamic spatial data. Environmetrics, 16, 2005.

[26] A. E. Gelfand, A. M. Schmidt, S. Banerjee, and C. F. Sirmans. Nonstationary multivariate process modeling through spatially varying coregionalization. Sociedad de Estadística e Investigación Operativa Test, 13(2), 2004.

[27] T.C. Haas. Lognormal and moving window methods of estimating acid deposition. Journal of the American Statistical Association, 85, 1990.

[28] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. Biometrika, 57(1), 1970.

[29] D. Higdon. A process-convolution approach to modelling temperatures in the north atlantic ocean. Environmental and Ecological Statistics, 5, 1998.

[30] R. J. Hijmans, S. E. Cameron, J. L Parra, P. G. Jones, and A. Jarvis. Very high resolution interpolated climate surfaces for global land areas. International Journal of Climatology, 25, 2005.

[31] J. P. Hughes and P. Guttorp. A non-homogeneous hidden markov model for precipitation occurrence. Applied Statistics, 48, 1999.

[32] M. F. Hutchinson. Interpolating mean rainfall using thin plate smoothing splines. International Journal of Geographical Information Systems, 9, 1995.

[33] G. L. Johnson, C. Daly, G. H. Taylor, and C. L. Hanson. Spatial variability and interpolation of stochastic weather simulation model parameters. Journal of Applied Meteorology, 39, 2000.

[34] M. Jun. Non-stationary cross-covariance models for multivariate processes on a globe. Scandinavian Journal of Statistics, 38, 2011.

[35] R. W. Kats. Precipitation as a chain-dependent process. Journal of Applied Meterology, 16, 1977.

[36] C. G. Kilsby, P. D. Jones, A. Burton, A. C. Ford, H. J. Fowler, C. Harpham, P. James, A. Smith, and R. L. Wilby. A daily weather generator for use in climate change studies. Environmental Modelling and Software, 22, 2007.

[37] H.-M. Kim, B. K. Mallick, and C. C. Holmes. Analyzing nonstationary spatial data using piecewise gaussian processes. Journal of the American Statistical Association, 100, 2005.

[38] W. Kleiber and D. Nychka. Nonstationary modeling for multivariate spatial processes. Journal of Multivariate Analysis, 112, 2012.

[39] William Kleiber, Richard W. Katz, and Balaji Rajagopalan. Daily spatiotemporal precipitation simulation using latent and transformed gaussian processes. Water Resources Research, 48, 2012.

[40] William Kleiber, Richard W. Katz, and Balaji Rajagopalan. Daily minimum and maximum temperature simulation over complex terrain. The Annals of Applied Statistics, 7(1), 2013.

[41] W.P. Kustas, A. Rango, and R. Uijlenhoet. A simple energy budget algorithm for the snowmelt runoff model. Water Resources Research, 30, 1994.

[42] U. Lall and A. Sharma. A nearest neighbor bootstrap for resampling hydrological time series. Water Resources Research, 32(3), 1996.

[43] D. R. Legates and C. J. Willmott. Mean seasonal and spatial variability in global surface air temperature. Theoretical and Applied Climatology, 41, 1990.

[44] P. Marjoram, J. Molitor, V. Plagnol, and S. Tavaré. Markov chain monte carlo without likelihoods. Proceedings of the National Academy of Sciences of the United States of America, 100(26), 2003.

[45] D. et. al. Mauran. Precipitation downscaling under climate change: Recent developments to bridge the gap between dynamical models and the end user. Reviews of Geophysics, 48, 2010.

[46] Linda O. Mearns, Cynthia Rosenzweig, and Richard Goldberg. Mean and variance change in climate scenarios: Methods, agricultural applications, and measures of uncertainty. Climatic Change, 35, 1997.

[47] R. Mehrotra and A. Sharma. Development and application of a multisite rainfall stochastic downscaling framework for climate change impact assessment. Water Resources Research, 46, 2010.

[48] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, and Augusta H. Teller. Equation of state calculations by fast computing machines. The Journal of Chemical Physics, 21(6), 1953.

[49] Peter Müller. A generic approach to posterior integration and gibbs sampling. 1993.

[50] David J. Nott, Lucy Marshall, and Jason Brown. Generalized likelihood uncertainty estimation (glue) and approximate bayesian computation: What's the connection? Water Resources Research, 48, 2012.

[51] C. J. Paciorek and M. J. Schervish. Spatial modelling using a new class of nonstationary covariance functions. Environmetrics, 17, 2006.

[52] N. Pepin and M. Losleben. Climate change in the colorado rocky mountains: Free air versus surface temperature trends. International Journal of Climatology, 22, 2002.

[53] T. C. Peterson and R. S. Vose. An overview of the global historical climatology network temperature database. Bulletin of the American Meterological Society, 78, 1997.

[54] A. Pintore and C. Holmes. Spatially adaptive non-stationary covariance functions via spatially adaptive spectra. Unpublished manuscript, 2006.

[55] D. T. Price, D. W. McKenney, I. A. Nalder, M. F. Hutchinson, and J. L. Kesteven. A comparison of two statistical methods for spatial interpolation of canadian monthly mean climate data. Agricultural and Forest Meteorology, 101, 2000.

[56] J.K. Pritchard, M.T. Seielstad, A. Perez-Lezaun, and M.W. Feldman. Population growth of human y chromosomes: a study of y chromosome microsatellites. Molecular Biology and Evolution, 16, 1999.

[57] P. Racsko, L. Szeidl, and M. Semenov. A serial approach to local stochastic weather models. Ecological Modeling, 57, 1991.

[58] Balaji Rajagopalan and Upmanu Lall. A *k*-nearest-neighbor simulator for daily precipitation and other weather variables. Water Resources Research, 35(10), 1999.

[59] C.W. Richardson. Stochastic simulation of daily precipitation, temperature, and solar radiation. Water Resources Research, 17(1), 1981.

[60] B.D. Ripley. Stochastic Simulation. Wiley, New York, 1982.

[61] Donald Rubin. Bayesianly justifiable and relevant frequency calculations for the applied statistician. The Annals of Statistics, 12(4), 1984.

[62] S. W. Running, R. R. Nemani, and R. D. Hungerford. Extrapolation of synoptic meteorological data in mountainous terrain and its use for simulating forest evapotranspiration and photosynthesis. Canadian Journal of Forest Research, 17, 1987.

[63] M. Sadegh and J. A. Vrugt. Bridging the gap between glue and formal statistical approaches: Approximate bayesian computation. Hydrology and Earth System Sciences, 17, 2013.

[64] Mojtaba Sadegh and Jasper A. Vrugt. Approximate bayesian computation using markov chain monte carlo simulation: Dream-abc. Water Resources Research, 50, 2014.

[65] P. D. Sampson and P. Guttorp. Nonparametric estimation of nonstationary spatial covariance structure. Journal of the American Statistical Association, 87, 1992.

[66] B. Sansó and L. Guenni. A nonstationary multisite model for rainfall. Journal of the American Statistical Association, 95, 2000.

[67] Mikhail A. Semenov and Elaine M. Barrow. Use of a stochastic weather generator in the development of climate change scenarios. Climatic Change, 35, 1997.

[68] G. Shaddick and J. Wakefield. Modelling daily multivariate pollutant data at multiple sites. Journal of the Royal Statistical Society: Series C (Applied Statistics), 51(3), 2002.

[69] S.A. Sisson, Y. Fan, and Mark M. Tanaka. Sequential monte carlo without likelihoods. Proceedings of the National Academy of Sciences of the United States of America, 104(6), 2007.

[70] R. D. Stern and R. Coe. A model fitting analysis of daily rainfall data. Journal of the Royal Statistical Society: Series A, 147, 1984.

[71] H. Strandman, H. Vaisanen, and S. Kellomaki. A procedure for generating synthetic weather records in conjunction of climate scenario for modelling of ecological impacts of changing climate in boreal conditions. Ecological Modeling, 70, 1993.

[72] J. R. Stroud, P. Müller, and B. Sansó. Dynamic models for spatiotemporal data. Journal of the Royal Statistical Society. Series B (Statistical Methodology), 63(4), 2001.

[73] M. Sunnaker, A. G. Busetto, E. Numminen, J. Corander, M. Foll, and C. Dessimoz. Approximate bayesian computation. PLoS Computational Biology, 9(1), 2013.

[74] S. Tavaré, D. J. Balding, R. C. Griffiths, and P Donnelly. Inferring coalescence times from dna sequence data. Genetics Society of America, 145, 1997.

[75] C. S. Thompson, P. J. Thomson, and X. Zheng. Fitting a multisite daily rainfall model to new zealand data. Journal of Hydrology, 340, 2007.

[76] P. E. Thornton, S. W. Running, and M. A. White. Generating surfaces of daily meteorological variables over large regions of complex terrain. Journal of Hydrology, 190, 1997.

[77] G. Weiss and A. von Haeseler. Inference of population history using a likelihood approach. Genetics, 149, 1998.

[78] D. S. Wilks. Multisite generalization of a daily stochastic precipitation generation model. Journal of Hydrology, 210, 1998.

[79] D. S. Wilks. Use of stochastic weather generators for precipitation downscaling. Wiley Interdisciplinary Review, 1, 2010.

[80] Daniel S. Wilks. Estimating the consequences of CO_2-induced climatic change on north american grain agriculture using general circulation model information. Climatic Change, 13(1), 1988.

[81] D.S. Wilks. Simultaneous stochastic simulation of daily precipitation, temperature and solar radiation at multiple sites in complex terrain. Agricultural and Forest Meteorology, 96, 1999.

[82] D.S. Wilks and R.L. Wilby. The weather generation game: a review of stochastic weather models. Progress in Physical Geography, 23(3), 1999.

[83] C. J. Willmott and K. Matsuura. Smart interpolation of annually averaged air temperature in the united states. Journal of Applied Meteorology, 34, 1995.

[84] D. A. Woolhiser and G. G. S. Pegram. Maximum likelihood estimation of fourier coefficients to describe seasonal variations of paramters in stochastic daily precipitation models. Journal of Applied Meteorology, 18, 1979.

[85] C. Yang, R. E. Chandler, V. S. Isham, and H. S. Wheater. Spatial-temporal rainfall simulation using generalized linear models. Water Resources Research, 41, 2005.

[86] X. Zheng and R. W. Katz. Simulation of spatial dependence in daily rainfall using multisite generators. Water Resources Research, 44, 2008.

[87] X. Zheng, J. Renwick, and A. Clark. Simulation of multisite precipitation using an extended chain-dependent process. Water Resources Research, 46, 2010.

