INVESTIGATION OF NON-HOMOGENEOUS HIDDEN MARKOV MODELS AND THEIR APPLICATION TO SPATIALLY-DISTRIBUTED PRECIPITATION TYPES

A Thesis

by

JAE YOUNG SONG

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

December 2012

Major Subject: Civil Engineering

ABSTRACT

Precipitation is an important element in the hydrological cycle. To predict and simulate large-scale precipitation, Global Circulation Models (GCMs) are widely used. However, their grid scale is too big to apply to local hydrologic fields. In this study, non-homogenous hidden Markov models (NHMM) are explored as a means of generating the probability of precipitation occurrence in small scale given large-scaled weather patterns. Three different spatial models: (1) independent (2) auto-logistic (3) Chow-Liu tree, are also explored, along with methods and steps for parameter estimation. From this exploration, independent models with NHMM are recommended for very small precipitation networks, and the maximum likelihood method is found to be the most practical fitting method. If there are many points for downscaling, Chow-Liu tree models with the Expectation-Maximization (EM) algorithm are recommended. If more exact solutions are needed, auto-logistic models can be employed. If many points are considered in auto-logistic models, the (EM) algorithm should be used to estimate parameters separately and global optimization methods should be used for emission matrix. The major problem found with the NHMM model in this study is matching the rainfall amount for each year or month. This problem can be addressed by whether combining occurrence models with amount modes or by improving only occurrence models.

DEDICATION

Thanks to my family for supporting me and their patience. In financial difficulty and distress, I really appreciate to my father and mother who always allow my every single decision without any dissatisfaction.

ACKNOWLEDGEMENTS

First, thanks to my advisor, Dr. Cahill who helps me to be able to conduct this thesis work, and to my committee member, Dr. Brumbelow and Dr. Mohanty. Also, thanks to Dr. Hughes who helped me with understanding his paper.

Thanks to all my friends and the department faculty and staff. In my first studying a
broad, I had great experiences and met good people at Texas A&M University.

Again, thanks to my family for their support and encouragement.

NOMENCLATURE

S	Hidden State
T	Entire Time
t	Time
R	Rainfall Occurrence
X	Weather Pattern
r	Rainfall Occurrence Data
x	Weather Pattern data

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Precipitation is an important element in the hydrological cycle, and has an obvious impact on water supply, flood prevention, ecosystem, contaminant management, etc. Given the potential for global climate change to cause nonstationarity in precipitation patterns, many researchers have tried to capture the phenomenon of the climate change in precipitation models, to improve future predictions. With current General Circulation Models (GCMs) many key elements of climate phenomenon are well understood and modeled. However, although the GCM has a good ability to do climate prediction, and has been validated by various researchers, a GCM's grid scale is too big to apply to the local hydrologic fields such as precipitation. In this study, we will focus on Nonhomogeneous Hidden Markov Model (NHMM) as one prediction model, also known as a downscaling method, which generates probability of precipitation occurrence given large-scale weather patterns. Recently, the NHMM has been widely employed and precipitation occurrence results have been well simulated [*Khalil et al.*, 2010; *Kwon et al.*, 2011; *Robertson et al.*, 2003; 2004]. Also such downscaling methods have become more important in hydrological investigations because of the increase in use of image data sucah as remotely sensed precipitation fields. The purpose of this research is to explore the NHMM model and several commonly used spatial model types for the NHMM: Independent model, Auto-Logistic model, and Bayesian Tree Structure model (Chow-Liu Tree). In this study, we will present how NHMM is formulated, how it works and how it can be verified. Additionally, through comparing the spatial types in NHMM, we will explore how well the downscaling models give good predictions for the future precipitation occurrences and also precipitation amounts.

1.1 Global Circulation Models (GCMs)

GCMs data have a key role in our downscaling models, because predictions of future rainfall occurrence in NHMM are conditional on climate prediction data from GCMs. In GCMs, the atmosphere is divided vertically and horizontally and the atmospheric values are estimated by integrating a system of partial differential equations (the Navier-Stokes equations applied to the atmosphere) from particular initial conditions. In the models, predictions of potential climate change are estimated by changing the levels of important inputs such as CO_2 and other greenhouse gas emissions. GCMs typically have a "run up period", in which they are trained with several decades of measured input parameters representing the present day's climate conditions. In this training process, the control runs are compared with historical data to improve the models validity. Then, the model is rerun and estimates future climates according to the various experimental scenarios. In global scale, the GCMs give reasonable results for current climate conditions. However, the results of GCMs cannot be interpreted for scales less than thousands of kilometers. This is because regional or local scale phenomenon given broad scale areal average can be related with a diversity of different regional or local patterns [*Hughes and Guttorp*, 1994]. For example, precipitation models are usually concerned with small scale or sub-catchment scale, which are much smaller than the scale of GCMs. GCMs handle fluid dynamics at the continental scale, parameterizing regional and local scale processes. Therefore, they naturally have the most uncertainties in the climate model resulting in amplifying scale-related sensitivities and scale-incongruity problems [*Wilby and Wigley*, 1997].

1.2 Downscaling

These incongruous scales gave rise to “downscaling” approaches, which can be classified in three ways: 1) empirical approaches using the paleoclimatic record; 2) modeling approaches by physically improving GCM’s resolution; 3) semi-empirical approaches using stochastic models reflecting observed data [*Hughes and Guttorp*, 1994].

To capture regional effects within climate change, paleoclimatic analogues have been inspected and three periods related with global warming have been identified [*Flohn*, 1979; *Flohn and Fantechi*, 1984]. However, the periods cannot be regarded as identical referents for present and future climate. This is because circulation patterns could be different according to the distribution of some elements such as vegetation, sea ice and sea surface temperature between each era [*Crowley*, 1990]. Another problem is that rate of the greenhouse effect resulting in global temperature change is different from time to time [*Giorgi and Mearns*, 1991]. From this reason, the empirical method has limitations in assisting regional effects.

The second approach is to improve the GCM resolution directly. This is a plausible means of improvement, but the problem of computation time and uncertainty of the result in higher resolution model has always occurred. Moreover, many unexplainable and complex factors exacerbate uncertainties in higher resolution models [*Hughes and Guttorp*, 1994]. Currently, because computer technology and developing circulation models have been better, higher resolution is actually available to aid regional circulation models (RCMs). However, these results of RCMs still cannot give resolution satisfactory for capturing local climate pattern. Even if improved grid scale is available, downscaling techniques should be needed as long as a resolution of the circulation models cannot reach to adequately small scale required for a particular interest.

Third, a stochastic or statistic approach has been widely applied because it is rational and convenient to explain complex phenomenon. The appropriate stochastic method to use is still a matter of research; the choice can be categorized in two ways: 1) a purely statistical or purely stochastical way; or 2) a semi-statistical or semi-stochastic way. There have been many efforts to downscale GCM's results capture local precipitation patterns using purely statistic or stochastic. For example, [*Gabriel and Neumann*, 1962] introduced a Markov chain model to generate daily rainfall occurrence at Tel Aviv in Israel. Their assumption was the first-order Markov assumption:

$$\mathrm{P}(\mathrm{R_t}|\mathrm{R_{t-1}},\mathrm{R_{t-2}},\ldots,\mathrm{R_1}) = \mathrm{P}(\mathrm{R_t}|\mathrm{R_{t-1}}) \tag{1.2.1}$$

where $\mathrm{R_t}$ is a rainfall random process at time t. In their model, the assumption is that the probability of $\mathrm{R_t}$ only depends on a previous random process $\mathrm{R_{t-1}}$, and consists of wet/dry states. The transition matrix $\mathrm{A(t)}$ reflecting Markov chain has the form

$$\mathrm{A(t)} = \begin{bmatrix} \mathrm{P_{dry}(t)} & 1-\mathrm{P_{dry}(t)} \\ 1-\mathrm{P_{wet}(t)} & \mathrm{P_{wet}(t)} \end{bmatrix} \tag{1.2.2}$$

where $\mathrm{P_{dry}(t)}$ is probability of dry weather occurrence at time t depended on $\mathrm{P_{dry}(t-1)}$ and $\mathrm{P_{wet}(t)}$ is probability of wet weather occurrence at time t depended on $\mathrm{P_{wet}(t-1)}$. It is obvious that the model is over-parameterized because every time has to be considered. If we set $\mathrm{P_{dry}(t)}$ and $\mathrm{P_{wet}(t)}$ to $\mathrm{P_{dry}}$ and $\mathrm{P_{wet}}$, they become a possible way for practical use, and are defined as homogeneous or stationary models. However, this assumption is only useful for short time period like monthly or seasonal [*Hughes and Guttorp*, 1994]. The other approach is to assume the $P_{dry}(t)$ and $P_{wet}(t)$ are following Fourier series by [*Stern and Coe*, 1984]. Although the assumption and its results show good fitting, it is not only fail to reproduce statistically hydrological distribution of interest, but it is also unsuccessful to fit their model for multiple stations.

These attempts, purely statistical or purely stochastical models, cannot succeed at “downscaling” because referential atmospheric circulation patterns are not reflected in these models. In other words, if we want to downscale GCMs data, the downscaling model should apply the GCMs data to rationally generate local precipitation. From this reason, many theories of downscaling have been proposed and they can be categorized in two ways: 1) regression methods; 2) weather-pattern based.

Regression methods can be regard as a simplest way using a linear or non-linear relationship between GCMs data and local precipitation data. A more complex model is “expanded downscaling”, which uses covariance between GCMs and local precipitation patterns to generate the average and short-term variability between two scales [*Bürger*, 1996]. Also, artificial neural networks (ANN) have been used; these are categorized as regression methods because ANN also consists of nonlinear regression models [*Hewitson and Crane*, 1992a; b; 1996]. In spatial scaling, these approaches work by regressing the same parameters from a regional to local scale. From this reason, although the log-moments of these models for precipitation models show linearity, a more complex process has to be considered to get more reliability for those spatial scaling [*Wilby and Wigley*, 1997].

The second way is a downscaling method based on stochastically capturing the patterns between observed station and weather information given from GCMs. The weather information, also known as weather classification, is objectively or automatically defined by methods such as principal components, canonical correlation analysis, fuzzy rules, compositing, neural networks, correlation-based pattern recognition techniques and analogue procedures. In this category, one of the representative models is the hidden Markov model (HMM) and it is based on an assumption that the rainfall patterns have binary process such as rainfall occurring or not occurring. However, because HMM does not use large scale data as an input variable when generating small scale simulation, this model is not structured enough to use for nonstationary precipitation process. To overcome this problem, Nonhomogeneous hidden Markov model (NHMM) have been introduced and the number of these studies have been conducted using simple assumptions by [*Bardossy and Plate*, 1992; *Hay et al.*, 1991; *Hughes et al.*, 1993; *Zucchini and Guttorp*, 1991], also see [Table 1].

TABLE 1 : HMM AND NHMM MODELS [*Hughes and Guttorp*, 1994]

HMM and NHMM	$P(X_t)$	$P(S_t \mid S_{t-1}, X_t)$	$P(R_t \mid S_t)$
Markov Chain		$P(S_t \mid S_{t-1})$	$P(R_t \mid S_t) = I$
Zucchini and Guttorp [1991]		$P(S_t \mid S_{t-1})$	$P(R_t \mid S_t)$
Hay et al. 1991	Semi-Markov	$P(S_t \mid X_t) = I$	$P(R_t \mid S_t)$
Bardossy and Plate 1992		$P(S_t \mid X_t) = I$	$P(R_t \mid R_{t-1}, S_t)$
Hughes et al. 1993	Mixture transition distribution model	$P(S_t \mid X_t) = I$	$P(R_t \mid S_t)$

X_t : Weather data or Weather synoptic data
R_t : Rainfall data
S_t : Hidden State
I : set $S_t = X_t$ or $S_t = R_t$

The major advantage of this model type is that they are built based on sensible physical associations between global scale and local scale. From this reason, this method has been regarded as one of the most promising approaches and has been applied by many authors in climate research based on statistical and physical linkages to demonstrate temporal circulation changes [*Wilby and Wigley*, 1997]. Another common method is the stochastic weather generator known as WGEN model, introduced by [*Richardson and Wright*, 1984]. This model is used for climate impacts of particular interest: precipitation amount, solar radiation, minimum and maximum temperature. However, in the WGEN model all values are generated conditional on precipitation occurrence, so this model is out of scope for our interest.