Appendix : utils.R

```
# Load libraries
library(fields)
library(ggplot2)
library(MASS)
library(glmulti)
library(grid)

# Function definitions
loadParams <- function(filename) {
  res <- as.matrix(read.csv(filename, header=FALSE))
  res <- t(res)
  return(res)
}

saveParams <- function(params, filename) {
  write(params, filename, ncolumns=dim(params)[1], sep=',')
}

saveRDA <- function(data.object, filename) {
  saveRDS(data.object, file=filename)
}

loadRDA <- function(filename) {
  return( readRDS(filename) )
}

multiplot <- function(plots, plotlist=NULL, file, cols=1, layout=
   NULL) {
  library(grid)

  # Make a list from the ... arguments and plotlist
  numPlots = length(plots)

  # If layout is NULL, then use 'cols' to determine layout
  if (is.null(layout)) {
    # Make the panel
    # ncol: Number of columns of plots
    # nrow: Number of rows needed, calculated from # of cols
    layout <- matrix(seq(1, cols * ceiling(numPlots/cols)),
                     ncol = cols, nrow = ceiling(numPlots/cols))
  }

 if (numPlots==1) {
    print(plots[[1]])
```

```
  } else {
    # Set up the page
    grid.newpage()
    pushViewport(viewport(layout = grid.layout(nrow(layout), ncol
       (layout))))

    # Make each plot, in the correct location
    for (i in 1:numPlots) {
      # Get the i,j matrix positions of the regions that contain
         this subplot
      matchidx <- as.data.frame(which(layout == i, arr.ind = TRUE
         ))
      print(plots[[i]], vp = viewport(layout.pos.row = matchidx$
         row,
                                      layout.pos.col = matchidx$
                                         col))
    }
  }
}

gg.barplot <- function(ds) {
  n <- dim(ds)[2]
  vs <- {}
  for(i in 1:n) {
    name <- names(ds)[i]
    column <- get(name, ds)
    vs <- rbind(vs, data.frame(column, name))
  }
  spell.counts <- rep(seq(1, dim(ds)[1]), n)
  names(vs) <- c("type", "group")
  data.columns <- cbind.data.frame(vs, spell.counts)
  p <- ggplot(data.columns, aes(spell.counts, type, fill=group))
     +
    geom_bar(stat="identity", position="dodge")
  return(p)
}

plot.counts <- function(vs, wet=FALSE, dry=FALSE) {
  numWetCounts <- 12
  numDryCounts <- 30
  n <- dim(vs)[2]
  wet.counts <- data.frame(matrix(NA, nrow=numWetCounts, ncol=n))
  dry.counts <- data.frame(matrix(NA, nrow=numDryCounts, ncol=n))
  names(wet.counts) <- names(vs)
  names(dry.counts) <- names(vs)
  for(v in 1:n) {
    vec <- vs[, v]
```

```
    for(i in 1:numWetCounts) {
      wet.counts[i, v] <- getSpellCount(vec, i, i + 1, TRUE)
      dry.counts[i, v] <- getSpellCount(vec, i, i + 1, FALSE)
    }
    for(i in (numWetCounts + 1):numDryCounts) {
        dry.counts[i, v] <- getSpellCount(vec, i, i + 1, FALSE)
    }
  }

  log.wet.counts <- log(wet.counts)
  log.dry.counts <- log(dry.counts)
  log.wet.counts <- replace(log.wet.counts, log.wet.counts==-Inf,
      0)
  log.dry.counts <- replace(log.dry.counts, log.dry.counts==-Inf,
      0)

  ps <- {}
  index <- 1
  if(wet) {
    ps[[index]] <- gg.barplot(data.frame(wet.counts)) + ggtitle("
       Wet spell counts") +
              labs(x="Spell length", y="Spell count")
    ps[[index + 1]] <- gg.barplot(data.frame(log.wet.counts)) +
        ggtitle("log(Wet spell counts)") +
              labs(x="Spell length", y="log(Spell count)")
    index <- index + 2
  }
  if(dry) {
    ps[[index]] <- gg.barplot(data.frame(dry.counts)) + ggtitle("
       Dry spell counts") +
              labs(x="Spell length", y="Spell count")
    ps[[index + 1]] <- gg.barplot(data.frame(log.dry.counts)) +
           ggtitle("log(Dry spell counts)") +
              labs(x="Spell length", y="log(Spell count)")
  }
  multiplot(ps, col=length(ps)/2)
}

maskWithNA <- function(srcVec, targetVec)
{
  stopifnot( length(srcVec) == length(targetVec) )
  targetVec[is.na(srcVec)] <- NA
  return(targetVec)
}

# Mask a general time series matrix with the NA values of another
    time series matrix. This goes by row.
```

```
# So, for example, we can mask our Gaussian process with the
   precipOcc for a given range of days.
maskTimeSeriesWithNA <- function(srcMatrix, targetMatrix, start,
   end) {
  masked <- {}
  for(t in start:end) {
    i <- t - start + 1
    m <- maskWithNA(srcMatrix[t, ], targetMatrix[i, ])
    masked <- rbind(masked, m)
  }
  return(masked)
}

priorIndicator <- function(params, lefts, rights) {
  for(i in 1:(length(params))) {
    if(params[i] < lefts[i] || params[i] > rights[i]) {
      return( 0 )
    }
  }
  return ( 1 )
}
```

Appendix : tempSetup.R

```
rm(list=ls())

## Load data
directory <- getwd()
file = "../Data/SetupCOGHCND2.RData"
load(paste(directory, file, sep = '/'))
source("utils.R")

## Removing bad data, whenever tmin > tmax, also removing any
   flags
these <- TN > TX
TN[these] <- NA
TX[these] <- NA
rm(these)
TN[TNQ != " "] <- NA
TX[TXQ != " "] <- NA
PP[PPQ != " "] <- NA

## Units are in 10th of degrees C and 10th of mm
TN <- TN/10
TX <- TX/10
PP <- PP/10

## Loading coordinates
lat <- as.numeric(sf[,2])
lon <- as.numeric(sf[,3])
lon[lon > 0] <- -lon[lon > 0]
lon.lat <- cbind(lon,lat)
elev <- as.numeric(sf[,4])
rm(lon,lat)
np <- length(uniqueID)
nt <- dim(PP)[1]
# one elevation is negative, use nearest elevation from Doug's
   elevation dataset
elev[26] <- 1999.125

## Covariates
ct <- rep(cos((2*pi*(1:365))/365),times=length(unique(yr))) #
   cosines
st <- rep(sin((2*pi*(1:365))/365),times=length(unique(yr))) #
   sines

## Now part of setup
txsum <- apply(!is.na(TX),2,sum)
tnsum <- apply(!is.na(TN),2,sum)
```

```
ppsum <- apply(!is.na(PP),2,sum)

perc <- 0.2 # only look at locations with at least 20% days
   having data
thesekeep <- ppsum/nt > perc & txsum/nt > perc & tnsum/nt > perc
sf <- sf[thesekeep,]
PP <- PP[,thesekeep]
PPQ <- PPQ[,thesekeep]
TN <- TN[,thesekeep]
TNQ <- TNQ[,thesekeep]
TX <- TX[,thesekeep]
TXQ <- TXQ[,thesekeep]
elev <- elev[thesekeep]
lon.lat <- lon.lat[thesekeep,]
np <- sum(thesekeep)
uniqueID <- uniqueID[thesekeep]
rm(perc,ppsum,thesekeep,tnsum,txsum);gc()

pv <- c(1,1:(nt-1))
dr <- seq(-1,1,length.out=nt) # linear drift across time
yrda <- rep(1:365,times=length(unique(yr)))


spatialIndex <- 53 # Glenwood's spatial index
glenwoodData <- TX[, spatialIndex]

dataStart <- 1
dataEnd <- nt
desiredRange <- dataStart:dataEnd
winterMonths <- c(12, 1, 2)
summerMonths <- c(6, 7, 8)
springMonths <- c(3,4,5)
fallMonths <- c(9, 10, 11)

len <- 1
i <- len:nt
cs <- cos(2*pi*i/365)
ss <- sin(2*pi*i/365)
ys <- c(glenwoodData[1], glenwoodData[len:(nt - 1)])

o <- glenwoodData[desiredRange]
```

Appendix : tempFunctions.R

```
# Function definitions
createSimulation <- function(params, data.range) {
  beta0 <- params[1]
  beta1 <- params[2]
  beta2 <- params[3]
  beta3 <- params[4]
  alpha0 <- params[5]
  alpha1 <- params[6]
  alpha2 <- params[7]
  alpha3 <- params[8]
  alpha4 <- params[9]
  sim <- rep(NA, length(data.range))
  sim[1] <- beta0 + beta2*cos(2*pi*data.range[1]/365) +
            beta3*sin(2*pi*data.range[1]/365)
  index <- 2
  loop.range <- data.range[2:length(data.range)]
  for(d in loop.range) {
    mu <- beta0 + beta1*sim[index - 1] + beta2*cos(2*pi*d/365) +
       beta3*sin(2*pi*d/365)
    sigma <- exp(alpha0 + alpha1*cos(2*pi*d/365) + alpha2*sin(2*
       pi*d/365) +
                 alpha3*cos(4*pi*d/365) + alpha4*sin(4*pi*d/365))
    sim[index] <- mu  + rnorm(n = 1, mean=0, sd=sqrt(sigma))
    if(sim[index] == Inf) { sim[index] <- 0 }
    index <- index + 1
  }
  stopifnot( length(sim) == length(mo[data.range]) )
  return(sim)
}

meanError <- function(observed, simulated) {
  value <- abs( mean(observed, na.rm=T) - mean(simulated, na.rm=T
     ) )
  return( value )
}

sdError <- function(observed, simulated) {
  value <- abs( sd(observed, na.rm=T) - sd(simulated, na.rm=T) )
  return( value )
}

estimateMeanErrors <- function(params, observed, data.range) {
  sample.months <- 1:12
  sim <- maskWithNA(observed, createSimulation(params, data.range
     ))
```

```
  error <- 0
  for(month in sample.months) {
    month.obs <- observed[mo[data.range] %in% month]
    month.sim <- sim[mo[data.range] %in% month]
    month.error <- meanError(month.obs, month.sim)/mean(month.obs
       , na.rm=T)
    error <- error + month.error
  }
  error <- error/length(sample.months) # Try to scale with the
     number of sample months
  return(error)
}

estimateSDErrors <- function(params, observed, data.range) {
  sample.months <- 1:12
  sim <- maskWithNA(observed, createSimulation(params, data.range
     ))
  error <- 0
  for(month in sample.months) {
    month.obs <- observed[mo[data.range] %in% month]
    month.sim <- sim[mo[data.range] %in% month]
    month.error <- sdError(month.obs, month.sim)/sd(month.obs, na
       .rm=T)
    error <- error + month.error
  }
  error <- error/length(sample.months) # Try to scale with the
     number of sample months
  return(error)
}

estimateTemp <- function(varParams, meanParams, observed, data.
    range) {
  params <- c(meanParams, varParams)
  winterObs <- observed[mo[data.range] %in% winterMonths]
  summerObs <- observed[mo[data.range] %in% summerMonths]
  springObs <- observed[mo[data.range] %in% springMonths]
  fallObs <- observed[mo[data.range] %in% fallMonths]
  sim <- maskWithNA(observed, createSimulation(params, data.range
     ))
  winterSim <- sim[mo[data.range] %in% winterMonths]
  summerSim <- sim[mo[data.range] %in% summerMonths]
  springSim <- sim[mo[data.range] %in% springMonths]
  fallSim <- sim[mo[data.range] %in% fallMonths]
  winterSDError <- sdError(winterObs, winterSim)/sd(winterObs, na
     .rm=T)
  summerSDError <- sdError(summerObs, summerSim)/sd(summerObs, na
     .rm=T)
```

```
  springSDError <- sdError(springObs, springSim)/sd(springObs, na
     .rm=T)
  fallSDError <- sdError(fallObs, fallSim)/sd(fallObs, na.rm=T)
  sdErrors <- winterSDError + summerSDError + springSDError +
     fallSDError
  print(sprintf("sd errors = %f", sdErrors))
  error <- sdErrors
  return(error)
}
```