1.3 Non-homogeneous Hidden Markov Model (NHMM)

The NHMM is categorized by the assumed existence of a finite, discrete-valued, hidden "weather state" process which follows a nonhomogeneous Markov chain [*Hughes et al.*, 1999]. The basic Markov chain model (MC) consists of a transition matrix which explains the probability of the state change. For example, there are 5 days precipitation occurrence data and if we assume the state means precipitation occurrence, we can simply calculate the first-order probability of the rainfall state change $P(R_t|R_{t-1})$. See [Table 1], [Table 2] and [Table 3].

TABLE 2 : RAINFALL OCCURRENCE DATA

Day	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Rainfall	Occur	Not Occur	Occur	Occur	Not Occur	Not Occur	Occur

TABLE 3 : COUNT THE NUMBER OF STATE CHANGE

Number of State Change	**Occur(To)**	**Not Occur(To)**
Occur(From)	1	2
Not Occur(From)	2	1

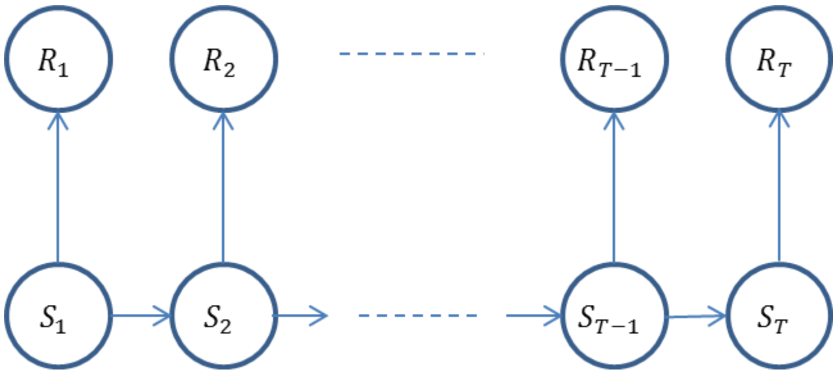
TABLE 4 : TRANSITION MATRIX

Probability of State Change	Occur(To)	Not Occur(To)
Occur(From)	1/6	2/6
Not Occur(From)	2/6	1/6

If we want to know the probability of that the first rainfall occurs on 5th day, it can be calculated as

$$P(NNNNO|\theta) = \delta * \left(\frac{1}{6}\right)^3 * \frac{2}{6}, \qquad \text{where } \delta \text{ is initial value}$$

FIGURE 1 : HMM MODELS



Usually this is called the transition probability distribution or transition matrix in Hidden Markov model (HMM) which relies on the previous state in time series. In HMM the transition matrix becomes hidden and, after transition matrix process is made, we need to one more parameter, called the emission matrix for the HMM; it produces an observation output depending on the current state [Figure 1]. Therefore, The HMM is called a doubly stochastic process within a hidden stochastic process. Formally, we can write

- $P(S_t = j|S_{t-1} = i) = A_{ij}$ Probability of State Change
- $P(R_t = k|S_t = j) = B_{kj}$ Probability of Observation Sequence given hidden State

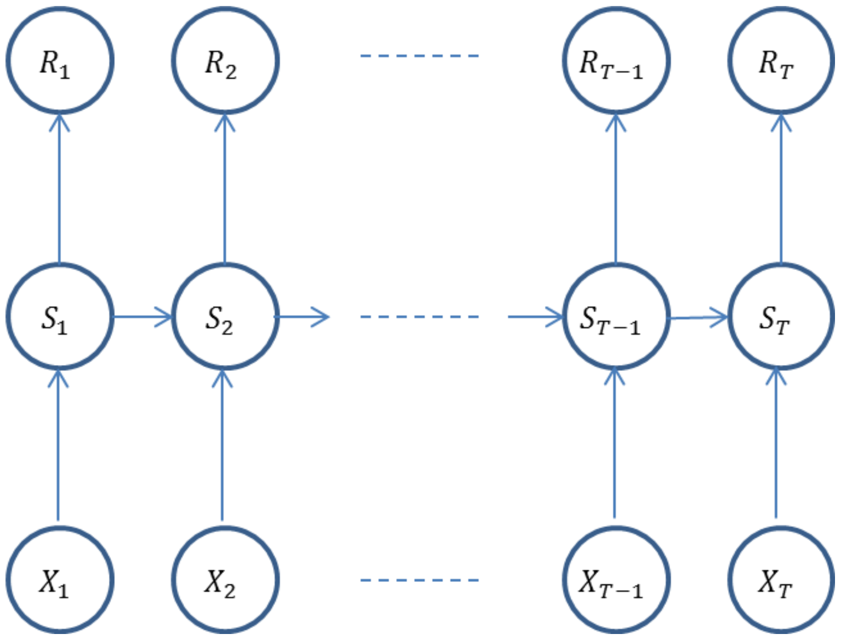
where R_t is rainfall occurrence at time t and S_t is hidden weather state at time t. Originally, this theory had been applied various fields and hydrologic studies for downscaling GCM data since it was introduced for a speech recognition by [*Rabiner and Juang*, 1986]. In hydrologic fields, there have been various approaches using the HMM to make classification of synoptic atmospheric information into small number of "hidden states" in the model, see [Table 1]. The Nonhomogeneous Hidden Markov Model was introduced by [*Hughes et al.*, 1999] to reflect varying synoptic atmospheric processes in the model, especially for GCM. In short, the difference between the HMM and his model (the NHMM) is in the transition matrix, where the transition matrix in HMM is the probability of hidden state changes without given weather data in time series. [*Hughes and Guttorp*, 1994] assumes

- R_t = measurement of the local process at time t
- S_t = weather state at time t (unobserved)
- X_t = measurement of the atmospheric data at time t

$$(1)\quad P(R_t|S_1^T, R_1^{t-1}, X_1^T) = P(R_t|S_t) \tag{1.3.1}$$

$$(2)\quad P(S_t|S_1^{t-1}, X_1^T) = P(S_t|S_{t-1}, X_t) = \frac{P(X_t|S_1^T)P(S_t|S_{t-1})}{\sum_{S_t} P(X_t|S_1^T)P(S_t|S_{t-1})} \tag{1.3.2}$$

FIGURE 2 : NHMM MODELS



Usually, the transition matrix and emission matrix have simple forms as $A(X_t)$ and $B(S_t)$. The transition matrix $A(X_t)$ refers to probability of state change given large-scale weather data X_t. The emission matrix $B(S_t)$ is probability of rainfall occurrence given a current state S_t [Figure 2]. Using this basic structure, various researches and applications have been conducted. Detailed information and usage are explained at [*Khalil et al.*, 2010; *Robertson et al.*, 2003; 2004].

1.4 Multivariate Spatial Model

The spatial model type is important in the choice of downscaling method, because it affects weather type classification (the hidden state) and its simulation results. The task is simple if we are looking at just one gauge station. However, for practical purposes, a multisite model should be considered. Many spatial model types exist but selection of the spatial model types in NHMM should be practical and applicable for both parameter estimation and simulation. This is because more complex spatial models make NHMM much more difficult to control or impractical, even impossible to estimate parameters. The simplest and most broadly used model is an independent model for each gauge station, which also called a conditional independent model in NHMM. Though the model seems obviously independent because there are no parameters explaining spatial dependency between gauge stations, each group of point locations have common hidden states. From this reason, if too many spatial points with many diverse rainfall patterns are considered, the number of hidden state should be increased to capture the patterns in NHMM. In the independent models, the number of parameters becomes $[\text{Number of State}(\mathrm{M}) * \text{Number of Station}(\mathrm{N})]$ and the equation has the form:

$$\mathrm{B}(\mathrm{S_t}) = \mathrm{P}(\mathrm{R_t}|\mathrm{S_t}) = \mathrm{p}_s{}^{\mathrm{r_t}}(1 - \mathrm{p}_s)^{1-\mathrm{r_t}} \tag{1.4.1}$$

where B is emission matrix, $\mathrm{S_t}$ is a state at time t, $\mathrm{R_t}$ is rainfall occurrence at time t, p_s is probability of rainfall occurrence conditional on state s, and $\mathrm{r_t}$ is rainfall occurrence of data consisting of 0 and 1 at time t.

Dependent models always have many parameters to explain every all (first and higher-order) spatial dependency. Therefore, an optimal spatial model is needed to reduce the number of parameters and to improve the model efficiency. Past research has use a number of approaches as candidate methods; for example, auto-logistic model [*Hughes and Guttorp*, 1994], and the Chow-Liu Tree model [*Kwon et al.*, 2011] etc.

The auto-logistic model was first introduced by Besag [*Besag*, 1974; 1975] and is commonly used for spatially-dependent models. In this study, the model has a restriction of only considering first and second order. In the binary data case, parameters are estimated using the logistic regression model by natural choice and its equation in NHMM has a form of

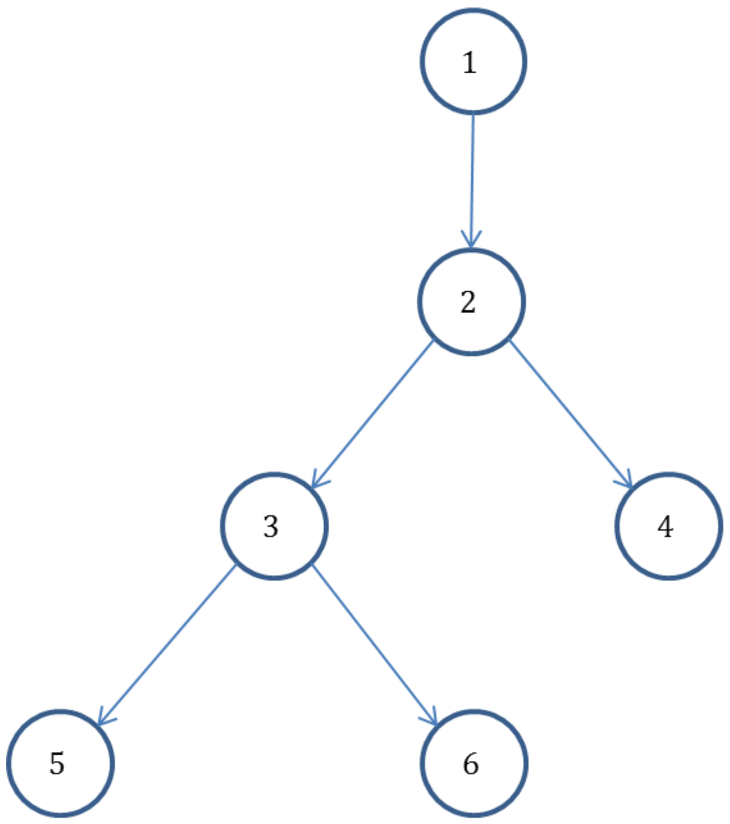
$$P(R_i|R_{j\neq i}) = \frac{\exp(\alpha_i r_i + \sum_{j\neq i} \beta_{ij} r_i r_j)}{1 + \exp(\alpha_i + \sum_{j\neq i} \beta_{ij} r_j)} \quad (1.4.2)$$

where i and j are locations of gauge stations, α is a parameter for rainfall occurrence at station i, and β are parameters of spatial dependency between station i and j.

If β_{ijs} in the auto-logistic model is set to zero, the model becomes an independent model, in which way is often used to give initial point for huge dependent models. This model has $MN(N-1)/2$ parameters. If β_{ijs} is positive value, the two sites are positively correlated, otherwise they are negative correlated [*Hughes and Guttorp*, 1994].

Another method for dependent spatial model is the Chow-Liu Tree (CLT) [*Chow and Liu*, 1968], based on Bayesian networks to approximate the discrete joint distribution, which consists of products of distributions containing no more than pairs of variables [Figure 3]. To maximize its likelihood, a Maximum Weight Spanning Tree (MWST) is employed using Mutual Information (MI). (This optimization method will be discussed later.) More detailed information and improved methods are explained at [*Kirshner et al.*, 2004], which shows how to use the CLT and introduces Chow-Liu Forest Tree structure (CLFT) in HMM.