Appendix : COTemperature.R

```
data.start <- 31000
data.end <- data.start + 20*365
data.range <- data.start:data.end
o <- glenwoodData[data.range]
month.names = c("Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul",
   "Aug", "Sep", "Oct",
                "Nov", "Dec")

cs2 <- cos(4*pi*(1:nt)/365)
ss2 <- sin(4*pi*(1:nt)/365)

coses <- cos(2*pi*(1:365)/365)
sines <- sin(2*pi*(1:365)/365)
coses2 <- cos(4*pi*(1:365)/365)
sines2 <- sin(4*pi*(1:365)/365)

range.coses <- cos(2*pi*data.range/365)
range.sines <- sin(2*pi*data.range/365)

if(!exists("day.partition")) {
  cat("Getting day partition...", '\n')
  day.partition <- {}
  for(i in 0:364) {
    temp.partition <- rep(NA, nt/365)
    counter <- 1
    for(j in 1:nt) {
      if(j %% 365 == i) {
        temp.partition[counter] <- j
        counter <- counter + 1
      }
    }
    day.partition <- cbind(day.partition, temp.partition)
  }
  day.partition <- cbind(day.partition[, 2:365], day.partition[,
     1])

  spatialIndex <- 15
  o <- TX[, spatialIndex]
  m <- {}
  sds <- {}
  for(i in 1:365) {
    m[i] <- mean(o[day.partition[, i]], na.rm=T)
    sds[i] <- sd(o[day.partition[, i]], na.rm=T)
  }
}
```

```
cat('Forming models for mean and s.d...\n')

# Get linear regression for means and sds of each day of the year
prevs <- c(0, m[2:length(m)])
sd.model <- lm(log(sds) ~ coses + sines + coses2 + sines2)
mean.model <- lm(glenwoodData ~ ys + cs + ss) # + cs2 + ss2)

sd.vals <- fitted.values(sd.model)

get.model.val <- function(d, prev, coefs) {
  res <- coefs[1] + coefs[2]*prev + coefs[3]*cos(2*pi*d/365) +
     coefs[4]*sin(2*pi*d/365)
  return(res)
}

# Check that our MLE model gives good estimates of the mean,
   which it does
mean.vals <- {}
index <- 1
for(d in data.range) {
  mean.vals[index] <- get.model.val(d, glenwoodData[d - 1], mean.
     model$coef)
  index <- index + 1
}

mean.params <- mean.model$coef
sd.params <- sd.model$coef
mean.param.count <- length(mean.params)
sd.param.count <- length(sd.params)
param.count <- mean.param.count + sd.param.count
mean.indices <- 1:length(mean.params)
sd.indices <- (length(mean.indices) + 1):(length(mean.indices) +
   length(sd.params))
cat(mean.indices, sd.indices, '\n')

cat('Starting ABC...\n')
gammas <- c(mean.params, sd.params)
variances <- rep(0.00001, param.count)
sigma <- diag(variances)
unifLeft <- gammas - 2
unifRight <- gammas + 2
unifLeft[2] <- 0; unifRight[2] <- 1
mean.eps <- 0.07
sd.eps <- 0.2
epsilon <- 1
priors <- NA
```

```
desiredCount <- 500
currentCount <- 1
iteration <- 0
accepted <- cbind(gammas, {})
uniform <- TRUE
# Get initial theta_1 via ABC-REJ
while(currentCount < 1) {
  cat("Iterating through ABC-REJ...\n")
  if(uniform) {
    candidates <- rep(NA, 7)
    for(j in 1:param.count) {
      candidates[j] <- runif(n=1, min=unifLeft[j], max=unifRight[
         j])
    }
  }
  else {
    candidates <- mvrnorm(n=1, mu=gammas, Sigma=sigma)
  }
  cmeans <- candidates[mean.indices]; cvars <- candidates[sd.
     indices]
  zetaMean <- estimateMeanErrors(c(cmeans, cvars), o, data.range)
  zetaSD <- estimateSDErrors(c(cmeans, cvars), o, data.range)
  zeta <- abs(zetaMean) + abs(zetaSD)
  print(sprintf("zetaMean = %f, zetaSD = %f, zeta = %f",
                  zetaMean, zetaSD, zeta))
  if( priorIndicator(candidates, unifLeft, unifRight) ) {
    if(!is.na(zeta) && zetaMean < mean.eps && zetaSD < sd.eps) {
        accepted <- cbind(accepted, candidates)
        currentCount <- currentCount + 1
    }
  }
}
# Continue with ABC-MCMC
while(currentCount < desiredCount) {
  print(sprintf("iteration %d", iteration))
  print(sprintf("Acceptances = %d", dim(accepted)[2]))
  candidates <- mvrnorm(n=1, mu=accepted[, currentCount], Sigma
     =0.01*diag(param.count))
  print(candidates)
  cmeans <- candidates[mean.indices]; cvars <- candidates[sd.
     indices]
  zetaMean <- estimateMeanErrors(c(cmeans, cvars), o, data.range)
  zetaSD <- estimateSDErrors(c(cmeans, cvars), o, data.range)
  zeta <- abs(zetaMean) + abs(zetaSD)
  print(sprintf("zetaMean = %f, zetaSD = %f, zeta = %f",
                  zetaMean, zetaSD, zeta))
  if( priorIndicator(candidates, unifLeft, unifRight) ) {
```

```
    if(!is.na(zeta) && zetaMean < mean.eps && zetaSD < sd.eps) {
        accepted <- cbind(accepted, candidates)
        currentCount <- currentCount + 1
    }
  }
  iteration <- iteration + 1
}

# Generate values for prior plots
rvs <- rep(NA, 7)
for(i in 1:1000) {
  if(uniform) {
    for(j in 1:7) {
      rvs[j] <- runif(n=1, min=unifLeft[j], max=unifRight[j])
    }
  }
  else {
    rvs <- mvrnorm(n=1, mu=gammas, Sigma=sigma)
  }
  priors <- cbind(rvs, priors)
}

# Plot priors and posteriors
par(mfrow=c(4,2))
for(i in 1:7) {
  ymax <- max(density(accepted[i, ], na.rm=T)$y,
              density(priors[i, ], na.rm=T)$y)
  xmin <- min(density(accepted[i, ], na.rm=T)$x,
              density(priors[i, ], na.rm=T)$x)
  xmax <- max(density(accepted[i, ], na.rm=T)$x,
              density(priors[i, ], na.rm=T)$x)
  yrange <- c(0, ymax); xrange <- c(xmin, xmax)
  plot(density(accepted[i, ], na.rm=TRUE), col="red", xlim=xrange
     , ylim=yrange)
  lines(density(priors[i, ], na.rm=TRUE), col="darkblue")
}

# Use ggplot to plot priors and posteriors for mean params
titles <- c(expression(beta[0]), expression(beta[1]), expression(
   beta[2]),
            expression(beta[3]), expression(alpha[0]), expression
               (alpha[1]),
            expression(alpha[2]), expression(alpha[3]),
               expression(alpha[4]))
plots <- vector(mode="list", length=mean.param.count)
for(i in 1:mean.param.count) {
  values <- accepted[i, ]
```

```
  xmin <- unifLeft[i]
  xmax <- unifRight[i]
  height <- 1/(unifRight[i] - unifLeft[i])
  d <- data.frame(values)
  plots[[i]] <- ggplot(d, aes(values)) + geom_density(colour="
     brown", alpha=0.4,
                        fill="red") +
                scale_x_continuous(limits=c(xmin, xmax)) +
                annotate("rect", xmin=xmin, xmax=xmax, ymin=0,
                   ymax=height,
                         colour="purple", alpha=0.2, fill="blue")
                             +
                ggtitle(titles[i])
}
multiplot(plots, cols=ceiling(mean.param.count/2))

# Use ggplot to plot priors and posteriors for s.d. params
sd.plots <- vector(mode="list", length=sd.param.count)
for(i in (mean.param.count + 1):param.count) {
  values <- accepted[i, ]
  xmin <- unifLeft[i]
  xmax <- unifRight[i]
  height <- 1/(unifRight[i] - unifLeft[i])
  d <- data.frame(values)
  sd.plots[[i - mean.param.count]] <-
        ggplot(d, aes(values)) + geom_density(colour="brown",
           alpha=0.4,
                        fill="red") +
                scale_x_continuous(limits=c(xmin - 0.2, xmax +
                   0.2)) +
                annotate("rect", xmin=xmin, xmax=xmax, ymin=0,
                   ymax=height,
                         colour="purple", alpha=0.2, fill="blue")
                             +
                ggtitle(titles[i])
}
multiplot(sd.plots, cols=ceiling(sd.param.count/2))

# Create a simulation based on each sample from the posterior
sample.count <- dim(accepted)[2]
sims <- {}
for(j in 1:sample.count) {
  params <- {}
  for(i in 1:param.count) {
    params[i] <- accepted[i, j]
  }
  sims[[j]] <- maskWithNA(o, createSimulation(params, data.range)
```

```
    )
}

# Get mean and sd of each of the sample.count # of samples
sim.means <- {}
sim.sds <- {}
for(j in 1:sample.count) {
  mean.vec <- {}
  sd.vec <- {}
  for(i in 1:12) {
    mean.vec[i] <- mean(sims[[j]][mo[data.range] == i], na.rm=T)
    sd.vec[i] <- sd(sims[[j]][mo[data.range] == i], na.rm=T)
  }
  sim.means <- rbind(sim.means, mean.vec)
  sim.sds <- rbind(sim.sds, sd.vec)
}

obs.means <- obs.sds <- {}
for(i in 1:12) {
  obs.means[i] <- mean(o[mo[data.range] == i], na.rm=T)
  obs.sds[i] <- sd(o[mo[data.range] == i], na.rm=T)
}

# Spaghetti plot of mean
plot(obs.means, col="blue", type='l', lwd=5, ylim=c(0, 36))
for(i in 1:sample.count) {
  lines(sim.means[[i]], col=(i %% 16))
}
lines(obs.means, type='l', lwd=5, ylim=c(0, 35))

# Spaghetti plot of s.d.
plot(obs.sds, col="blue", type='l', lwd=5, ylim=c(0, 8))
for(i in 1:sample.count) {
  lines(sim.sds[[i]], col=(i %% 16))
}
lines(obs.sds, type='l', lwd=5)

# Box plot of means for all samples
sim.df <- stack(data.frame(sim.means))
obs.df <- data.frame(obs.means)
ggplot() + geom_boxplot(d=sim.df, aes(x=sort(ind), y=values),
   outlier.shape=NA) +
          geom_line(d=obs.df, aes(x=seq_along(obs.means), y=obs.
             means), colour="blue") +
          labs(x='Month', y='Mean maximum temperature') +
          scale_x_discrete(breaks=waiver(), labels=month.names)
```

```
# Box plot of sds for all samples
sim.sd.df <- stack(data.frame(sim.sds))
obs.sd.df <- data.frame(obs.sds)
ggplot() + geom_boxplot(d=sim.sd.df, aes(x=sort(ind), y=values),
   outlier.shape=NA) +
           geom_line(d=obs.sd.df, aes(x=seq_along(obs.sds), y=obs
              .sds), colour="blue") +
           labs(x='Month', y='Standard deviation of maximum
              temperature') +
           ylim(0, 7.5) + scale_x_discrete(breaks=waiver(),
              labels=month.names)

# Functional box plots
library(fda)
fbplot(t(sim.means), xlab="Month", ylab="Mean temperature", ylim=
   c(min(sim.means), max(sim.means)), xaxt="n")
lines(obs.means, col="green", lwd=3)
axis(side=1, at=1:12, labels=month.names)
legend('topright', "Observed", lty=1, lwd=3, col="green")

fbplot(t(sim.sds), xlab="Month", ylab="Standard deviation of
   temperature", ylim=c(0, max(sim.sds)), xaxt="n")
lines(obs.sds, col="green", lwd=3)
axis(side=1, at=1:12, labels=month.names)
legend('topright', "Observed", lty=1, lwd=3, col="green")

abcp <- {}
for(i in 1:param.count) {
  abcp[i] <- mean(accepted[i, ])
}

# Make 3 simulations
sim1 <- maskWithNA(o, createSimulation(abcp, data.range))
sim2 <- maskWithNA(o, createSimulation(abcp, data.range))
sim3 <- maskWithNA(o, createSimulation(abcp, data.range))

# Plot empirical mean for simulations and observation
obs.means <- sim.means <- {}
sim.means2 <- sim.means3 <- {}
for(i in 1:12) {
  obs.means[i] <- mean(o[mo[data.range] == i], na.rm=T)
  sim.means[i] <- mean(sim1[mo[data.range] == i], na.rm=T)
  sim.means2[i] <- mean(sim2[mo[data.range] == i], na.rm=T)
  sim.means3[i] <- mean(sim3[mo[data.range] == i], na.rm=T)
}
odf2 <- data.frame(Values=obs.means, Month=1:12, tag="
   Observations")
```

```
sim1df2 <- data.frame(Values=sim.means, Month=1:12, tag="
   Simulation 1")
sim2df2 <- data.frame(Values=sim.means2, Month=1:12, tag="
   Simulation 2")
sim3df2 <- data.frame(Values=sim.means3, Month=1:12, tag="
   Simulation 3")
df <- rbind(odf2, sim1df2, sim2df2, sim3df2)
ggplot(df, aes(x=Month, y=Values, colour=tag)) + geom_line() +
           scale_x_discrete(breaks=waiver(), labels=month.names)

# Plot empirical standard deviation for simulations and
   observation
obs.sds <- sim.sds <- {}
sim.sds2 <- sim.sds3 <- {}
for(i in 1:12) {
  obs.sds[i] <- sd(o[mo[data.range] == i], na.rm=T)
  sim.sds[i] <- sd(sim1[mo[data.range] == i], na.rm=T)
  sim.sds2[i] <- sd(sim2[mo[data.range] == i], na.rm=T)
  sim.sds3[i] <- sd(sim3[mo[data.range] == i], na.rm=T)
}
odf <- data.frame(Values=obs.sds, Month=1:12, tag="Observations")
sim1df <- data.frame(Values=sim.sds, Month=1:12, tag="Simulation
   1")
sim2df <- data.frame(Values=sim.sds2, Month=1:12, tag="Simulation
    2")
sim3df <- data.frame(Values=sim.sds3, Month=1:12, tag="Simulation
    3")
df <- rbind(odf, sim1df, sim2df, sim3df)
ggplot(df, aes(x=Month, y=Values, colour=tag)) + geom_line() +
           ylim(0, 7.5) + scale_x_discrete(breaks=waiver(),
              labels=month.names)

stop()

acceptedCount <- dim(accepted)[2]
i <- 1
while(i <= acceptedCount) {
  par(mfrow=c(2, 2));
  for(j in 0:3) {
    if(i + j <= acceptedCount) {
      plot(o[plotRange], type="l")
      lines(createSimulation(accepted[, i + j], data.range)[
         plotRange], col="red")
    }
    else { break }
  }
  i <- i + 4
```

```
  par(ask=TRUE)
}

accepted <- loadRDA("../Data/GlenwoodTempAcceptedDec25")
abc.params <- {}
for(i in 1:param.count) {
  abc.params[i] <- accepted[i, 500]
}
sim <- maskWithNA(o, createSimulation(abc.params, data.range))

prevs <- glenwoodData[data.range - 1]
mod <- lm(o ~ prevs + cos(2*pi*data.range/365) + sin(2*pi*data.
   range/365))
means <- {}
means[1] <- o[1]
index <- 2
for(i in (data.start+1):data.end) {
  means[index] <- mod$coef[1] + mod$coef[2]*means[index - 1] +
                  mod$coef[3]*cos(2*pi*i/365) + mod$coef[4]*sin(2*
                     pi*i/365)
  index <- index + 1
}

get.count.lengths <- function(v, tau) {
  len <- length(v)
  count.lengths <- {}
  i <- 1
  while(i <= len) {
    if( v[i] >= tau & !is.na(v[i]) ) {
      count <- 1
      i <- i + 1
      while(v[i] >= tau & i <= len & !is.na(v[i])) {
        i <- i + 1
        count <- count + 1
      }
      count.lengths <- c(count.lengths, count)
    } else {
      i <- i + 1
    }
  }
  count.lengths
}

get.cold.count.lengths <- function(v, tau) {
  len <- length(v)
  count.lengths <- {}
  i <- 1
```

```
  while(i <= len) {
    if( v[i] <= tau & !is.na(v[i]) ) {
      count <- 1
      i <- i + 1
      while(v[i] <= tau & i <= len & !is.na(v[i])) {
        i <- i + 1
        count <- count + 1
      }
      count.lengths <- c(count.lengths, count)
    } else {
      i <- i + 1
    }
  }
  count.lengths
}

obs.res <- o
sim.res <- sim

ggplot() + geom_line(data=data.frame(Res=obs.res[1:1000], Label='
   Obs'),
       aes(seq_along(Res), y=Res, color=Label)) +
        geom_line(data=data.frame(Res=sim.res[1:1000], Label='Sim
           '),
       aes(seq_along(Res), y=Res, color=Label)) + labs(x='Day', y
          ='Residual')

o.counts <- get.count.lengths(obs.res, 30)
sim.counts <- get.count.lengths(sim.res, 30)
ds1 <- data.frame(count=o.counts, label='Obs')
ds2 <- data.frame(count=sim.counts, label='Sim')
ds <- rbind(ds1, ds2)
ggplot(ds, aes(factor(count), fill=label)) + geom_histogram(
   position="dodge") +
    labs(title="Hot spells", x='Spell length', y='Spell count')

o.counts <- get.cold.count.lengths(obs.res, 0)
sim.counts <- get.cold.count.lengths(sim.res, 0)
ds1 <- data.frame(count=o.counts, label='Obs')
ds2 <- data.frame(count=sim.counts, label='Sim')
ds <- rbind(ds1, ds2)
ggplot(ds, aes(factor(count), fill=label)) + geom_histogram(
   position="dodge") +
    labs(title="Cold spells", x='Spell length', y='Spell count')
```