FIGURE 3 : $P(x_1)P(x_2|x_1)P(x_3|x_2)P(x_4|x_2)P(x_5|x_3)P(x_6|x_3)$



2. NHMM DOWNSCALING MODEL

2.1 Weather Model

The most complex parameter in the NHMM is the transition matrix, due to the high dimensional atmospheric data. Hughes et al. [*Hughes et al.*, 1999] introduced the transition model with the assumption it is multivariate normal:

$$\begin{aligned} A(X_t) &= P(S_t \mid S_1^{t-1}, X_1^T) \\ &\propto \gamma_{ij} \exp\left(-0.5\left(X_t - \mu_{ij}\right)\Sigma^{-1}\left(X_t - \mu_{ij}\right)'\right) \end{aligned} \tag{2.1.1}$$

where ij denotes a state change from i to j, μ_{ij} is the average of X_t, conditional on S_{t-1} and S_t, Σ^{-1} is the covariance matrix of X_t, and γ_{ij} is original transition matrix. In a special case, this model becomes a simple normal distribution model of X_t corresponding state change when X_t is a single-variable time series. Moreover, if we assume $\sum_j \mu_{ij} = 0$ and $\sum_j \gamma_{ij} = 1$, the transition model can become a logistic model, which makes parameter estimation significantly more convenient due to the reduction of the number of parameters [*Hughes et al.*, 1999].

$$A(X_t) = P(S_t \mid S_1^{t-1}, X_1^T) \propto \exp\left(a_{ij} + b_{ij}x_t\right) \tag{2.1.2}$$

where, $b_{ij} = \mu_{ij}\Sigma^{-1}$ and $a_{ij} = \ln\gamma_{ij} - 0.5\mu_{ij}\Sigma^{-1}\mu_{ij}'$. If input data x_t are normally distributed, multivariate normal distributions are recommended. If it is not, logistic models are recommended.

2.2 Precipitation Model

2.2.1 Independent Model

For considering one gauge station, the equation is the same as equation (1.4.1). In a multisite case, the model or the joint distribution has a product form of all sites N.

$$B(S_t) = P(R_t|S_t) = \prod_{i=1}^{N} {p_{is}}^{r_{it}}(1 - p_{is})^{1-r_{it}} \tag{2.2.1}$$

2.2.2 Auto-logistic Model

The auto-logistic model can be used for both the spatially-independent model and dependent model. The spatial dependent model has the form:

$$B(S_t) = P(R_t|S_t) \propto \exp\left(\sum_{i=1}^{n} \alpha_{is} r_i + \sum\nolimits_{j<i} \beta_{ijs} r_i r_j\right) \tag{2.2.1}$$

As explained above, each term of α_{is} is an independent parameter of the model and another term of β_{ijs} explains spatial dependency of each pair. If we want to use independent model in the auto-logistic model, the β_{ijs} term is set to zero:

$$B(S_t) = P(R_t|S_t) \propto \exp\left(\sum_{i=1}^{n} \alpha_{is} r_i\right) \tag{2.2.2}$$

2.2.3 Chow-Liu Tree Model

The Chow-Liu Tree has own method for parameter estimation, which differs significantly from that of the other models presented above. The algorithm provided by [*Chow and Liu*, 1968]is as follows:

TABLE 5 : CHOW-LIU TREE ALGORITHM [*KIRSHNER ET AL.*, 2004]

1. Compute all $P(r_i)$ and all pair $P(r_i, r_j)$
2. Compute all Mutual Information $I(r_i, r_j)$
3. Construct Maximum Weight Spanning Tree using all $I(r_i, r_j)$
4. $E =$ Select $P(r_i, r_j)$ in Maximum Weight Spanning Tree

This optimization process promises to minimize the Kullback-Leibler divergence. The distribution of tree structure is

$$B(S_t) = P(R_t|S_t) = P_s(r_1)\frac{\prod_{(i,j)\in E} P_s(r_i, r_j)}{\prod_{j\in J} P_s(r_j)} \qquad (2.2.1)$$

and the mutual information $I(r_i, r_j)$ is

$$I(r_i, r_j) = \sum_{r_i}\sum_{r_j} P(r_i, r_j)\log\left(\frac{P(r_i, r_j)}{P(r_i)P(r_j)}\right) \qquad (2.2.2)$$

The Maximum Weight Spanning Tree algorithm is conducted continuing to select highest pair value of the mutual information as long as it does not make a circle formed with previously selected branches. The number of branches or pairs is always $N(N-1)/2$ [*Chow and Liu*, 1968].

2.3 Model Selection

In the NHMM or HMM, there are not perfect methods to give exact how many hidden states have to be considered. In this case, Bayes Information Criterion (BIC) is used as a usual manner.

$$\mathrm{BIC} = 2\ell - vlogn \tag{2.3.1}$$

where ℓ is log-likelihood, v is the number of free parameters, and n is the sample size. This criterion cannot be an absolute manner for NHMM, but it gives good insight for the selection of the number of hidden states.

3. PARAMETER ESTIMATION

In using the HMM or NHMM model, the most difficult aspect is the parameter estimation, because these models are generally complex stochastic processes and have many parameters. To estimate the parameters, the maximum likelihood method and expectation-maximization (EM) method were both tested. According to [*Hughes et al.*, 1999] and[*Rabiner and Juang*, 1986], in past work the EM algorithm has been mainly used. The problem with this method is that, because EM algorithm basically was derived with a convexity assumption, it tends to find local optimal values and strongly depends on initial values. Therefore, an another optimization method is also needed; an initial run of iteration of maximum likelihood estimation is often useful to get close to the global optimal values before using EM algorithm[*Hughes et al.*, 1999]. To estimate the parameters several assumption are needed to reduce computational problems and to avoid over-parameterization. For multivariate normal distributions we have two assumptions.

- Assumption 1 : $\sum_j \mu_{ij} = 0$
- Assumption 2 : $\sum_j \gamma_{ij} = 1$

With logistic models, there are no certain assumptions, but the usual approach is

- Assumption 1 : $\sum_j a_{ij} = 0$
- Assumption 2 : $\sum_j b_{ij} = 0$

3.1 Derivation of Maximum Likelihood

The likelihood of NHMM can be expressed with matrix form

$$L(\theta) = \delta(x_1)B(r_1)A(x_2) \dots A(x_T)B(r_T)$$

where $\mathrm{A}(\mathrm{x}_\mathrm{t})$ is transition matrix, $B(r_t)$ is joint probability distribution given current state and $\delta(\mathrm{x}_1)$ is an initial hidden state. This expression cannot only make the numerical errors but it can also be computationally expensive for other purposes such as EM algorithm or first order differentiation, because every combination of the hidden states should be considered through the time steps. Therefore, the forward-backward procedure has to be used.

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) A_{ij}(x_{t+1})\right] B(r_{t+1}) \tag{3.1.1}$$

$$\beta_t(i) = \sum_{i=1}^{N} A_{ij}(x_{t+1}) B(r_{t+1}) \beta_{t+1}(j) \tag{3.1.2}$$

where α and β are the forward-backward procedure parameters. Choosing different optimization methods also can affect a computation time. Using some of the methods with gradients of the parameters can show great improvement. The forms of the gradients are

For $\mathbf{P}(\mathbf{S_t}|\mathbf{S_{t-1}}, \mathbf{X_t})$

$$\begin{aligned}\frac{\mathrm{dL}}{\mathrm{d}\theta} &= \delta(\mathrm{x}_1) * \frac{\mathrm{d}\big(\mathrm{B}(\mathrm{r}_1) \ldots \mathrm{A}(\mathrm{x}_\mathrm{T})\mathrm{B}(\mathrm{r}_\mathrm{t})\big)}{\mathrm{d}\theta} \\ &\qquad + \frac{\delta(\mathrm{x}_1)}{\mathrm{d}\theta}\big(\mathrm{B}(\mathrm{r}_1) \ldots \mathrm{A}(\mathrm{x}_\mathrm{T})\mathrm{B}(\mathrm{r}_\mathrm{t})\big) \qquad (3.1.1) \\ &= \big(\mathrm{I} - \mathrm{A}(\mathrm{x}_1)\big)^{-1} * \frac{\mathrm{dA}(\mathrm{x}_1)}{\mathrm{d}\theta}\alpha_\mathrm{T} + \sum_{\mathrm{t}=2}^{\mathrm{T}} \alpha_{\mathrm{t}-1}\frac{\mathrm{dA}(\mathrm{x}_\mathrm{t})}{\mathrm{d}\theta}\mathrm{B}(\mathrm{r}_\mathrm{t})\beta_\mathrm{t}\end{aligned}$$

Continue for $\mathbf{P(S_t|S_{t-1},X_t)}$

$$\begin{aligned}A(X_t) = P(S_t|S_{t-1},X_t) &= \frac{P(X_t|S_1^T)P(S_t|S_{t-1})}{\sum_{S_t} P(X_t|S_1^T)P(S_t|S_{t-1})} \\ &= \frac{\exp(a_{ij}+b_{ij}x_t)}{\sum_{s_t}\exp(a_{ij}+b_{ij}x_t)}\end{aligned} \tag{3.1.2}$$

where θ refers to all parameters in transition matrix $A(X_t)$, $b_{ij} = \mu_{ij}\Sigma^{-1}$, $a_{ij} = ln\gamma_{ij} - 0.5\mu_{ij}\Sigma^{-1}\mu_{ij}'$ and where α and β are the forward-backward procedure

For $\boldsymbol{P(R_t|S_t)}$

$$\frac{dL}{d\vartheta} = \delta(x_1) * \frac{dB(r_t)}{d\vartheta}\beta_1 + \sum_{t=2}^{T} \alpha_{t-1}A(x_t) * \frac{dB(r_t)}{d\vartheta}\beta_t \tag{3.1.3}$$

where ϑ refers to all parameters in emission matrix $B(r_t)$, and where α and β are the forward-backward procedure.

3.2 Expectation-maximization (EM algorithm)

This algorithm was the original method used to estimate parameters for the HMM which has unknown data like a hidden state. It creates updated parameters, θ', from the current parameters, θ, while at every iteration maximizing an auxiliary function $\varphi(\theta'|\theta)$. The EM algorithm is explained in detail at [*Dempster et al.*, 1977], and general EM principles, including the iterative scheme, are discussed at [*Baum et al.*, 1970]. From [*Baum* *et al.*, 1970], increase of the likelihood is guaranteed in each step of the iteration. In NHMM, the EM algorithm has the form of

$$\begin{aligned} Maximize\ \varphi(\theta'|\theta) &= \sum_S \boldsymbol{P}\left(\boldsymbol{S}\middle|\boldsymbol{r}_1^T,\boldsymbol{\theta},\boldsymbol{X}_1^T\right) lnP(r_1^T,S|\theta',X_1^T) \\ &= \sum_S \boldsymbol{P}\left(\boldsymbol{S}\middle|\boldsymbol{r}_1^T,\boldsymbol{\theta},\boldsymbol{X}_1^T\right)\left[\sum_{t=1}^T lnB(r_t,s_t)\right. \\ &\qquad + ln\delta(x_1,s_1) \\ &\qquad \left. + \sum_{t=1}^T lnA(x_t,s_t|s_{t-1})\right] \end{aligned} \qquad (3.2.1)$$

The auxiliary function means the expectation of $lnP(r_1^T,S|\theta',X_1^T)$ about distribution $\boldsymbol{P}\left(\boldsymbol{S}\middle|\boldsymbol{r}_1^T,\boldsymbol{\theta},\boldsymbol{X}_1^T\right)$. The parameters' solution θ could be found by maximum likelihood $lnP(r_1^T,S|\theta',X_1^T)$ for the complete data. However, it has unknown data (hidden state), so we can maximize its expectation, using observed data and current parameters. In other words, in E-step, $P(S|r_1^T,\theta,X_1^T)$ is estimated by current parameters, θ. In M-step, parameters θ' in $B(r_t,s_t)$ and $A(x_t,s_t|s_{t-1})$ are earned by maximizing $\varphi(\theta'|\theta)$. Then, the updated parameters would be used for the next iteration until meeting a convergence of its likelihood. Additionally, in [Hughes et al., 1999], if $c_i > 0, i = 1,2,3,..,n$ and $\sum_i x_i = k$, the function will be

$$f(x) = \sum_i c_i\, ln(x_i) \qquad (3.2.2)$$

Then, the unique optimal values can be found from

$$x_i = \frac{kc_i}{\sum_i c_i} \tag{3.2.3}$$

In HMM and NHMM, there are constraints in the transition matrix and emission matrix having a form of $\sum_i x_i = k$. Usually, in HMM the equation (3.2.3) is used at each iteration, which makes EM algorithm fast. However, in this case (NHMM), we can realize that this method is not applicable to $A(X_t)$ due to the fact that the input value of X_t changes every time. Therefore, This is true that it is only for HMM, and we need to use a numerical optimization to maximize $\varphi(\theta'|\theta)$ at every iteration for NHMM. This EM algorithm process is also naturally needed to use Chow-Liu Tree model, which should be solved separately with transition matrix.

4. SIMULATION (REALIZATION)

The simulation of NHMM is quite straight forward and computationally simple, like the HMM [Figure 1][Figure 2]. First, a hidden state is selected by initial distribution $\delta(x_1)$ conditional on weather data x_1 at time 1, or we can arbitrary select an initial hidden state at time 0 and the hidden state for time 1 is chosen by transition matrix $A(x_1)$. Then, from the selected state in previous step, a rainfall occurrence is selected by emission matrix for time 1. For time 2, using the hidden state picked at time 1 and transition distribution $\delta(x_1)B(r_1)A(x_2)$, a hidden state is earned to select rainfall occurrence. The same procedure continues until the end. However, while calculating the probability of rainfall occurrences given the weather data, a numerical problem must occurs due to that a computation with the parameters tends to quickly close to zero. [*Hughes and Guttorp*, 1994] also introduce a method to avoid this numerical error.

$$P(R_1 \dots R_t \middle| X_1^T)$$
$$= P(R_1 \middle| X_1^T) P(R_2 \middle| R_1, X_1^T) P(R_3 \middle| R_1^2, X_1^T) \dots P(R_T \middle| R_1^{T-1}, X_1^T)$$

$$P(R_1 \middle| X_1^T) = \delta(x_1)B(r_1)$$

$$P(R_T \middle| R_1^{T-1}, X_1^T) = \frac{\delta(x_1)B(r_1)A(x_2) \dots A(x_T)B(r_T)}{\delta(x_1)B(r_1)A(x_2) \dots A(x_{T-1})B(r_{T-1})}$$

5. OPTIMIZATION

To estimate the maximum likelihood, an optimization algorithm is needed. In NHMM, there are generally many local optima and possibly several global optima. The case of the existing several global optima is that the order of hidden states in transition matrix and emission matrix can change in every optimization execution. However, any estimated global optimal parameter has the same likelihood and result , so it can be used for NHMM. Although, there are noble global optimization algorithms, all these takes very long time to find global optima. In this research, two usual global optimization methods are used first, which are multi-start optimization and scatter search optimization. Then, faster methods are compared to find an efficient way.

Multi-Start Optimization is conceptually simple, which randomly starts with certain number of the initial points and finds optimal values from each point using a numerical optimization. More start points increase in a possibility to reach the global optima, but there is a need to control the number of the start points, in order to not over-search the parameters' domain. On the other hand, Scatter Search optimization uses trial or potential points which are generated from the scatter search algorithm [*Ugray et al.*, 2006]. By combining the prior solutions, this method is executed to generate a new solution. Usually, using linear combination between two solutions, a reference set (solution) for the next is selected. After evaluating the potential points, a numerical optimization is used if the points satisfy basin, score, and constraints.

Apart from the methods explained above, there are some faster methods which search the parameters' domain based on random sampling with a statistic or stochastic aid. These are much faster but not always guaranteed to get close to global optima such as genetic algorithm and simulated annealing. Therefore, those processes and characteristics should be known how they give different results according to various setting.

6. RESCALING PARAMETERS USING VITERBI ALGORITHM

In the parameter estimation done in this work, several assumptions are made for the convenience and reducing uncertainty but a problem reoccurs because the average parameters μ_{ij} are not true when we compare to the real average in the data if multivariate normal distributions are used for transition matrix. In this case, the Viterbi Algorithm (VA) is useful because it can gives recursive optimal solution about hidden states in the Markov Chain. If NHMM parameters are already estimated by the simulated annealing or the EM algorithm performed under the assumption taken above, the mean, the covariance, and the transition matrix can be recalculated by the weather data with the states change, using the Viterbi algorithm. Briefly, the goal of the Viterbi algorithm is to find the single best state sequence (path) given observation sequence [*Hughes and Guttorp*, 1994].

Recursion: calculate the probability of path

$$\delta_t(j) = \max_{1 \le i \le N}\left[\delta_{t-1}(i)A_{ij}(X_t)\right]B_j(R_t) \tag{6.1.1}$$

$$\boldsymbol{\varphi_t} = \underset{1 \le i \le N}{argmax}\left[\delta_{t-1}(i)A_{ij}(X_t)\right] \tag{6.1.2}$$

where $\delta_t(j)$ is probability of path at time t, $\boldsymbol{\varphi_t}$ is selected state at time t, $\delta_{t-1}(i)$ is probability of path at time $\mathrm{t}-1$, A_{ij} is transition matrix, and B_j is emission matrix.

Rescaling Parameters:

$$\hat{\Sigma} = \frac{\sum_t \left(X_t - \bar{X}_{s_{t-1}s_t}\right)^2}{T_{s_{t-1}s_t} - 1} \tag{6.1.3}$$

Continue Rescaling Parameters:

$$\mu_{ij}^{*} = \mu_{ij}\Sigma^{-1} * \hat{\Sigma} \tag{6.1.4}$$

$$\gamma_{ij}^{*} \propto \gamma_{ij}\, exp\left(-0.5\mu_{ij}\Sigma^{-1}\left(\mu_{ij} - \mu_{ij}^{*}\right)'\right) \tag{6.1.5}$$

7. MODEL VALIDATION

After model is constructed, the simulation results are examined to determine how close (in a statistical sense) the simulation results are to actual data. First, a sampling method is needed, and its sample mean and confidence interval line are drawn to compare with real data. Second, as was done in [*Hughes and Guttorp*, 1994], the first and second order are estimated to show how the spatial models are well generated. Third, because we focus on precipitation, storm lengths (through survivor functions) are estimated. Using these validation methods, simulation results of NHMM according to the three spatial types are also compared with observed data.

7.1 Survival Function

A survival function (P_s) is to estimate the probability of how many days rainfall occurrence continues, which is related to the probability of rainfall duration. The survivor distribution provides an important measure of goodness-of-fit between data and simulation. The survivor function is

$$\begin{aligned} &P_s(\mathrm{r},\mathrm{d}|\mathrm{X}) \\ &= \mathrm{P}\left(R_{t+d}=r, R_{t+d-1}=r, \dots, R_{t+1}=r \middle| R_{t+1}=r, R_t \neq r, X_t^{t+d}\right) \end{aligned} \tag{7.1.1}$$

where d is number of the survived days after time t. The survivor distribution is estimated using the Kaplan-Meier estimator [*Hughes and Guttorp*, 1994].

$$\widehat{P_s}(\mathrm{r},\mathrm{d}|\mathrm{X}) = \frac{\sum_{t=1}^{T-1} P_s(\mathrm{r},\mathrm{d}|\mathrm{X}) I_d(R_t \neq r, R_{t+1}=r)}{\sum_{t=1}^{T-1} I_d(R_t \neq r, R_{t+1}=r)} \tag{7.1.2}$$

where $I_d(R_t \neq 1, R_{t+1}=1)=1$, and 0 otherwise.

7.2 First and Second Order Properties

First and second order properties are to evaluate statistical properties of each local

uge station and spatial dependences between stations, which explain how well results of

ga the model simulation are close to real data. The first order properties is the expected value of

infall occurrence for a station i, $\mathrm{E}(R_t^i|\mathrm{X}_t)$. It becomes the average value of rainfall

ra currences and amount. On the other hand, the second order properties shows relationship

octween two stations ij, $\mathrm{E}(R_t^i, R_t^j|\mathrm{X}_t)$, which are measured by estimating the covariance or

be orrelations. The spatial correlation has the form of

c

$$\rho_t^{ij} = \frac{\mathrm{Cov}\left(R_t^i, R_t^j|\mathrm{X}_t\right)}{\mathrm{Var}(R_t^i|\mathrm{X}_t)Var(R_t^j|\mathrm{X}_t)} \qquad (7.2.1)$$

8. CASE STUDY

In this section, a case study is presented using the NHMM model with the unconstrained transition matrix equation (3.1.2) and the three spatial models. Five rainfall gauge stations located in one grid scale of the weather image data are considered. The 4 hidden states for independent models are selected by Bayes Information Criterion (BIC) [Table 6]. In the case of using dependent models with the 4 hidden states, the results are not different from independent models in any statistical property. This implies that the 4 hidden states are sufficient to capture spatial dependencies. Therefore, BIC for the dependent models also needs to be estimated: in our case, the 3 hidden states are selected [Table 7]. In this case study, the rainfall amounts are separately simulated using Weibull distribution conditional on the rainfall occurrence.

TABLE 6: MODEL SELECTION USING BIC FOR INDEPENDENT MODELS

Number of Hidden States	Number of Parameters	Log-Likelihood	BIC
2	26	-1683.30	-3472.71
3	51	-1442.16	-3092.46
4	84	-1331.11	**-3005.03**
5	125	-1278.83	-3067.80

TABLE 7 : MODEL SELECTION USING BIC FOR DEPENDENT MODELS

Number of Hidden States	Number of Parameters	Log-Likelihood	BIC
2	46	-1533.64	-3255.01
3	81	-1384.57	**-3099.71**
4	124	-1354.15	-3214.36
5	175	-1327.36	-3368.91

8.1 Data Use

8.1.1 Rainfall Data

Selection of rainfall data is important because precipitation data usually has a lot of missing data, which can result in errors and invalid results in NHMM. In this study, the data for NHMM will be selected which include a small number of missing data. Rainfall data 'R', five stations, will be daily data in Texas and downloaded from National Climatic Data Center (NCDC). The locations and characteristic can see at [Figure 3][Figure 4][Table 8]. The data period is 43 years from 1/1/1958 to 12/31/2000, and the data is transformed into binary data such as 'Rainfall occur' set into '1' and 'not occur' or below 0.25 mm is '0'. Also, the 33 years from 1958 to 1990 are used for training the NHMM and the other 10 years from 1991 to 2000 are employed for a validation of the model prediction.