Appendix : precipSetup.R

```
rm(list=ls())

source("utils.R")

## Load data
iowa.file <- "../Data/SetupIA.RData"
colorado.file <- "../Data/SetupCOGHCND2.RData"
file = iowa.file
load(file)

# Iowa setup
if(file == iowa.file) {
  # stationfile has lat/lon
  stationfile <- read.csv("../Data/ushcn-stations.csv", header=F)

  uniqueID <- uniqueID[-21]
  PP <- PP[,-21]
  PPQ <- PPQ[,-21]
  TN <- TN[,-21]
  TNQ <- TNQ[,-21]
  TX <- TX[,-21]
  TXQ <- TXQ[,-21]

  lat <- stationfile[,2][stationfile[,1] %in% uniqueID]
  lon <- stationfile[,3][stationfile[,1] %in% uniqueID]
  elev <- stationfile[,4][stationfile[,1] %in% uniqueID]
  lon.lat <- cbind(lon,lat)
  rm(lon,lat)
  dist.mat <- rdist.earth(lon.lat,miles=F)
  diag(dist.mat) <- 0
  PP[PP==-9999]=NA
  TX[TX==-9999]=NA
  TN[TN==-9999]=NA

  np <- length(uniqueID)

  ## Cleanup: kill everything with a flag
  PP[PPQ!=" "] <- NA

  rm(PPQ,TXQ,TNQ) # save space

  OCC <- PP > 0

  ## Get years from 12*31 back to 365, removing all leap days too
  toremove <- mo%in%c(2,4,6,9,11) & da==31
```

```
  toremove[mo==2 & da > 28] <- TRUE
  toremove <- !toremove
  OCC <- OCC[toremove,]
  PP <- PP[toremove,]
  TN <- TN[toremove,]
  TX <- TX[toremove,]
  yr <- yr[toremove]
  mo <- mo[toremove]
  da <- da[toremove]
  rm(toremove)

  ## Covariates
  ct <- rep(cos((2*pi*(1:365))/365),times=length(unique(yr)))
  st <- rep(sin((2*pi*(1:365))/365),times=length(unique(yr)))
  POCC <- OCC
  POCC[1,] <- NA
  POCC[2:dim(POCC)[1],] <- OCC[1:(dim(OCC)[1]-1),]

  ## Precip intensities: set 0 precips to NA
  PPI <- PP
  PPI[!OCC] <- NA
  sf <- stationfile
} else {

## Removing bad data, whenever tmin > tmax, also removing any
   flags
PP[PPQ != " "] <- NA

## Units are in 10th of degrees C and 10th of mm
PP <- PP/10

## Loading coordinates
lat <- as.numeric(sf[,2])
lon <- as.numeric(sf[,3])
lon[lon > 0] <- -lon[lon > 0]
lon.lat <- cbind(lon,lat)
elev <- as.numeric(sf[,4])
rm(lon,lat)
np <- length(uniqueID)
nt <- dim(PP)[1]

# one elevation is negative, use nearest elevation from Doug's
   elevation dataset
elev[26] <- 1999.125

## Now part of setup
ppsum <- apply(!is.na(PP),2,sum)
```

```
perc <- 0.2 # only look at locations with at least 20% days
   having data
thesekeep <- ppsum/nt > perc
sf <- sf[thesekeep,]
PP <- PP[,thesekeep]
PPQ <- PPQ[,thesekeep]
elev <- elev[thesekeep]
lon.lat <- lon.lat[thesekeep,]
np <- sum(thesekeep)
uniqueID <- uniqueID[thesekeep]
rm(perc,ppsum,thesekeep)
gc()
}

# Variable definitions

nt <- dim(PP)[1]

pv <- c(1,1:(nt-1))
dr <- seq(-1,1,length.out=nt) # linear drift across time
yrda <- rep(1:365,times=length(unique(yr)))

fall.months <- c(9, 10, 11)
winter.months <- c(12, 1, 2)
spring.months <- c(3, 4, 5)
summer.months <- c(6, 7, 8)
seasons <- c("fall", "winter", "spring", "summer")

means <- {}
for(i in 1:365) {
  temp.means <- {}
  for(j in 1:nt) {
    if(j %% 365 == i) {
      temp.means <- append(temp.means, j)
    }
  }
  means <- cbind(means, temp.means)
}

dryTaus <- c(1, 5, 9, 13, 17, 21, 25, 30)
wetTaus <- c(1, 2, 4, 6, 8, 10, 12)
paramCount <- 6

glenwoodIndex <- grep("GLENWOOD", sf[, 6])
limonIndex <- grep("LIMON", sf[, 6])
durangoIndex <- grep("DURANGO", sf[, 6])
```

```
hartselIndex <- grep("HARTSEL", sf[, 6])

dataStart <- 1
precipOcc <- PP
precipOcc[precipOcc > 0] <- 1
numLocations <- dim(precipOcc)[2]
spatialIndex <- hartselIndex
spatialIndex <- 13
glenwoodPrecip <- precipOcc[dataStart:nt, spatialIndex]
observations <- precipOcc[, spatialIndex]

occs <- append(NA, observations[1:(nt - 1)])

gp.year.count <- 10
precip.occ.start <- 30000
precip.occ.end <- precip.occ.start + 365*gp.year.count
sim.start <- 1
sim.end <- sim.start + 365*gp.year.count
precip.range <- precip.occ.start:precip.occ.end

# Get the MLE values for each location
rng <- 1:nt
cosines <- cos(2*pi*rng/365)
sines <- sin(2*pi*rng/365)
cosines.2 <- cos(4*pi*rng/365)
sines.2 <- sin(4*pi*rng/365)
locations <- 1:dim(PP)[2]


if(!exists("beta.mles") || is.null(beta.mles)){
  cat("Getting beta MLEs...", '\n')

  beta.mles <- {}
  for(location in locations) {
    occurrences <- precipOcc[, location]
    previousOcc <- append(NA, occurrences[1:(nt-1)])
    d <- data.frame(cbind(occurrences, previousOcc, cosines,
       sines, cosines.2, sines.2))
    model <- glm(occurrences ~ previousOcc + cosines + sines +
       cosines.2 + sines.2,
                 family=binomial(link = "probit"), data=d)
    beta.mles <- cbind(beta.mles, model$coef)
  }
}
```