FIGURE 4: RAINFALL GAUGE STATION MAP (STARS ARE RAINFALL GAUGE STATIONS AND GRID IS 2.5° BY 2.5° SCALES IN ERA40

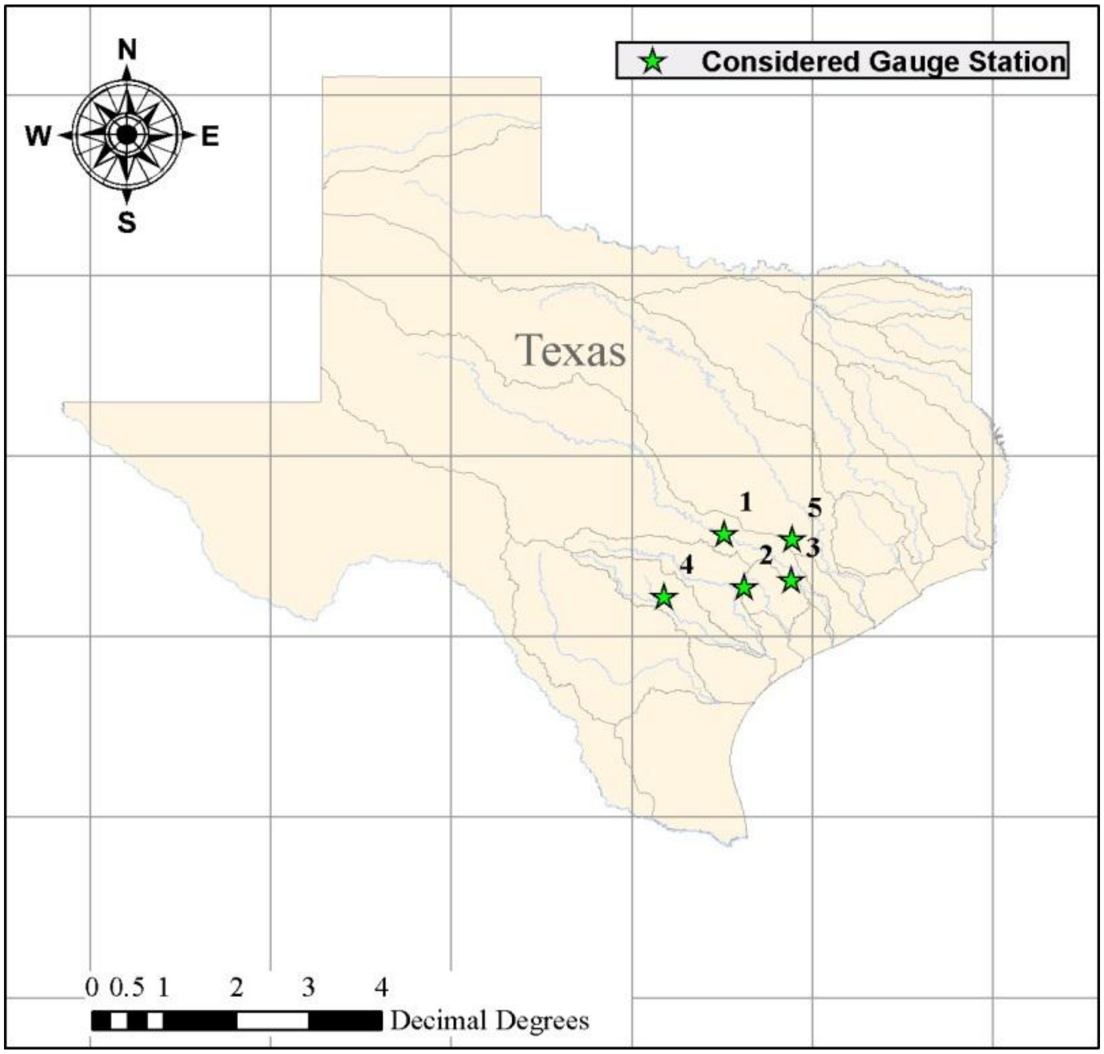
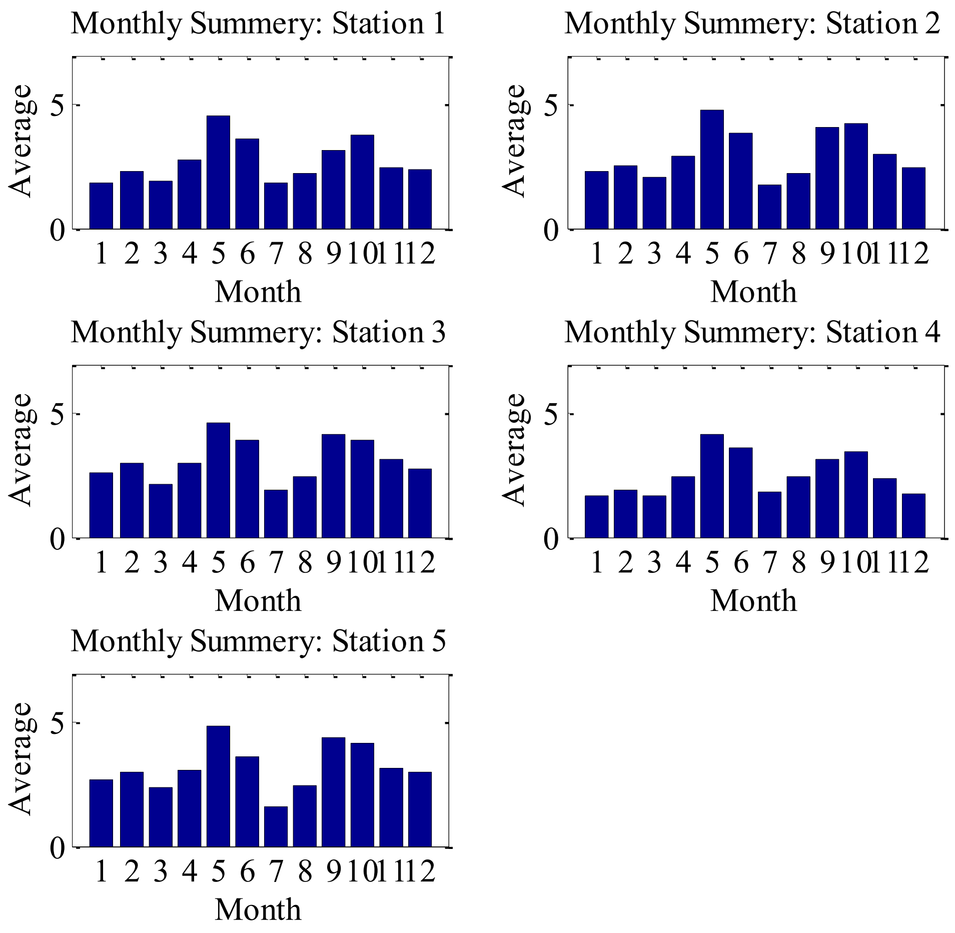


TABLE 8 RAINFALL GAUGE STATION

NUMBER	STATION NAME	LATITUDE	LONGITUDE	ELEVATION
1	AUSTIN-CAMP MABRY	30.19	-97.45	670
2	JEDDO 3S	29.45	-97.18	415
3	LA GRANGE	29.55	-96.52	357
4	SAN ANTONIO	29.32	-98.29	789
5	GIDDINGS 5E	30.11	-96.51	550

In estimating parameters, seasonality has to be reflected, and the order of NHMM models varies according to the number of the seasonality. In this case study, the model is examined using data from February because for these sites the February monthly mean rainfall occurrences are the most highly correlated with both ENSO and SST.

FIGURE 5: MONTHLY SUMMERY FOR EACH STATION



8.1.2 *Atmospheric Data*

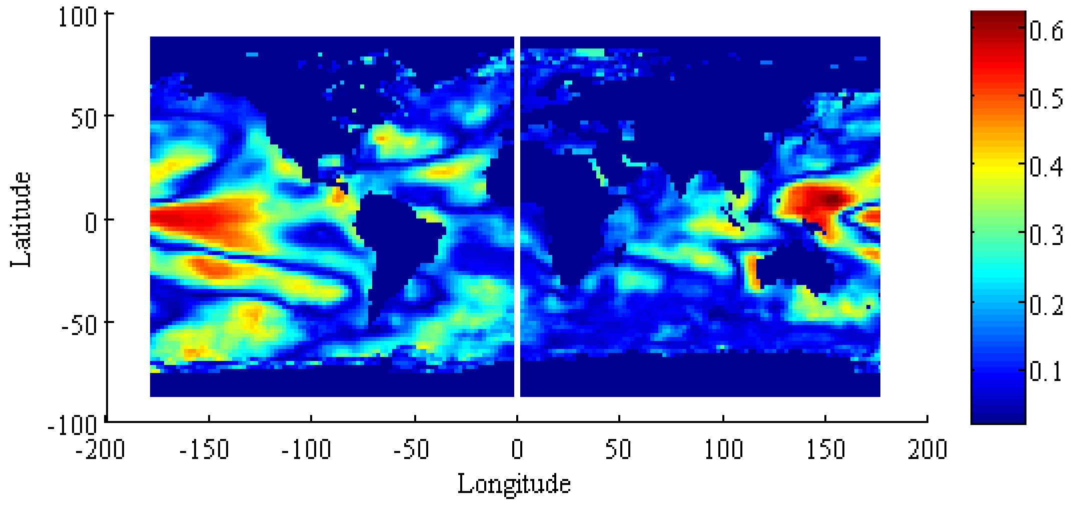
Atmospheric data ‘X’ are ERA40 daily data, El Niño-Southern Oscillation (ENSO) and Sea Surface Temperature (SST) Monthly data. The ERA40 can be downloaded from

European Centre for Medium-Range Weather Forecasts (ECMWF). ERA40 has many atmospheric data sets and the data is used for validation of GCMs because it is made by reanalysis of measurement data. Using ERA40 is different from using GCM results, but the downscaling approach and meaning is not different. Therefore, NHMM model will be conducted using the ERA40 precipitation data having 2.5° by 2.5° grid scale [Figure 3]. The ENSO data are from Climate Prediction Center (CPC) in National Weather Service (NWS). Also Extended Reconstructed Sea Surface Temperature (SST) V3b data with 2.0° by 2.0° grid scale are from Earth System Research Laboratory in National Oceanic and Atmospheric Administration (NOAA). Analysis of correlation between observed rainfall occurrence mean and weather data are presented [Table 9][Figure 5]. In this case study, the first data x1 for the NHMM is ENSO, and the second x2 is SST and third x3 is large scale precipitation.

TABLE 9 : CORRELATION BETWEEN ENSO AND OBSERVED AVERAGE OF MONTHLY RAINFALL OCCURRENCES

Station	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
1	-0.15	**-0.31**	-0.27	-0.03	-0.35	-0.13	0.26	0.31	0.11	0.12	-0.28	-0.35
2	-0.07	**-0.47**	-0.08	-0.20	-0.13	0.06	0.26	0.26	0.05	0.03	-0.17	-0.27
3	-0.09	**-0.26**	0.02	-0.05	-0.19	0.11	0.40	0.29	0.09	0.06	-0.14	-0.31
4	-0.23	**-0.36**	-0.29	-0.03	-0.34	-0.10	0.17	0.35	-0.02	0.10	-0.25	-0.40
5	-0.20	**-0.42**	-0.22	-0.18	-0.21	-0.09	0.36	0.17	-0.16	0.06	-0.17	-0.31
Abs Mean	0.15	**0.36**	0.18	0.10	0.24	0.10	0.29	0.28	0.08	0.07	0.20	0.33

FIGURE 6 : AVERAGE OF CORRELATION BETWEEN SST AND OBSERVED AVERAGE OF MONTHLY RAINFALL OCCURRENCES IN FEBRUARY



8.2 Parameter Estimation and Optimization

The parameter estimation for NHMM is complex and computationally expensive. Some optimization methods were found to be unsuitable (WHICH ONES), since they displayed instability in the simulation or took an overly-long time to reach a global location. Usually the EM algorithm is used but there is also a need to construct various starting points to avoid convergence to local optima. Because the computational time problem and inconvenience of tis formulation, using maximum likelihood is preferred. To reduce these instability and computation-time problems, some steps are needed, which optimize HMM first and then optimize again remaining parameters with the other parameters estimated at the previous step. This procedure shows much improvement in both the optimization time and the stability of optimization results. However, even with the likelihood of the HMM, the parameter estimation is not simple. In this section, some comparisons are explored to find an efficient optimization method for HMM. The EM algorithm also can be more efficient with

HMM than with NHMM but the optimization of the likelihood is only measured because maximizing likelihood shows faster when the two methods are compared.