Appendix : precipFunctions.R

```
# Function definitions
cat("Loading function definitions...", '\n')

# Assumptions: vec is a binary vector
getSpellCount <- function(vec, tau1, tau2, wet) {
  spells <- rle(vec)
  if (wet == TRUE) {
    spells <- spells$lengths[spells$values == 1]
  } else {
    spells <- spells$lengths[spells$values == 0]
  }
  spells <- Filter(function(x) {
                       tau1 <= x
                     }, spells)
  spells <- Filter(function(x) {
                       x < tau2
                     }, spells)
  count <- length(spells)
  return( count )
}

simulatePrecip <- function(dataStart=1, dataEnd=nt, params) {
  beta0 <- params[1]
  beta1 <- params[2]
  beta2 <- params[3]
  beta3 <- params[4]
  beta4 <- params[5]
  beta5 <- params[6]
  occLength <- length(dataStart:dataEnd)
  occ <- rep(NA, occLength)
  occ[1] <- rbinom(1, 1, 0.5) # Arbitrarily assign P[rain on day
     1] = 0.5
  occ.index <- 1
  for(d in (dataStart+1):dataEnd) {
    argument <- beta0 + beta1*occ[occ.index] + beta2*cos(2*pi*d/
       365) +
                beta3*sin(2*pi*d/365) + beta4*cos(4*pi*d/365) +
                beta5*sin(4*pi*d/365)
    p <- pnorm(argument)
    occ[occ.index + 1] <- rbinom(1, 1, prob=p)
    occ.index <- occ.index + 1
  }
  return( occ )
}
```

```
getCountError <- function(vec1, vec2, is.wet) {
  if(is.wet) {
    tau.vec <- wetTaus
  } else {
    tau.vec <- dryTaus
  }
  err <- 0
  actual.counts <- sim.counts <- {}
  for(i in 1:(length(tau.vec - 1))) {
    actual.counts[i] <- getSpellCount(vec1, tau.vec[i],
                                          tau.vec[i + 1],  is.wet
                                             )
    if(actual.counts[i] == 0) actual.counts[i] <- 1
    sim.counts[i] <- getSpellCount(vec2, tau.vec[i],
                                       tau.vec[i + 1], is.wet)
  }
  err <- sum(abs(actual.counts - sim.counts)/actual.counts)/
     length(tau.vec)
  return(err)
}


simulateSinusoidalExponentialGP <- function(params, first.day,
   last.day) {
  a0 <- params[1]
  a1 <- params[2]
  a2 <- params[3]
  tau <- params[4]
  choleskies <- {}
  for(i in 1:365) {
    denom <- exp(a0 + a1*cos(2*pi*i/365) + a2*sin(2*pi*i/365))
    cov.mat <- exp(-rdist.earth(lon.lat)/denom)
    choleskies[[i]] <- t(chol(cov.mat))
  }

  numLocations <- dim(precipOcc)[2]
  day.count <- last.day - first.day
  W <- rbind(rep(0, numLocations), matrix(NA, day.count ,
     numLocations))
  occ <- W
  mus <- rep(0, numLocations)
  samp <- sample.int(dim(mean.accepted)[2], 1)
  for(d in 2:(day.count + 1)) {
    day.of.year <- d + first.day
    for(s in 1:numLocations) {
      mus[s] <- probit.mean(mean.params, occ[d - 1, s], day.of.
         year)
```

```
    }
    eps.normals <- rnorm(length(mus), 0, 1)
    day <- day.of.year %% 365
    Z <- choleskies[[day + 1]] %*% eps.normals
    W[d, ] <- Z + mus + rnorm(length(mus), 0, tau)
    occ[d, ] <- ifelse(W[d, ] > 0, 1, 0)
  }
  return(occ)
}

callVgram <- function(occurrences) {
  variogram <- vgram(lon.lat, occurrences, lon.lat=TRUE, N=20)
  v1 <- variogram$stats[2, ]
  v2 <- variogram$centers
  return(list(v1, v2))
}

# Get the binned variogram for a given day of precipitation data
getDayVariogram <- function(occurrences) {
  # For the bin number, it seems like N = 20 is drastically
     better than N = 15
  variogram <- callVgram(occurrences)[[1]]
  return(variogram)
}

getClusterVariogram <- function(occurrences, start.day, vgram.
   length) {
  start.day <- start.day - 5
  end.day <- start.day + 10
  vs <- rep(0, vgram.length)
  for(day in start.day:end.day) {
    variogram <- getDayVariogram(occurrences[day, ])
    if(length(variogram) == length(vs)) {
      vs <- vs + variogram
    }
  }
  return(vs/length(start.day:end.day)) # Scale by the number of
     trials
}

# Get the aggregate variogram for a given day of the year, over
   many years
getAggregateVariogram <- function(occurrences, start.day=1) {
  temp.variogram <- callVgram(occurrences[start.day, ])
  vgram.length <- length(temp.variogram[[1]])
  vs <- rep(0, vgram.length)
  year.count <- gp.year.count
```

```
  for(i in 1:year.count) { # Loop through sample years
    variogram <- getClusterVariogram(occurrences, start.day + 365
       *i - 5, vgram.length)
       # -5 since GP does not generate the final 5 days
    if(!is.null(variogram) && length(variogram) == length(vs)) {
      variogram[is.na(variogram)] <- 0
      vs <- vs + variogram
    }
  }
  vs <- vs/year.count # Scale by the number of trials
  centers <- temp.variogram[[2]]
  return(list(vs, centers))
}

estimateExponential <- function(params, start, end, obs, obs.
   variogram=NULL) {
  print(params)
  error <- 0
  sim <- simulateSinusoidalExponentialGP(params, start, end)
  gp <- maskWithNA(precipOcc[start:end, ], sim)
  scale.factor <- 12
  for(month in 1:12) {
    obs.start <- 30*month
    sim.start <- 30*month
    obs.variogram <- getAggregateVariogram(precipOcc[start:end,
       ], obs.start)[[1]]
    sim.variogram <- getAggregateVariogram(gp, sim.start)[[1]]
    # Only look at distances less than half the maximum distance
    distance.count <- length(sim.variogram)
    mid <- floor(distance.count/2)
    if(sum(obs.variogram) != 0) {
      month.error <- sum(abs(sim.variogram[1:mid] - obs.variogram
         [1:mid])/abs(obs.variogram[1:mid]))
      print(month.error)
      error <- error + month.error
    } else {
      scale.factor <- scale.factor - 1
    }
  }
  if(error > 0) {
    error <- error/scale.factor # Scale by number of months used
  }
  return(error)
}

gridSearch <- function(a.range, tau.range, precip.start.day) {
  a <- NULL
```

```
  tau <- NULL
  err <- Inf
  iteration.count <- 20
  start.day <- 1
  end.day <- start.day + 365*gp.year.count
  obs.variogram <- getAggregateVariogram(precipOcc, precip.occ.
     start)[[1]]
  for(a.candidate in a.range) {
    for(tau.candidate in tau.range) {
      theta <- c(a.candidate, tau.candidate)
      candidate.err <- estimateExponential(theta, start.day, end.
         day, obs.variogram)
      cat("Error is", candidate.err, '\n')
      if(!is.nan(candidate.err)) {
        if(candidate.err < err) {
          err <- candidate.err
          a <- a.candidate
          tau <- tau.candidate
          cat("Reassigning a to ", a, " and tau to ", tau, '\n')
        }
      }
    }
  }
  return(c(a, tau))
}

estimatePrecip <- function(params, observations, first.day=1,
   last.day=nt, verbose=TRUE,
                             sim=NULL) {
  sample.months <- 1:12
  if(verbose) print(sprintf("params = %f", params))
  if(is.null(sim)) {
    sim <- maskWithNA(observations, simulatePrecip(first.day,
       last.day, params))
  }
  dry.error <- getCountError(observations, sim, FALSE)
  wet.error <- getCountError(observations, sim, TRUE)
  for(season in sample.months) {
    obs.months <- mo[first.day:last.day]
    month.obs <- observations[obs.months %in% season]
    if(length(month.obs) > 0) {
      month.sim <- sim[mo %in% season]
      dry.month.error <- getCountError(month.obs, month.sim,
         FALSE)
      dry.error <- dry.error + dry.month.error
      wet.error <- wet.error + getCountError(month.obs, month.sim
         , TRUE)
```

```
    }
  }
  wet.error <- wet.error/length(sample.months)
  dry.error <- dry.error/length(sample.months)
  total.error <- dry.error + wet.error
  if(verbose) {
    cat("Wet error is ", wet.error, ", Dry error is ", dry.error,
        ", Total is ",
        total.error, '\n')
    print( sprintf("error is %f", total.error) )
  }
  return( total.error )
}

# Make a function for just the argument of the probit regression,
    used for
# the simulation of a thresholded Gaussian Process
probit.mean <- function(params, prev.occ, d) {
  theta0 <- params[1]
  theta1 <- params[2]
  theta2 <- params[3]
  theta3 <- params[4]
  theta4 <- params[5]
  theta5 <- params[6]
  res <- theta0 + theta1*prev.occ + theta2*cos(2*pi*d/365) +
         theta3*sin(2*pi*d/365) + theta4*cos(4*pi*d/365) +
         theta5*sin(4*pi*d/365)
  return(res)
}

probit <- function(params, occ, d) {
  if(length(occ) == 1) {
    return(probit.solo(params, occ, d))
  }
  if(is.na(occ[d])) return(NA)
  if( d < 2 || d > nt ) {
    stop("d must be in [2, nt]")
  }
  theta0 <- params[1]
  theta1 <- params[2]
  theta2 <- params[3]
  theta3 <- params[4]
  theta4 <- params[5]
  theta5 <- params[6]
  res <- theta0 + theta1*occ[d-1] + theta2*cos(2*pi*d/365) +
         theta3*sin(2*pi*d/365) + theta4*cos(4*pi*d/365) +
         theta5*sin(4*pi*d/365)
```

```
  res <- pnorm(res)
  return(res)
}

probit.solo <- function(params, prev.occ, d) {
  theta0 <- params[1]
  theta1 <- params[2]
  theta2 <- params[3]
  theta3 <- params[4]
  theta4 <- params[5]
  theta5 <- params[6]
  res <- theta0 + theta1*prev.occ + theta2*cos(2*pi*d/365) +
         theta3*sin(2*pi*d/365) + theta4*cos(4*pi*d/365) +
         theta5*sin(4*pi*d/365)
  res <- pnorm(res)
  return(res)
}

cat("Functions loaded.")
```

Appendix : precipVault.R

```
# Plot the density of accepted values of a
if(FALSE) {
multiplot(list(
ggplot(data.frame(accepted[1,]) + geom_density(aes(accepted[1,]),
    colour="purple", fill="red", alpha=0.4) +
      scale_x_continuous(limits=c(unif.left, unif.right)),
ggplot(data.frame(accepted[2,]) + geom_density(accepted[2,]),
    colour="purple", fill="red", alpha=0.4) +
      scale_x_continuous(limits=c(unif.left, unif.right))
)))

# Plot the simulated and observed aggregate variogram
test.sim.range <- 1:((365*10*2)+1)
test.obs.range <- test.sim.range + precip.occ.start
theta <- c(mean(accepted[1, ]), mean(accepted[2, ]))
gp <- maskWithNA(precipOcc[test.obs.range, ],
                 simulateExponentialGP(theta, 1, 1+365*gp.year.
                    count))
v1 <- getAggregateVariogram(gp)
v1 <- data.frame(v1)
names(v1) <- c("Values", "Distances")
v2 <- data.frame(getAggregateVariogram(precipOcc, precip.occ.
   start))
names(v2) <- c("Values", "Distances")
ggplot(v1, aes(x=Distances, y=Values)) + geom_point(aes(color="
   Simulated Gaussian process")) +
    geom_point(data=v2, aes(x=Distances, y=Values, color="
       Observations")) + scale_y_continuous(limits=c(0, max(v1$
       Values, v2$Values)))
}

# Linear regression analysis for mean parameters. Not used for
   the thesis,
# but included here for completeness
if(FALSE) {
  # Check spatial dependence
  lons <- lon.lat[, 1]
  lats <- lon.lat[, 2]

  plots <- vector(mode="list", length=4)
  for(i in 1:6) {
    mles.i <- mles[i, ]
    df <- data.frame(mles.i, lons, lats)
    plots[[i]] <- ggplot(df, aes(x=lons, y=lats)) +
      geom_point(aes(color=mles.i)) +
```

```
    scale_colour_gradient(low="white", high="purple")
}
multiplot(plots, col=2)

# Did some model fitting analysis. These seem to work the best
model1 <- lm(mles[1, ] ~ elev*lats*lons)
model2 <- lm(mles[2, ] ~ elev*lats*lons)
model3 <- lm(mles[3, ] ~ elev*lats*lons)
model4 <- lm(mles[4, ] ~ elev*lats*lons)
model5 <- lm(mles[5, ] ~ elev*lats*lons)
model6 <- lm(mles[6, ] ~ elev*lats*lons)

for(i in 1:6) {
  assign(paste("model", i, "reduced", sep=''), lm(mles[i, ] ~
     elev*lats*lons))
  assign(paste("model", i, "reduced", sep='.'),
         eval(parse(text=paste("lm(", summary(glmulti(mles[i, ]
             ~ lons + lats + elev, crit="BIC"))$bestmodel, ")",
             sep=''))))
}

# BIC Model Evaluation
for(i in 1:6) {
  model.i <- eval(parse(text=paste("model", i, sep='')))
  reduced.i <- eval(parse(text=paste("model", i, "reduced", sep
     ='.')))
  if( BIC(model.i) < BIC(reduced.i) ) {
    assign( paste("best.model", i, sep=''), model.i )
  } else {
    assign( paste("best.model", i, sep=''), reduced.i )
  }
  cat("Full model: ", BIC(model.i), ", Reduced model: ", BIC(
     reduced.i), ", Best model: ",
      BIC(eval(parse(text=paste("best.model", i, sep='')))), '\
         n', sep='')
}

# Estimate mles with alphas and simulate to see how good fit is
estimated.mles <- {}
for(i in 1:6) {
  model.i <- eval(parse(text=paste("model", i, sep='')))
  mle <- fitted.values(model.i)
  estimated.mles <- rbind(estimated.mles, mle)
}

plots <- vector(mode="list", length=4)
for(i in 1:6) {
```

```
    mles.i <- mles[i, ]
    df <- data.frame(mles.i, lons, lats)
    plots[[2*i - 1]] <- ggplot(df, aes(x=lons, y=lats)) +
      geom_point(aes(color=mles.i)) +
      scale_colour_gradient(low="white", high="red")
  }
  for(i in 1:6) {
    mles.i <- estimated.mles[i, ]
    df <- data.frame(mles.i, lons, lats)
    plots[[2*i]] <- ggplot(df, aes(x=lons, y=lats)) +
      geom_point(aes(color=mles.i)) +
      scale_colour_gradient(low="white", high="red")
  }
  multiplot(plots, col=6)

  par(mfrow=c(2, 2))
  for(i in 1:4) {
    plot(mles[i, ] ~ estimated.mles[i, ])
  }
}

if(FALSE) {
  get.reduced.model <- function(index) {
    res <- eval(parse(text=paste("model", i, "reduced", sep='.'))
       )
    return(res)
  }

  check.alpha.length <- function(alpha.length, index) {
    len <- switch(toString(index),
      "1" = 5, "2" = 6, "3" = 5, "4" = 3, "5" = 4, "6" = 5
    )
    if(alpha.length != len) {
      stop(cat("Alpha length mismatch; length(alphas) = ", alpha.
         length, ", index = ", index))
    }
  }

  get.beta.model <- function(alphas, beta.index, s) {
    check.alpha.length(length(alphas), beta.index)
    lon <- lons[s]
    lat <- lats[s]
    el <- elev[s]
    covs <- switch(toString(beta.index),
      "1" = {
        c(1, lon, el, el*lon, el*lat)
      },
```

```
    "2" = {
      c(1, lon, lat, el, lat*lon, el*lon)
    },
    "3" = {
      c(1, el, lat*lon, el*lon, el*lat)
    },
    "4" = {
      c(1, el*lon, el*lat)
    },
    "5" = {
     c(1, el, lat*lon, el*lat)
    },
    "6" = {
     c(1, lon, lat, lat*lon, el*lat)
    }
  )
  res <- (alphas %*% covs)[1, 1]
  return(res)
}

get.params.from.list.index <- function(target.list, beta.index,
    alpha.index) {
  v <- {}
  len <- length(target.list)
  for(i in 1:len) {
    v[[i]] <- target.list[[i]][[beta.index]][alpha.index]
  }
  return(v)
}

get.beta.from.alphas <- function(alphas, s) {
  betas <- {}
  for(i in 1:length(alphas)) {
    betas[i] <- get.beta.model(alphas[[i]], i, s)
  }
  return(betas)
}

for(i in 1:6) {
  len <- length(abc.multisite.accepted[[1]][[i]])
  alpha.list <- {}
  for(j in 1:len) {
    param.vals <- get.params.from.list.index(abc.multisite.
       accepted, i, j)
    alpha.list[j] <- mean(param.vals)
  }
  abc.multisite.params[[i]] <- alpha.list
```

```
}

beta.from.alphas <- get.beta.from.alphas(abc.multisite.params,
   spatialIndex)

cat("Running spatial ABC...", '\n')

variances <- rep(0.0001, paramCount)
alphas0 <- {}
unifLefts <- unifRights <- {}
variances <- 0.1*c(1, 1, 1, 1, 1, 1)

abc.multisite.accepted <- {}
currentCount <- 0
jumps <- 0
for(i in 1:paramCount) {
  alphas0[[i]] <- get.reduced.model(i)$coef
  unifLefts[[i]] <- alphas0[[i]] - 1
  unifRights[[i]] <- alphas0[[i]] + 1
}
alphas <- alphas0

epsilon <- 130

while(TRUE) {
  cat("Acceptances = ", length(abc.multisite.accepted), '\n')
  cat("Jumps = ", jumps, '\n')
  betas <- {}
  candidates <- {}
  for(i in 1:paramCount) {
    alpha.i <- alphas[[i]]
    candidates[[i]] <- mvrnorm(n=1, mu=alpha.i, Sigma
       =0.000000000001*diag(length(alpha.i)))
    betas[i] <- get.beta.model(candidates[[i]], i, spatialIndex
       )
  }
  rho <- estimatePrecip(betas, observations)
  cat("Rho = ", rho, '\n')
  if(!is.na(rho) && rho < epsilon) {
    alphas <- candidates
    jumps <- jumps + 1
  }
  if(jumps >= 1) {
    currentCount <- currentCount + 1
    abc.multisite.accepted[[currentCount]] <- alphas
  }
}
```

```
  plots <- list(list(rep(NA, 5)), list(rep(NA, 6)), list(rep(NA,
     5)), list(rep(NA, 3)), list(rep(NA, 4)), list(rep(NA, 5)))
  for(i in 1:6) {
    len <- length(abc.multisite.accepted[[1]][[i]])
    for(j in 1:len) {
      param.vals <- get.params.from.list.index(abc.multisite.
         accepted, i, j)
      xmin <- unifLefts[[i]][[j]]
      xmax <- unifRights[[i]][[j]]
      height <- 1/(xmax - xmin)
      d <- data.frame(param.vals)
      plots[[i]][[j]] <- ggplot(d) + geom_density(aes(x=param.
         vals))
    }
  }

  for(i in 1:6) {
    multiplot(plots[[i]])
    readline(prompt="Press enter to continue")
  }


  get.estimated.beta <- function(beta.index, spatial.index) {
    beta.model <- eval(parse(text=paste("model", beta.index, "
       reduced", sep='.')))
    beta.model <- get.beta.model(beta.index)
    res <- fitted.values(beta.model)[spatial.index]
    return(res)
  }
}

# Plot empiricial probabilities throughout the year
plot.empirical.means <- function(ds) {
  ggplot(ds, aes(x=seq_along(m))) +
    geom_point(aes(y=m, colour="actual means")) +
    geom_line(aes(y=model1.vals, colour="sine fit to actual means
       ")) +
    geom_point(aes(y=m.abc, colour="abc means")) +
    geom_line(aes(y=model.abc.vals, colour="sine fit to abc means
       ")) +
    geom_point(aes(y=m.post, colour="posterior means")) +
    geom_line(aes(y=model.post.vals, colour="sine fit to
       posterior means")) +
    labs(x="Day of year", y="Empirical mean of precip occurrence"
       ) +
    ggtitle(paste("Precip occurrence for", sf[spatialIndex, 6],
```

```
        sf[spatialIndex, 7], sep=' '))
}

plot.empirical.sds <- function(ds2, index) {
  ggplot(ds2, aes(x=seq_along(sd1))) +
    geom_point(aes(y=sd1, colour="Actual SDs")) +
    geom_point(aes(y=sd.abc, colour="ABC SDs")) +
    geom_point(aes(y=sd.post, colour="True posterior SDs")) +
    ylim(0, max(sd1)) +
    labs(x="Day of year", y="Empirical standard deviation of
       precip occurrence")
}

uniformIndicator <- function(params) {
  for(i in 1:(length(params))) {
    if(params[i] < unifLeft[i] || params[i] > unifRight[i]) {
      return( 0 )
    }
  }
  return ( 1 )
}

prob <- function(p, occ) {
  res <- p^occ * (1 - p)^(1 - occ)
  return(res)
}


log.posterior <- function(params, o) {