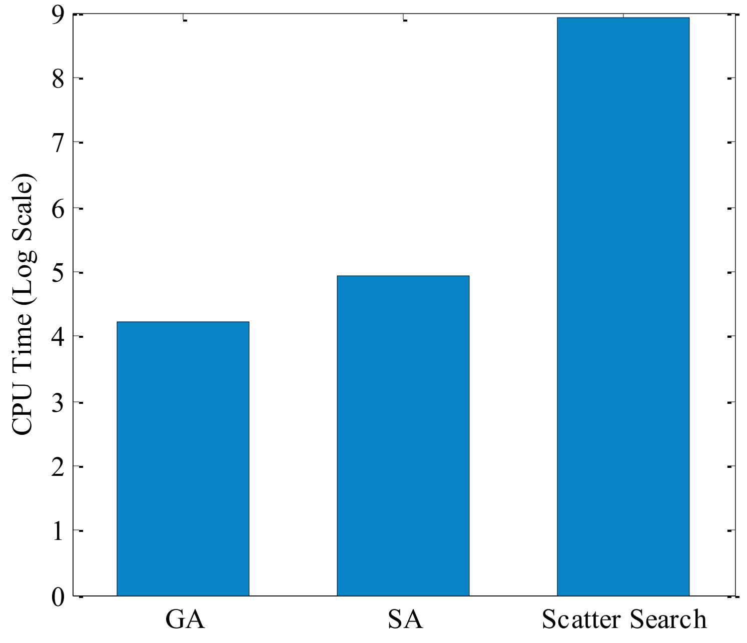
First, multi-start optimization and scatter search optimization are used to find true global optima to make it a standard for a comparison with likelihoods from the other methods. Then, using faster optimization methods, genetic algorithm (GA), simulated annealing (SA) and Markov chain Monte Carlo (MCMC) are compared. Naturally, these methods do not provide exact solutions but provide approximated solutions. Therefore, a numerical optimization should be accompanied to get precise solutions whatever they are global optima or local optima. From this exploration, it is recognized for these sampling based approaches to be very sensitive with the speed of convergence. If the option for the speed of convergence is too fast, the results tend to easily fall into local optimum area, which means that the searching parameter domain does not explore diverse points.

In GA, to increase the searching distance, five properties affect to the convergent time, a population size, a crossover rate, a migration interval, a migration rate and a mutation rate. The population size refers to the number of randomly generated points at the each iteration. An increase in the number of populations can augment all the aspect of the GA's ability but it is needed to be controlled not to dramatically expand the time of the optimization process. A crossover fraction and a mutation rate have a role to change and leave the part of the previously selected values (parents) for the next generation. A migration has a little different characteristic that replaces some parts of worst candidate with those of the best candidate for the next offspring. If the mutation rate increases, the average distance gets bigger: generated population has more randomness around previous parent points. Also, if the crossover rate decreases, the convergence is slower: the inherited rate from parents is low so there is more randomness. The worst case can occur if we do not set crossover and mutation rate properly. In MCMC, the searching ability can be controlled with the change of an acceptance rate. However, in an attempt of using MCMC, it shows instability due to a constraint transition matrix and possibly several global optima in NHMM: some trends and jumping aspects are presented. If using MCMC is preferred, the only spatial models in NHMM can be possible to get stable results as presented by [*Hughes and Guttorp*, 1994]. The SA optimization has a similar method with MCMC but it uses a convergence function on the acceptance rate. At the beginning in SA, it searches a large parameter space and then explores a smaller space according to increase in an iteration time. The convergence function can be controlled by using a different distribution, that there are logarithmic, exponential and linear. In this experiment, only ERA40 data are used and 2 months are considered. In this case study, SA and GA is used with sequential quadratic programming (SPQ) and the ML is not much different from global ML (-3586.09) [Table 9]. Also, the computational time is dramatically reduced when we comparing with a scatter search optimization (7459.7494 CPU time) [Figure 6]. In GA, Crossover fractions from 0.5 to 0.8 and Mutation rates from 0.01 to 0.1 are recommended. In SA, linear distribution is recommended.

TABLE 10: MAXIMUM LIKELIHOOD ESTIMATED BY SA AND GA

GA		SA	
Trial	ML	Trial	ML
1	-3603.04	1	-3602.93
2	-3586.12	2	-3587.03
3	-3586.92	3	-3610.64
4	-3610.31	4	-3587.03
5	-3602.92	5	-3602.95

FIGURE 7 : COMPUTATIONAL TIME COMPARISON



Apart from an independent spatial model and auto-logistic spatial model, EM algorithm has to be used for a Chow-Liu tree model. First, estimate HMM as the same as other spatial models. Then, EM algorithm is employed: through E-step, computes the expectation of the log-likelihood and state sequences are estimated by Viterbi Algorithm and, then, parameters of transition matrix and a Chow-Liu tree model are separately estimated in M-step.

8.3 Simulation Results

By SA and SPQ, NHMM parameters with each spatial type model are estimated as the procedure in section 7.2, and Monte Carlo simulation is employed. From the results are evaluated by a survival function, first-order, second order properties.

In independent models, 1000 times Monte Carlo simulations are first conducted with 95% intervals [Figure 7]. The monthly sum of simulated daily rainfall occurrence is well generated when observed rainfall occurrence is compared. Looking at each year, they have some differences between observation and simulation, but the NHMM well captures those trends. Even in not trained periods, the trends are also well predicted. In [Table 11], correlations are estimated between monthly observation and simulation, which shows more than 0.7 in all aspects. Also, survival analysis are also estimated at [Figure 8][Figure 9] and the characters related to storm durations are properly preserved in the models. When we see the observation and simulation in entire periods [Figure 10][Figure 11][Table 13], the number of rainfall occurrence and the amount are simulated closely to those observations. Although the number of rainfall occurrence in predictive periods at several stations seems to have much difference compared to its observation, the first order or the probability of rainfall occurrence shows they are also quite well preserved [Table 14]. [Figure 12][Table 15][Table 16] present the second order properties or spatial correlation among rainfall gauge stations. As I mentioned before, with 4 hidden states other spatial models, auto-logistic and Chow-Liu Tree, have very similar results with this independent models, which means the number of hidden states is adequate to capture spatial correlation. This is self-evidence because the number of gauge stations and the number of hidden states are similar. If we consider many and widely spread stations, the results will be very different or much more hidden states will be needed.

FIGURE 8 : SIMULATION RESULTS OF INDEPENDENT MODELS FOR RAINFALL OCCURRENCE WITH 4 STATES. VERTICALLY DOTTED LINES WITH BOXES REFER TO 1990 OR START POINTS OF PREDICTIONS.

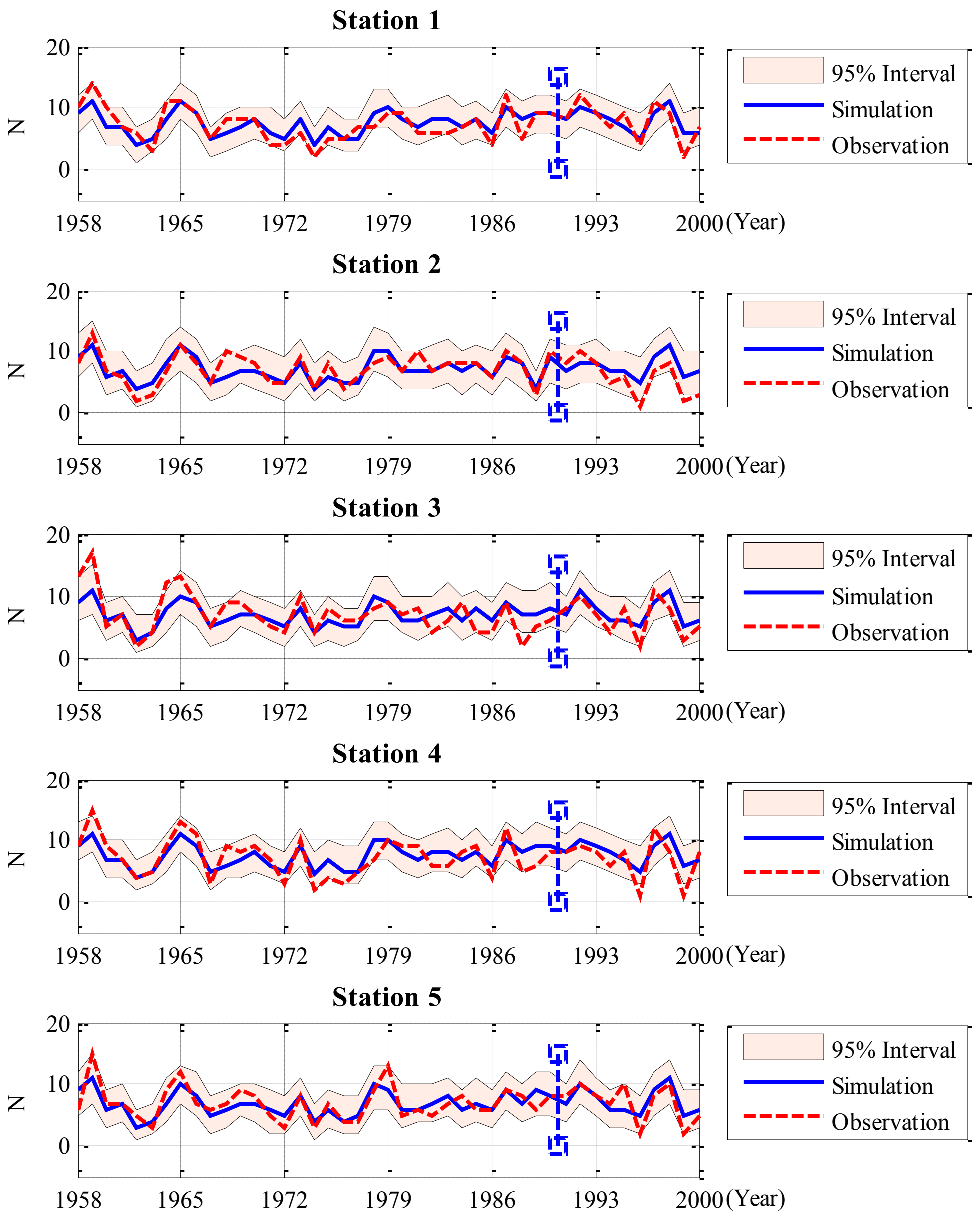


TABLE 11 : CORRELATION BETWEEN OBSERVATION AND SIMULATION OF [FIGURE 7]

Simulation Correlation

Station	Training Periods	Predictive Periods
1	0.7641	0.7882
2	0.8327	0.7242
3	0.7184	0.7523
4	0.8020	0.7240
5	0.8002	0.7746
Average	*0.7835*	*0.7527*

FIGURE 9 : SURVIVAL ANALYSIS OF INDEPENDENT MODELS WITH 4 STATES IN TRAINING PERIODS.