  psum <- 0
  for(d in 2:nt) {
    if(!is.na(o[d - 1]) && !is.na(o[d])) {
      if(o[d]) {
        psum <- psum + log(probit(params, o, d))
      } else if (!o[d]) {
        psum <- psum + log(1 - probit(params, o, d))
      }
    }
  }
  res <- psum - log.prior
  return(res)
}

### TESTING AGAINST TRUE POSTERIOR ###

do.posterior <- FALSE
if(do.posterior) {
```

```
theta0 <- glm(observations ~ occs + coses + sines + coses2 +
   sines2,
              family=binomial(link="probit"))$coef
variances <- c(1, 1, 1, 1, 1, 1)
sigma <- diag(variances)
unifLeft <- c(-1.5, 0, -0.5, -0.5, -0.5, -0.5)
unifRight <- c(1.5, 1, 0.5, 0.5, 0.5, 0.5)
log.prior <- sum(log(unifRight - unifLeft))
uniform=TRUE

currentCount <- 0
jumps <- 0
desiredCount <- Inf
posterior.accepted <- {}
theta <- theta0

while(currentCount < desiredCount) {
  print(sprintf("Acceptances = %d", dim(posterior.accepted)[2])
     )
  print(sprintf("Jumps = %d", jumps))
  candidates <- mvrnorm(n=1, mu=theta,
                        Sigma=0.0002*diag(length(theta)))
  print(candidates)
  if(uniformIndicator(candidates)) {
    phi1 <- log.posterior(candidates, observations)
    phi2 <- log.posterior(theta, observations)
    difference <- phi1 - phi2
    alpha <- min(difference, 0)
    u <- runif(1, min=0, max=1)
    cat("log(u) = ", log(u), ", alpha = ", alpha, '\n')
    if(log(u) <= alpha) {
        jumps <- jumps + 1
        theta <- candidates
    }
  }
  posterior.accepted <- cbind(posterior.accepted, theta)
  currentCount <- currentCount + 1
}

titles <- c("Constant", "A-R", "cos(*)", "sin(*)", "cos(2*)", "
   sin(2*)")
paramCount <- length(theta0)

plots <- list()
for(i in 1:6) {
  r <- posterior.accepted[i, ]
  xmin <- unifLeft[i]
```

```
    xmax <- unifRight[i]
    height <- 1/(unifRight[i] - unifLeft[i])
    d <- data.frame(r)
    plots[[i]] <- ggplot(d, aes(r)) + geom_density(colour="brown"
       , alpha=0.4, fill="red") +
      scale_x_continuous(limits=c(xmin - 0.2, xmax + 0.2)) +
      annotate("rect", xmin=xmin, xmax=xmax, ymin=0, ymax=height,
          colour="purple", alpha=0.2, fill="blue") +
      ggtitle(titles[i])
  }
  multiplot(plots, col=ceiling(paramCount/2))
}

### END TESTING AGAINST TRUE POSTERIOR ###

stop("End of file")

spatialIndex <- 13
theta0 <- {}
for(i in 1:6) {
  theta0[i] <- mean(posterior.accepted[i, ])
}
epsilon <- 1.09
currentCount <- 0
desiredCount <- Inf
variances <- c(1, 1, 1, 1, 1, 1)
sigma <- diag(variances)
uniform <- TRUE
abc.accepted <- cbind(theta0)
currentCount <- 1
jumps <- 0
theta <- theta0
# ABC-MCMC
while(currentCount < desiredCount) {
  print(sprintf("Acceptances = %d", dim(abc.accepted)[2]))
  print(sprintf("Jumps = %d", jumps))
  in.prior <- FALSE
  while(!in.prior) {
    candidates <- mvrnorm(n=1, mu=theta,
                          Sigma=0.00005*diag(dim(abc.accepted)[1]))
    if(priorIndicator(candidates, unifLeft, unifRight)) {
      in.prior <- TRUE
    }
  }
  zeta <- estimatePrecip(candidates, observations)
  cat(paste("\n", "Zeta = ", zeta, "\n\n"))
  if(!is.na(zeta) && zeta < epsilon) {
```

```
      theta <- candidates
      jumps <- jumps + 1
  }
  currentCount <- currentCount + 1
  abc.accepted <- cbind(abc.accepted, theta)
}

abc.params <- {}
for(i in 1:paramCount) {
  abc.params[i] <- mean(abc.accepted[i, ])
}

abc.simulation <- maskWithNA(observations, simulatePrecip(1, nt,
   abc.params))
ds <- data.frame(observations, abc.simulation)
names(ds) <- c("Observations", "Simulation")
plot.counts(ds)

plots <- list()
for(i in 1:6) {
  r <- posterior.accepted[i, ]
  r2 <- abc.accepted[i, ]
  xmin <- unifLeft[i]
  xmax <- unifRight[i]
  height <- 1/(unifRight[i] - unifLeft[i])
  rr <- data.frame(X = r)
  rr2 <- data.frame(X = r2)
  rr$group <- "True posterior"
  rr2$group <- "ABC posterior"
  rLengths <- rbind(rr, rr2)
  plots[[i]] <- ggplot(rLengths, aes(x=X, fill=group, colour=
     group)) + geom_density(alpha = 0.4) +
    scale_x_continuous(limits=c(xmin - 0.2, xmax + 0.2)) +
    annotate("rect", xmin=xmin, xmax=xmax, ymin=0, ymax=height,
       colour="purple", alpha=0.2, fill="blue") +
  ggtitle(titles[i])
}
multiplot(plots, col=ceiling(paramCount/2))

abc.params <- posterior.params <- {}
for(i in 1:paramCount) {
  posterior.params[i] <- mean(posterior.accepted[i, ])
  abc.params[i] <- mean(abc.accepted[i, ])
}

posterior.simulation <- maskWithNA(observations,
                          simulatePrecip(1, nt, posterior.params))
```

```
abc.simulation <- maskWithNA(observations, simulatePrecip(1, nt,
   abc.params))
ds <- data.frame(observations, abc.simulation, posterior.
   simulation)
names(ds) <- c("Observations", "Abc simulation", "True posterior
   simulation")
plot.counts(ds)

m <- m.abc <- m.post <- {}
sd1 <- sd.abc <- sd.post <- {}
for(i in 1:364) {
  m[i] <- mean(observations[means[, i]], na.rm=TRUE)
  m.abc[i] <- mean(abc.simulation[means[, i]], na.rm=TRUE)
  m.post[i] <- mean(posterior.simulation[means[, i]], na.rm=TRUE)
  sd1[i] <- sd(observations[means[, i]], na.rm=TRUE)
  sd.abc[i] <- sd(abc.simulation[means[, i]], na.rm=TRUE)
  sd.post[i] <- sd(posterior.simulation[means[, i]], na.rm=TRUE)
}

# Check to see what the precip means look like over the year for
   8 different locations.
# They seem to be very similar, which will motivate a common beta
    mean estimate
# across all sites
locs <- sample.int(22, 8)
par(mfrow=c(2, 4))
for(j in 1:8) {
  o <- precipOcc[, locs[j]]
  m <- {}
  for(i in 1:364) {
    m[i] <- mean(o[means[, i]], na.rm=TRUE)
  }
  plot(m)
}

cos.year <- cos(2*pi*(1:364)/365)
sin.year <- sin(2*pi*(1:364)/365)
cos.year.2 <- cos(4*pi*(1:364)/365)
sin.year.2 <- sin(4*pi*(1:364)/365)

abc.vals <- m[1]
post.vals <- m[1]
for(i in 1:(2*364)) {
  abc.vals[i+1] <- pnorm(abc.params[1] + abc.params[2]*abc.vals[i
     ] +
                  abc.params[3]*cos(2*pi*i/365) + abc.params[4]*
                     sin(2*pi*i/365) +
```

```
                    abc.params[5]*cos(4*pi*i/365) + abc.params[6]*
                       sin(4*pi*i/365))
  post.vals[i+1] <- pnorm(posterior.params[1] + posterior.params
     [2]*post.vals[i] +
                    posterior.params[3]*cos(2*pi*i/365) +
                    posterior.params[4]*sin(2*pi*i/365) +
                    posterior.params[5]*cos(4*pi*i/365) +
                    posterior.params[6]*sin(4*pi*i/365))
}

model1 <- lm(m ~ cos.year + sin.year + cos.year.2 + sin.year.2)
model1.vals <- fitted.values(model1)

model.post <- lm(m.post ~ cos.year + sin.year + cos.year.2 + sin.
   year.2)
model.post.vals <- fitted.values(model.post)

model.abc <- lm(m.abc ~ cos.year + sin.year + cos.year.2 + sin.
   year.2)
model.abc.vals <- fitted.values(model.abc)

ggplot() + geom_point(d=data.frame(Vals=m), aes(x=seq_along(Vals)
   , y=Vals,
        colour="Observed empirical means")) +
    geom_line(d=data.frame(Vals=post.vals[366:(2*365)]), aes(x=
       seq_along(Vals), y=Vals,
        colour="True posterior mean function")) +
    geom_line(d=data.frame(Vals=abc.vals[366:(2*365)]), aes(x=seq
       _along(Vals), y=Vals,
        colour="ABC mean function")) +
    labs(x="Day of year", y="Probability of rain") +
    theme(legend.title=element_blank())

ggplot() + geom_point(d=data.frame(Vals=m), aes(x=seq_along(Vals)
   , y=Vals,
        colour="Observed empirical means")) +
    geom_point(d=data.frame(Vals=m.abc), aes(x=seq_along(Vals), y
       =Vals,
        colour="ABC empirical means")) +
    geom_line(d=data.frame(Vals=abc.vals[366:(2*365)]), aes(x=seq
       _along(Vals), y=Vals,
        colour="ABC mean function")) +
    labs(x="Day of year", y="Probability of rain") +
    theme(legend.title=element_blank())

ds <- data.frame(m, m.abc, m.post, model1.vals, model.abc.vals,
   model.post.vals)
```

```
plot.empirical.means(ds)

ds2 <- data.frame(sd1, sd.abc, sd.post)
plot.empirical.sds(ds2)

m.avg <- {}
for(i in 1:12) {
  m.avg[i] <- sd(observations[mo %in% i], na.rm=T)
}

d1 <- data.frame(Vals=sd.abc, Month=mo[1:364], G="ABC")
d2 <- data.frame(Vals=sd.post, Month=mo[1:364], G="True posterior
    ")
d3 <- data.frame(Vals=sd1, Month=mo[1:364], G="Observations")
ggplot(rbind.data.frame(d1, d2, d3)) + geom_boxplot(aes(x=factor(
   Month), y=Vals,
    position="dodge",fill=G), outlier.shape=NA) +
    labs(x="Month", y="Empirical standard deviation of
       precipitation") +
    theme(legend.title=element_blank())

sum(observations[observations == 1], na.rm=T) + sum(!observations
   [observations == 0], na.rm=T) + length(observations[is.na(
   observations)])

sum(abc.simulation[abc.simulation == 1], na.rm=T) + sum(!abc.
   simulation[abc.simulation == 0], na.rm=T) + length(abc.
   simulation[is.na(abc.simulation)])

sum(posterior.simulation[posterior.simulation == 1], na.rm=T) +
   sum(!posterior.simulation[posterior.simulation == 0], na.rm=T)
    + length(posterior.simulation[is.na(posterior.simulation)])
```