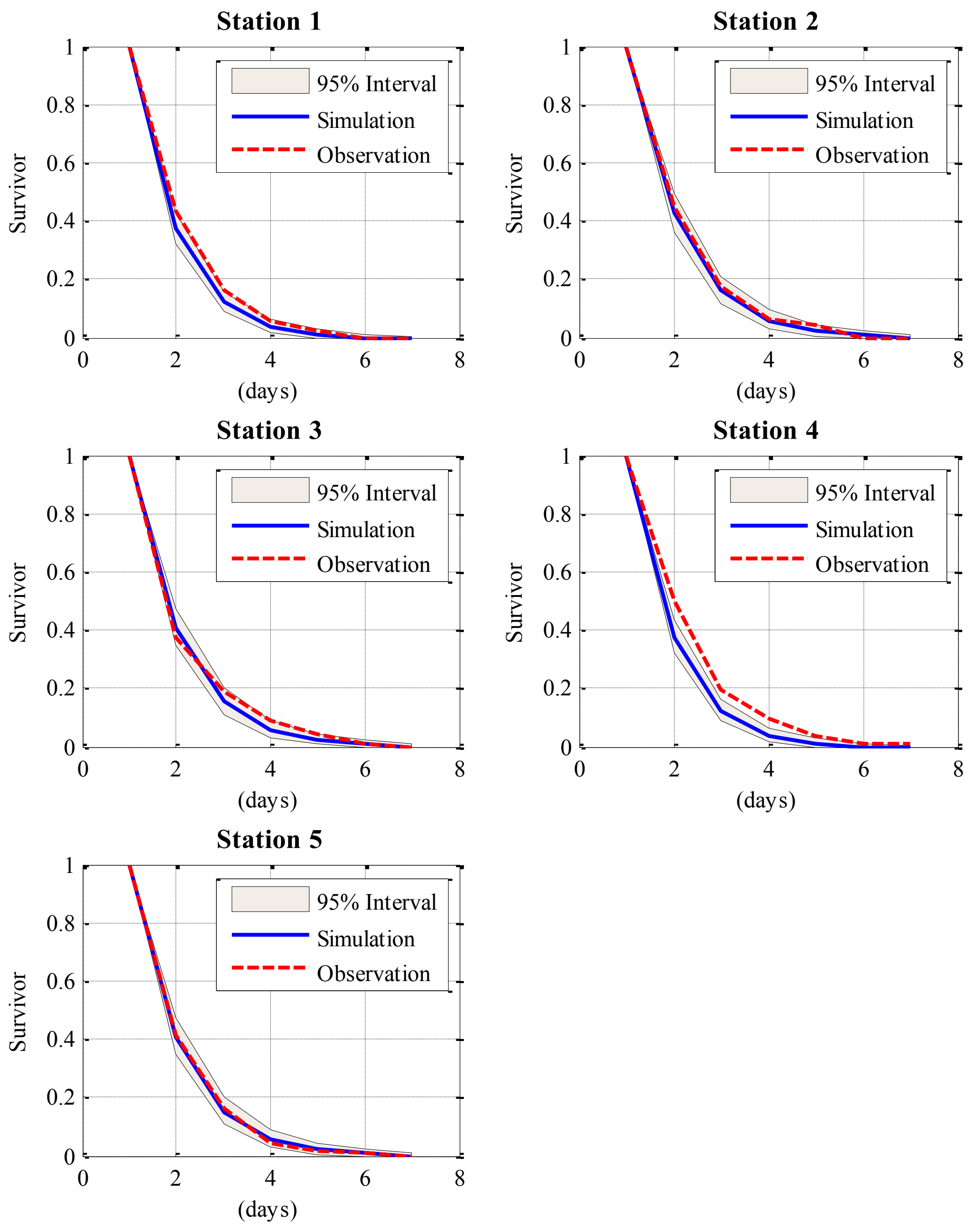


FIGURE 10 : SURVIVAL ANALYSIS OF INDEPENDENT MODELS WITH 4 STATES IN PREDICTIVE PERIODS.

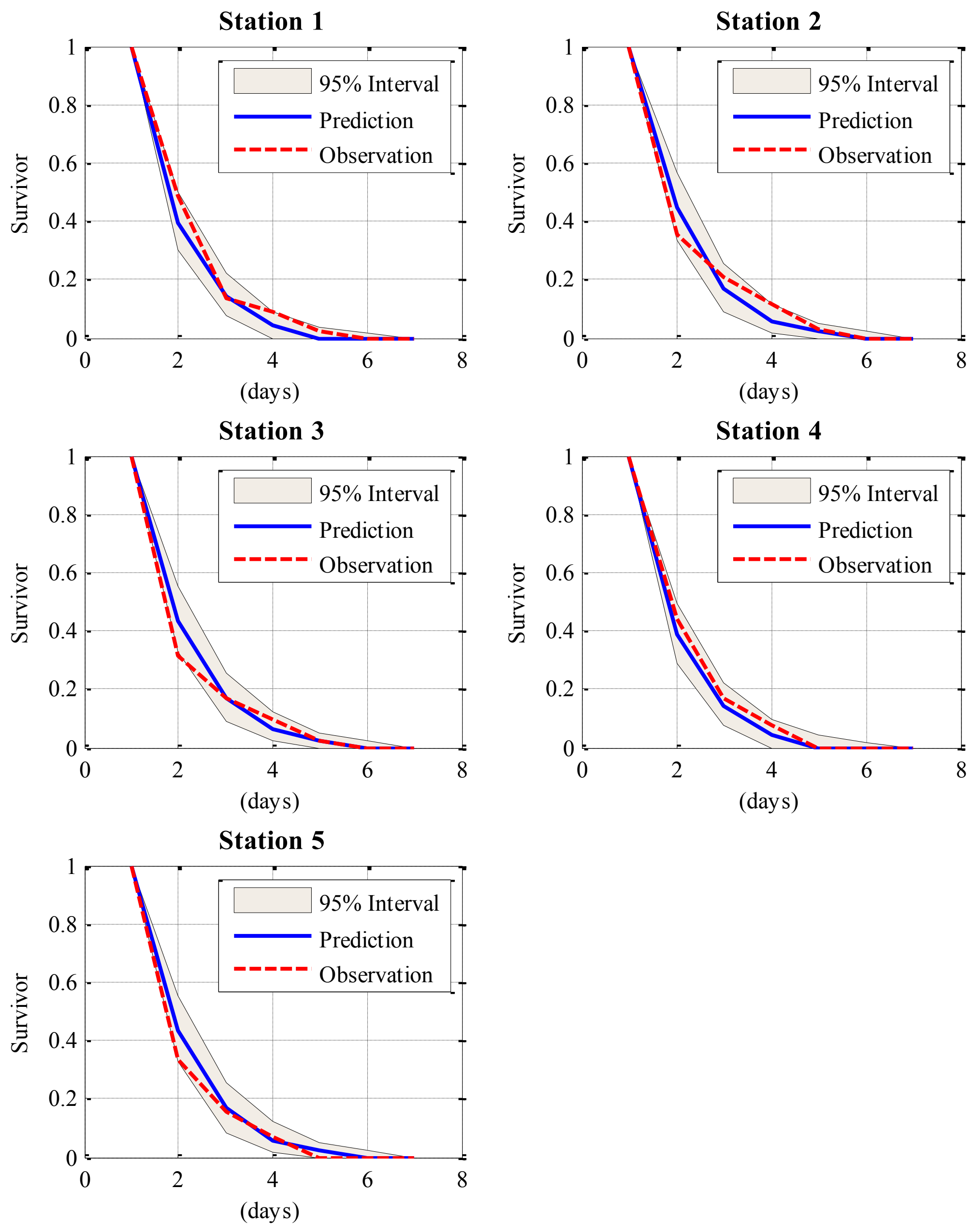


FIGURE 11 : RAINFALL OCCURRENCE AND AMOUNT OF INDEPENDENT MODELS WITH 4 STATES IN TRAINING PERIODS.

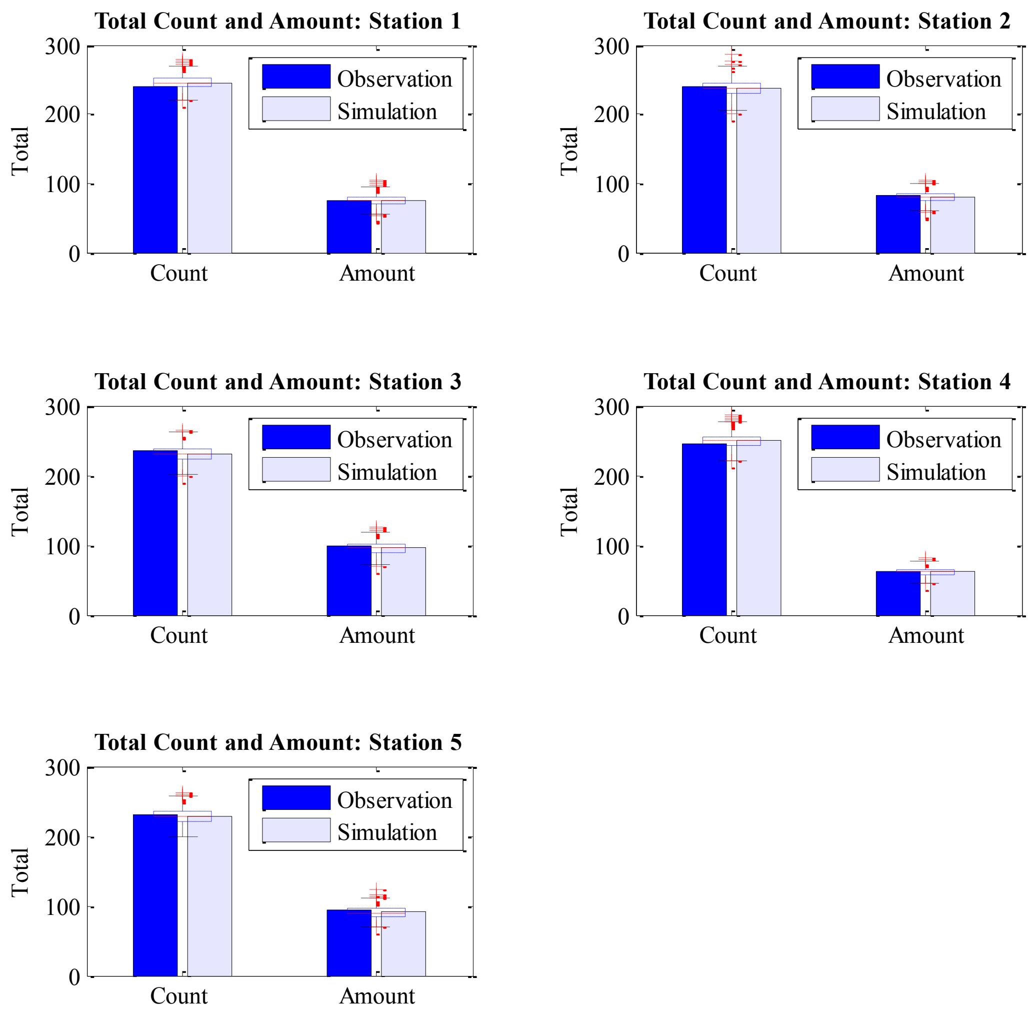


FIGURE 12 : RAINFALL OCCURRENCE AND AMOUNT OF INDEPENDENT MODELS WITH 4 STATES IN PREDICTIVE PERIODS.

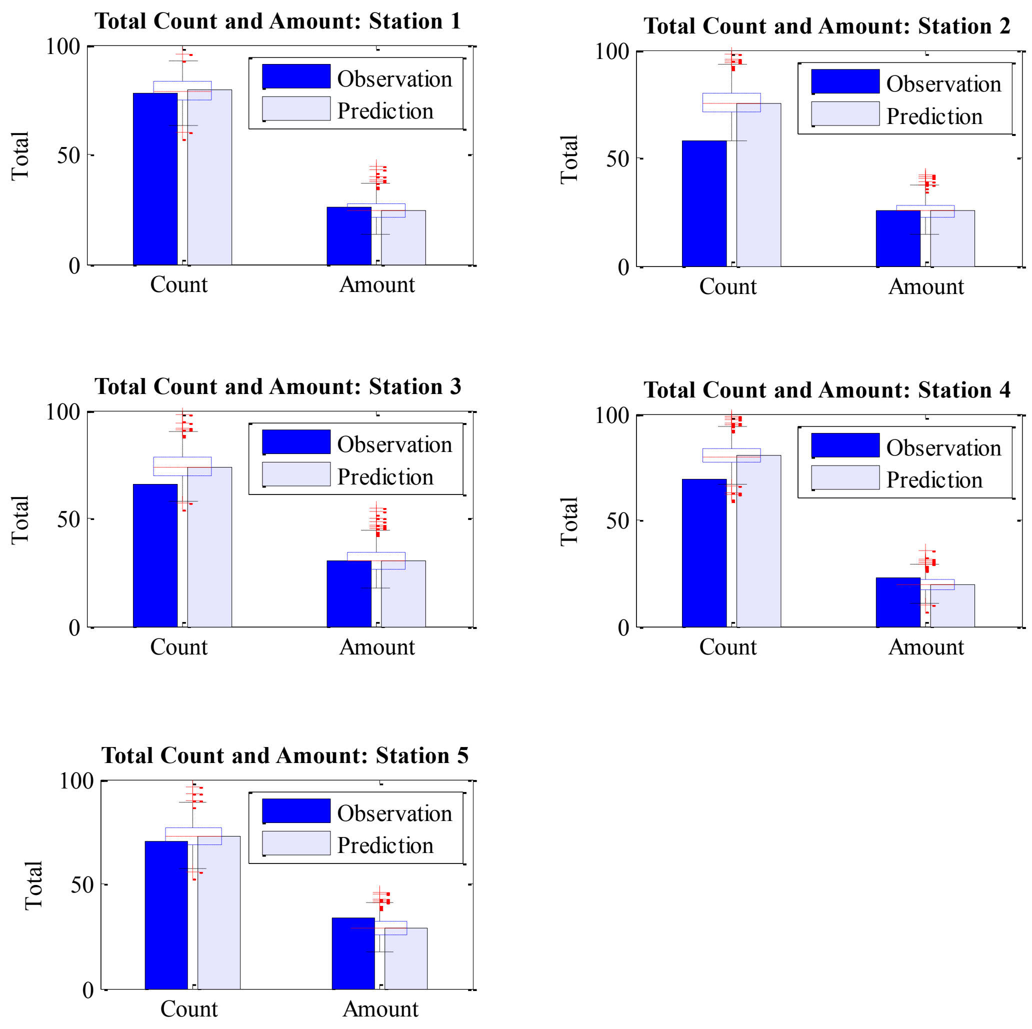


TABLE 12 : RAINFALL OCCURRENCE AND AMOUNT OF INDEPENDENT MODELS WITH 4 STATES IN TRAINING PERIODS AND PREDICTIVE PERIODS.