Appendix : precipSpatial.R

```
gp.year.count <- 20
precip.occ.start <- 1
precip.occ.end <- nt
precip.range <- precip.occ.start:precip.occ.end
sim.start <- 1
sim.end <- sim.start + 365*gp.year.count

location.count <- dim(precipOcc)[2]
data.start <- precip.occ.start
data.end <- precip.occ.end
data.range <- data.start:data.end
observations <- precipOcc[, 1]
coses <- cos(2*pi*(1:nt)/365)
sines <- sin(2*pi*(1:nt)/365)
coses2 <- cos(4*pi*(1:nt)/365)
sines2 <- sin(4*pi*(1:nt)/365)
occs <- append(NA, observations[1:(nt - 1)])

computePriors <- function(xs, means, sds) {
  p <- 1
  for(i in 1:6) {
    p <- p*dnorm(xs[i], means[i], sds[i])
  }
  return(p)
}

solo.epsilon <- 1.15
epsilon <- solo.epsilon*location.count
if(!exists("theta0")) {
  theta0 <- rep(0, 6)
  for(i in 1:22) {
  observations <- precipOcc[, 13]
  occs <- append(NA, observations[1:(nt - 1)])
  theta0 <- theta0 + glm(observations ~ occs + coses + sines +
     coses2 + sines2,
                family=binomial(link="probit"))$coef
  }
  theta0 <- theta0/22
  cat(theta0, '\n')
}
currentCount <- 0
desiredCount <- Inf
variances <- c(0.5, 1, 1, 1, 1, 1)
unifLeft <- theta0 - variances; unifRight <- theta0 + variances
uniform <- TRUE
```

```
abc.accepted <- cbind(theta0)
currentCount <- 1
jumps <- 0
prior.mean <- theta0 + 0.05
theta <- theta0
# ABC-MCMC
while(currentCount < desiredCount) {
  print(sprintf("Acceptances = %d", dim(abc.accepted)[2]))
  print(sprintf("Jumps = %d", jumps))
  in.prior <- FALSE
  while(!in.prior) {
    candidates <- mvrnorm(n=1, mu=theta,
                          Sigma=0.0001*diag(variances))
    if(priorIndicator(candidates, unifLeft, unifRight)) {
      in.prior <- TRUE
    }
  }
  cat(candidates, '\n')
  zeta <- 0
  for(loc in 1:location.count) {
    observations <- precipOcc[precip.range, loc]
    zeta.loc <- estimatePrecip(candidates, observations, precip.
       occ.start,
                               precip.occ.end, FALSE)
    cat(zeta.loc, zeta, '\n')
    zeta <- zeta + zeta.loc
  }
  cat(paste("\n", "Zeta = ", zeta, "\n\n"))
  if(!is.na(zeta) && zeta < epsilon) {
      u <- runif(1, 0, 1)
      alpha <- min(computePriors(candidates, prior.mean, rep(0.1,
          6))/
                   computePriors(theta, prior.mean, rep(0.1, 6)),
                       1)
      cat(u, alpha, '\n')
      if(u < alpha) {
        theta <- candidates
        jumps <- jumps + 1
      }
  }
  currentCount <- currentCount + 1
  abc.accepted <- cbind(abc.accepted, theta)
}

s <- sample.int(dim(abc.accepted)[2], 1)
print(s)
abc.params <- {}
```

```
for(i in 1:paramCount) {
  abc.params[i] <- mean(abc.accepted[i, s])
}

titles <- c(expression(beta[0]), expression(beta[1]), expression(
   beta[2]),
            expression(beta[3]), expression(beta[4]), expression(
               beta[5]))
plots <- {}
pdf("meandensities.pdf", width=8, height=10)
for(i in 1:6) {
  center <- prior.mean[i]
  print(center)
  xs <- seq(center - 0.3, center + 0.3, length.out=1000)
  prior.vals <- dnorm(xs, center, 0.1)
  plots[[i]] <- ggplot() +
                geom_area(d=data.frame(pVals=prior.vals, X=xs),
                    aes(x=X, y=pVals, fill="Prior"), alpha=0.4,
                       colour="black") +
                geom_density(d=data.frame(Vals=abc.accepted[i, ])
                   ,
                   aes(Vals, fill="ABC Posterior"), alpha=0.4) +
                scale_x_continuous(name="") + scale_y_continuous(
                   name="") +
                theme(legend.title=element_blank(), legend.
                   justification=c(1,1),
                     legend.position=c(1,1), legend.text=element_
                        text(size=6)) +
                 ggtitle(titles[i])
}
multiplot(plots, cols=2)
dev.off()

# Get abc params and set up for simulations
accepted <- loadRDA("../Data/CovarianceParams") # Covariance
   theta
samp <- sample.int(dim(accepted)[2], 1)

abc.theta <- {}
for(i in 1:4) {
  abc.theta[i] <- mean(accepted[i, ])
}

abc.betas <- {}
for(i in 1:6) {
  abc.betas[i] <- mean(abc.accepted[i, ])
}
```

```
abc.params <- theta0
mles <- m[1]
for(i in 1:(2*364)) {
  mles[i+1] <- pnorm(abc.params[1] + abc.params[2]*mles[i] +
                   abc.params[3]*cos(2*pi*i/365) + abc.params[4]*
                      sin(2*pi*i/365) +
                   abc.params[5]*cos(4*pi*i/365) + abc.params[6]*
                      sin(4*pi*i/365))
}
index <- sample.int(22, 1)

ms <- {}
ms[1] <- mles[1]
for(i in 1:(2*365)) {
  ms[i + 1] <- pnorm(abc.betas[1] + abc.betas[2]*ms[i] +
    abc.betas[3]*cos(2*pi*i/365) + abc.betas[4]*sin(2*pi*i/365) +
    abc.betas[5]*cos(4*pi*i/365) + abc.betas[6]*sin(4*pi*i/365)
  )
}

colors = rainbow(22, alpha=0.8)
plot(mles[366:(2*365)], ylim=c(0, 0.5), lwd=1, col="grey", xlab="
   Day of year",
        ylab="Empirical probability of precipitation")
for(loc in 1:22) {
  m <- {}
  for(i in 1:364) {
    m[i] <- mean(precipOcc[means[, i], loc], na.rm=TRUE)
  }
  lines(m, col=colors[loc])
}
lines(mles[366:(2*365)], ylim=c(0, 1), lwd=5, col="grey")

ms <- {}
for(loc in 1:22) {
  m <- {}
  for(i in 1:364) {
    m[i] <- mean(precipOcc[means[, i], loc], na.rm=TRUE)
  }
  m <- cbind(m, day=mo[1:364])
  ms <- rbind.data.frame(ms, m)
}

short.month.names = c("Jan","Feb", "Mar", "Apr", "May", "Jun", "
   Jul", "Aug", "Sep",
                      "Oct", "Nov", "Dec")
```

```
pdf("meanboxplot.pdf", width=6, height=6)
ggplot() +
    geom_boxplot(d=ms, aes(x=factor(day), y=m), outlier.shape=NA)
        +
    geom_line(d=data.frame(Vals=mles[366:730]),
              aes(x=seq(1, 12, length.out=length(Vals)), y=Vals),
                  colour="blue", lwd=2) + ylim(0, 0.5) +
    scale_x_discrete(labels=short.month.names) +
    labs(x="Month", y="Empirical probability of precipitation
       occurrence")
dev.off()

locs <- sample.int(22, 8)
par(mfrow=c(2, 4))
for(j in 1:8) {
  o <- precipOcc[, locs[j]]
  sim <- maskWithNA(o, simulatePrecip(1, nt, abc.params))
  m <- {}
  m2 <- {}
  for(i in 1:364) {
    m[i] <- mean(o[means[, i]], na.rm=TRUE)
    m2[i] <- mean(sim[means[, i]], na.rm=TRUE)
  }
  lo <- loess.smooth(1:length(m2), m2, span=1/8)
  plot(m, ylim=c(0, max(m))); lines(lo$x, lo$y, col="blue")
}

sample.count <- dim(abc.accepted)[2]
sims <- {}
for(j in 1:sample.count) {
  params <- {}
  for(i in 1:param.count) {
    params[i] <- abc.accepted[i, j]
  }
  sims[[j]] <- maskWithNA(o, createSimulation(params, data.range)
     )
}

# Get mean and sd of each of the sample.count # of samples
sim.means <- {}
sim.sds <- {}
for(j in 1:sample.count) {
  mean.vec <- {}
  sd.vec <- {}
  for(i in 1:12) {
    mean.vec[i] <- mean(sims[[j]][mo[data.range] == i], na.rm=T)
    sd.vec[i] <- sd(sims[[j]][mo[data.range] == i], na.rm=T)
```

```
  }
  sim.means <- rbind(sim.means, mean.vec)
  sim.sds <- rbind(sim.sds, sd.vec)
}

gridSearchExp <- function(a0.range, a1.range, a2.range, tau.range
   ) {
  a0 <- a1 <- a2 <- tau <- NULL
  err <- Inf
  start.day <- 1
  end.day <- start.day + 365*(gp.year.count + 1)
  obs <- precipOcc[sim.start:sim.end, ]
  for(a0.cand in a0.range) {
    for(a1.cand in a1.range) {
      for(a2.cand in a2.range) {
        for(tau.cand in tau.range) {
          cand.theta <- c(a0.cand, a1.cand, a2.cand, tau.cand)
          print(cand.theta)
          gp <- maskWithNA(obs, simulateSinusoidalExponentialGP(
             cand.theta,
                          sim.start, sim.end))
          error <- 0
          for(month in 1:12) {
            gp.start <- 30*month
            occ.start <- 30*month # Shift by the given month

            v1 <- getAggregateVariogram(gp, gp.start)
            v2 <- getAggregateVariogram(obs, occ.start)
            month.error <- sum(abs(v1[[1]] - v2[[1]])/v2[[1]])
            error <- error + month.error
          }
          error <- error/12
          if(error < err) {
            err <- error
            a0 <- a0.cand
            a1 <- a1.cand
            a2 <- a2.cand
            tau <- tau.cand
            cat('Error is ', err, ', a0 = ', a0, ', a1 = ', a1, '
               , a2 = ', a2,
                ', tau = ', tau, '\n')
          }
        }
      }
    }
  }
  return(c(a0, a1, a2, tau))
```

```
}

mean.accepted <- abc.accepted # Mean function theta

sim.start <- 30000
sim.end <- sim.start + (gp.year.count+1)*365
unif.left <- c(5.5, -0.5, -0.5, 0.2)
unif.right <- c(7.3, 0.5, 0.5, 0.6)
a0 <- 6.5
a1 <- 0
a2 <- 0
as <- c(a0, a1, a2)
tau <- 0.40076
sigma <- diag(0.5*c(0.1, 0.01, 0.01))
accepted <- {}
theta <- c(as, tau)
currentCount <- 0
desiredCount <- Inf
jumps <- 0
epsilon <- 1

# ABC-MCMC for Exponential(-||h||/A(t))
while(currentCount < desiredCount) {
  cat("Acceptances = ", dim(accepted)[2], '\n')
  cat("Jumps = ", jumps, '\n')
  in.prior <- FALSE
  while(!in.prior) {
    as <- theta[1:3]
    tau <- theta[4]
    a.candidate <- mvrnorm(n=1, mu=as, Sigma=sigma)
    print(exp(a.candidate[1] + a.candidate[2] + a.candidate[3]))
    tau.candidate <- rnorm(n=1, mean=tau, sd=0.01)
    candidate <- c(a.candidate, tau.candidate)
    if(priorIndicator(candidate, unif.left, unif.right)) {
      in.prior <- TRUE
    }
  }
  zeta <- estimateExponential(candidate, sim.start, sim.end,
     precipOcc)
  cat(paste("\n", "Zeta = ", zeta, "\n\n"))
  if(!is.na(zeta) && zeta < epsilon) {
      theta <- candidate
      jumps <- jumps + 1
  }
  currentCount <- currentCount + 1
  accepted <- cbind(accepted, theta)
}
```

```
stop("Stopping")

# Set up sim range and observation vector
cov.theta <- {}
for(i in 1:4) {
  cov.theta[i] <- mean(accepted[i, ])
}
test.sim.start <- 29940
test.sim.end <- test.sim.start + 365*(gp.year.count + 2)
test.range <- test.sim.start:test.sim.end
obs <- precipOcc[test.range, ]
test.vgram <- getAggregateVariogram(obs, 15)
dists <- test.vgram[[2]]
vgram.len <- length(dists)
# Look at box plots for individual month variograms
trial.count <- 20
sim.dfs <- {}
obs.dfs <- {}
gs <- {}
for(month in 0:11) {
  index <- month + 1  # Used as index for lists
  gp.start <-  1 + month*30 # Add 1 so that we are not accessing
     0 for list index
  sim.vgrams <- {}
  obs.vgram <- getAggregateVariogram(obs, gp.start)[[1]]
  for(i in 1:trial.count) {
    cat("trial = ", i, '\n')
    gp <- maskWithNA(obs,
                     simulateSinusoidalExponentialGP(cov.theta,
                        test.sim.start,
                                                     test.sim.end
                                                        ))
    sim.vgrams[[i]] <- getAggregateVariogram(gp, gp.start)[[1]]
  }
  vgram.vals <- {}
  for(i in 1:vgram.len) {
    vgram.vec <- {}
    for(j in 1:trial.count) {
      vgram.vec[j] <- sim.vgrams[[j]][i]
    }
    vgram.vals <- c(vgram.vals, vgram.vec)
  }
  dist.vec <- {}
  for(i in 1:vgram.len) {
    dist.vec <- c(dist.vec, rep(dists[i], trial.count))
  }
  # Box plot of means for all samples
```

```
  sim.dfs[[index]]  <- data.frame(Value=vgram.vals, Dist=dist.vec
     )
  obs.dfs[[index]]  <- data.frame(Value=obs.vgram, Dist=dists)
  index <- index + 1
}

month.names <- c("January", "February", "March", "April", "May",
   "June", "July", "August", "September", "October", "November",
   "December")
vgram.plots <- {}
pdf("variograms.pdf", height=10, width=8)
for(i in 1:12) {
  sim.df <- sim.dfs[[i]]
  obs.df <- obs.dfs[[i]]
  vgram.plots[[i]] <- ggplot() + geom_boxplot(d=sim.df, aes(x=
     factor(Dist), y=Value),
                                              outlier.shape=NA) +
             geom_line(d=obs.df,
                       aes(x=seq_along(Dist), y=Value), colour="
                          blue") +
             labs(x='Distance', y='Variogram') + ggtitle(month.
                names[i]) +
             theme(axis.text.x=element_text(angle=-45, hjust
                =-0.25, vjust=1.5, size=6)) +
             ylim(0, max(c(obs.df$Val, sim.df$Val), na.rm=T))
}
multiplot(vgram.plots, cols=3)
dev.off()

titles <- c(expression(alpha[0]), expression(alpha[1]),
   expression(alpha[2]),
            expression(tau))
plots <- list()
pdf("alphadensities.pdf", width=8, height=6)
for(i in 1:4) {
  values <- accepted[i, 200:1025]
  xmin <- unif.left[i]
  xmax <- unif.right[i]
  width = xmax - xmin
  height <- 1/(width)
  d <- data.frame(r)
  plots[[i]] <- ggplot() +
    geom_density(d=data.frame(Vals=values), aes(Vals, fill="ABC
       posterior"),
                 alpha=0.4) +
    scale_x_continuous(limits=c(xmin - width/20, xmax + width/20)
       , name="") +
```

```
    scale_y_continuous(name="") +
    geom_rect(xmin=xmin, xmax=xmax, ymin=0, ymax=height, colour="
       black",
        aes(fill="Prior"), alpha=0.4) +
    ggtitle(titles[i]) + theme(legend.title=element_blank())
}
multiplot(plots, col=2)
dev.off()

gp <- maskWithNA(obs,
                 simulateSinusoidalExponentialGP(abc.theta, test.
                    sim.start,
                                                 test.sim.end
                                                    ))

# Quilt plot
pdf("precip_plots.pdf", width=8, height=10)
par(mfrow=c(4, 3))
for(i in 10:21) {
  day <- 29940 + i
  title <- paste(mo[day], da[day], sep='/')
  quilt.plot(lon.lat, gp[i, ], col=c("orange", "blue"), main=
     title, add.legend=FALSE)
  legend("topright", legend=c("0", "1"), col=c("orange", "blue"),
      pch=c(15, 15))
  US(add=TRUE)
}
dev.off()

# Look at spell counts
taus <- 1:10
sim.counts <- obs.counts <- {}
for(tau in taus) {
  s.counts <- o.counts <- 0
  for(i in 1:(dim(gp)[2])) {
    o.counts <- o.counts + getSpellCount(obs[, i], tau, tau + 1,
       TRUE)
    s.counts <- s.counts + getSpellCount(gp[, i], tau, tau + 1,
       TRUE)
  }
  sim.counts[tau] <- s.counts
  obs.counts[tau] <- o.counts
}

vs <- vs2 <- {}

for(month in 1:12) {
```

```
  print("Yo")
  gp.start <- 30*month
  occ.start <- 30*month # Shift by the given month

  v1 <- getAggregateVariogram(gp, gp.start)
  v2 <- getAggregateVariogram(obs, occ.start)
  if(sum(v2[[1]]) != 0) {
    month.error <- sum(abs(v1[[1]] - v2[[1]])/abs(v2[[1]]))
    print(month.error)
  }
  vs[[month]] <- v2[[1]]
  vs2[[month]] <- v1[[1]]
}
cols <- rainbow(12)
par(mfrow=c(1, 2), mar=c(0, 0, 0, 0))
plot(vs[[1]], ylim=c(0, 0.20), main='Obs', col=cols[1]);
for(i in 2:12) { lines(vs[[i]], col=cols[i]) }
legend('bottomright', legend=c("Jan","Feb", "Mar", "Apr", "May",
   "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"),
       col=cols, lty=c(1,1))
plot(vs2[[1]], ylim=c(0, 0.20), main='Sim', col=cols[1]); for(i
   in 2:12) { lines(vs2[[i]], col=cols[i]) }
legend('bottomright', legend=c("Jan","Feb", "Mar", "Apr", "May",
   "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"),
       col=cols, lty=c(1,1))

par(mfrow=c(3, 4))
for(i in 1:12) {
  v1.temp <- data.frame(getAggregateVariogram(gp[mo %in% i]))
  v2.temp <- data.frame(getAggregateVariogram(precipOcc, precip.
     occ.start))
  plot(v1.temp[,1]); lines(v2.temp[,1])
}

vs <- {}
for(i in 1:12) {
  vs[i] <- data.frame(getAggregateVariogram(precipOcc[mo %in% i],
      precip.occ.start))
}
```