Count		Amount	
Observed	Trained	Observed	Trained
240.00	249.54	76.03	77.56
240.00	244.44	81.93	82.36
236.00	236.75	98.25	97.63
244.00	253.60	62.04	62.90
231.00	233.44	92.69	92.85

Count		Amount	
Observed	Predicted	Observed	Predicted
78.00	78.19	25.86	24.15
58.00	74.92	25.73	25.29
66.00	73.82	30.34	30.58
69.00	79.28	22.97	19.74
70.00	72.76	33.55	29.03

TABLE 13 : PROBABILITY OF RAINFALL OCCURRENCE WITH 4 STATES IN TRAINING PERIODS AND PREDICTIVE PERIODS.

Observed	Trained
0.26	0.27
0.26	0.27
0.25	0.25
0.26	0.27
0.25	0.25

Observed	Predicted
0.28	0.28
0.21	0.27
0.23	0.26
0.24	0.28
0.25	0.26

FIGURE 13 : SPATIAL CORRELATION OF INDEPENDENT MODELS WITH 4 STATES, DIFFERENCES BETWEEN OBSERVATION AND SIMULATION, AND ENSEMBLE STANDARD DEVIATION.

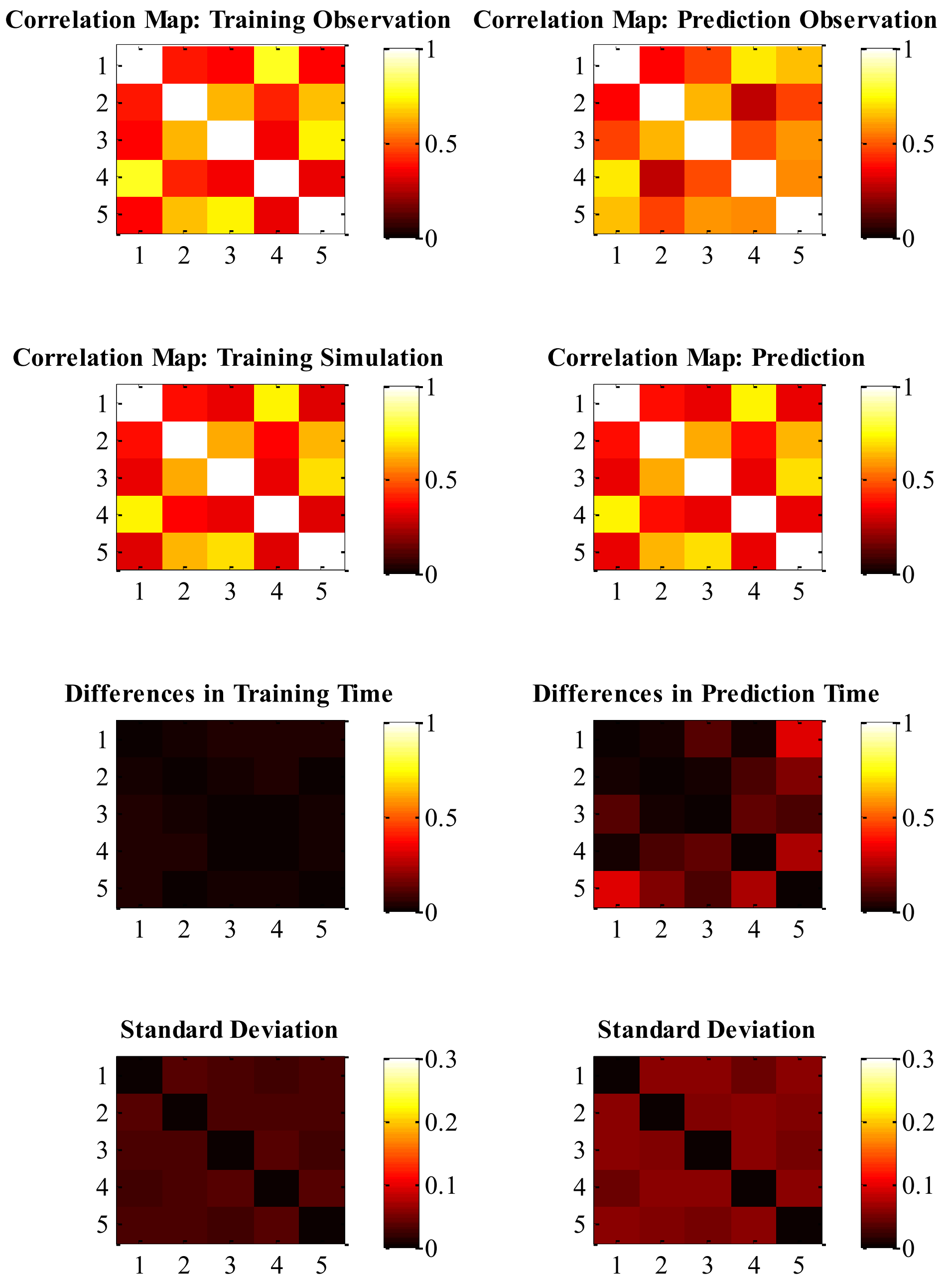


TABLE 14 : OBSERVED SPATIAL CORRELATION IN TRAINING PERIODS AND PREDICTIVE PERIODS

Training				j		
		1.0000	0.4020	0.3701	0.7698	0.3649
		0.4020	1.0000	0.6379	0.4160	0.6468
	i	0.3701	0.6379	1.0000	0.3451	0.7213
		0.7698	0.4160	0.3451	1.0000	0.3399
		0.3649	0.6468	0.7213	0.3399	1.0000

Prediction				j		
		1.0000	0.3639	0.4665	0.7074	0.6521
		0.3639	1.0000	0.6355	0.2730	0.4681
	i	0.4665	0.6355	1.0000	0.4690	0.5937
		0.7074	0.2730	0.4690	1.0000	0.5660
		0.6521	0.4681	0.5937	0.5660	1.0000

TABLE 15 : SIMULATED SPATIAL CORRELATION IN TRAINING PERIODS AND PREDICTIVE PERIODS WITH 4 STATES IN INDEPENDENT MODELS

Training				j		
		1.0000	0.3755	0.3312	0.7246	0.3192
		0.3755	1.0000	0.6105	0.3726	0.6387
	i	0.3312	0.6105	1.0000	0.3323	0.7001
		0.7246	0.3726	0.3323	1.0000	0.3216
		0.3192	0.6387	0.7001	0.3216	1.0000

Prediction				j		
		1.0000	0.3828	0.3421	0.7301	0.3304
		0.3828	1.0000	0.6142	0.3805	0.6403
	i	0.3421	0.6142	1.0000	0.3416	0.7006
		0.7301	0.3805	0.3416	1.0000	0.3315
		0.3304	0.6403	0.7006	0.3315	1.0000

TABLE 16 : PARAMETERS OF TRANSITION MATRIX WITH 4 STATES IN INDEPENDENT MODELS

a

	j			
i	-18.8231	-4.0869	-34.2107	-35.1237
	-22.0533	-12.3186	-27.0533	-31.0924
	-20.3762	-24.8358	-25.1711	-29.6773
	-24.9385	-22.5637	-21.0431	-24.2365

b

	i				
x1	0.0490	0.1076	-0.2674	0.1997	j(1)
x2	-0.1390	-0.2045	-0.1210	-0.3875	
x3	0.4796	-9.6435	1.0731	2.8938	

	i				
x1	-0.0932	-0.0787	-0.0741	-0.0494	j(2)
x2	-0.6732	-0.5623	-0.0450	-0.5501	
x3	0.8967	11.5725	1.3899	3.2405	

	i				
x1	0.1381	0.0074	-0.2250	-0.0631	j(3)
x2	0.4564	0.0459	0.1471	-0.4208	
x3	-2.3111	-12.7671	-2.5122	-4.6328	

	i				
x1	-0.0938	-0.0363	0.5666	-0.0872	j(4)
x2	0.4962	0.2031	0.0914	-0.4421	
x3	0.0742	10.3504	0.1192	-1.5015	

TABLE 17 : PARAMETERS OF EMISSION MATRIX WITH 4 STATES IN INDEPENDENT MODELS

		Station				
		1	2	3	4	5
States	1	0.7533	0.2104	0.0796	0.7341	0.0509
	2	0.9584	0.8415	0.8658	0.9496	0.8565
	3	0.0066	0.0197	0.0205	0.0159	0.0088
	4	0.0696	0.7401	0.7829	0.0966	0.8288

To compare spatial models, 3 hidden states are used as selected by BIC for auto-logistic models. First, independent models with 3 hidden states are simulated to compare other models. When we see both [Figure 12] and [Figure 13], with 3 hidden states the independent models cannot capture spatial correlation as well as with 4 hidden states. However, other models, auto-logistic and Chow-Liu tree models have ability to parameterize rainfall occurrence patterns even in smaller number of hidden states [Figure 14][Figure 15]. These spatial dependent models could also be comparable with 4 states independent models. This difference between independent models and dependent models are accelerated if the number of considered gauge stations increases. If we also compare the two dependent models, the auto-logistic models are little better than Chow-Liu tree models when we see the spatial correlations and correlation between the observation and simulation [Figure 14][Figure 15][Table 19]. However, in auto-logistic models, the number of parameters for spatial dependency dramatically increases when the number of considered point stations rises. Otherwise, the Chow-Liu tree not only has smaller number of parameters than the auto-logistic models with a large number of points: $\mathrm{N}(\mathrm{N}-1)/2$ parameters with auto-logistic models and $4(\mathrm{N}-1)$ parameters with Chow-Liu tree models in dependent terms, but it also has own parameter estimation method (MWST). This method makes the time of its parameter estimation not much affected with any number of small scale points.

Simulation results for rainfall amount at [Figure 10][Figure 11] seems well generated by Weibull distributions. However, at [Figure 16] which is also earned by the ensemble mean of Monte Carlo simulations, we can see that a simple distribution cannot capture rainfall amount patterns for each year. Also, because this rainfall amounts are generated conditional on rainfall occurrence, we can conclude that rainfall occurrence is not only related its amount. To simulate the amount, additional inputs are needed to improve the amount simulation.

TABLE 18 : CORRELATION BETWEEN OBSERVATION AND SIMULATION WITH 3 STATES

Independent			Auto-Logistic			Chow-Liu		
Station	Trained	Predicted	Station	Trained	Predicted	Station	Trained	Predicted
1	0.7825	0.7140	1	0.7068	0.7514	1	0.7722	0.7984
2	0.8350	0.6137	2	0.7918	0.9362	2	0.8369	0.7950
3	0.6849	0.7143	3	0.8022	0.7303	3	0.7435	0.6216
4	0.8249	0.5878	4	0.8056	0.6039	4	0.8624	0.6137
5	0.6900	0.8140	5	0.7934	0.8867	5	0.7570	0.7363
Mean	*0.7635*	*0.6888*	*Mean*	*0.7800*	*0.7817*	*Mean*	*0.7944*	*0.7130*

FIGURE 14 : SPATIAL CORRELATION OF INDEPENDENT MODELS WITH 3 STATES, DIFFERENCES BETWEEN OBSERVATION AND SIMULATION, AND ENSEMBLE STANDARD DEVIATION.

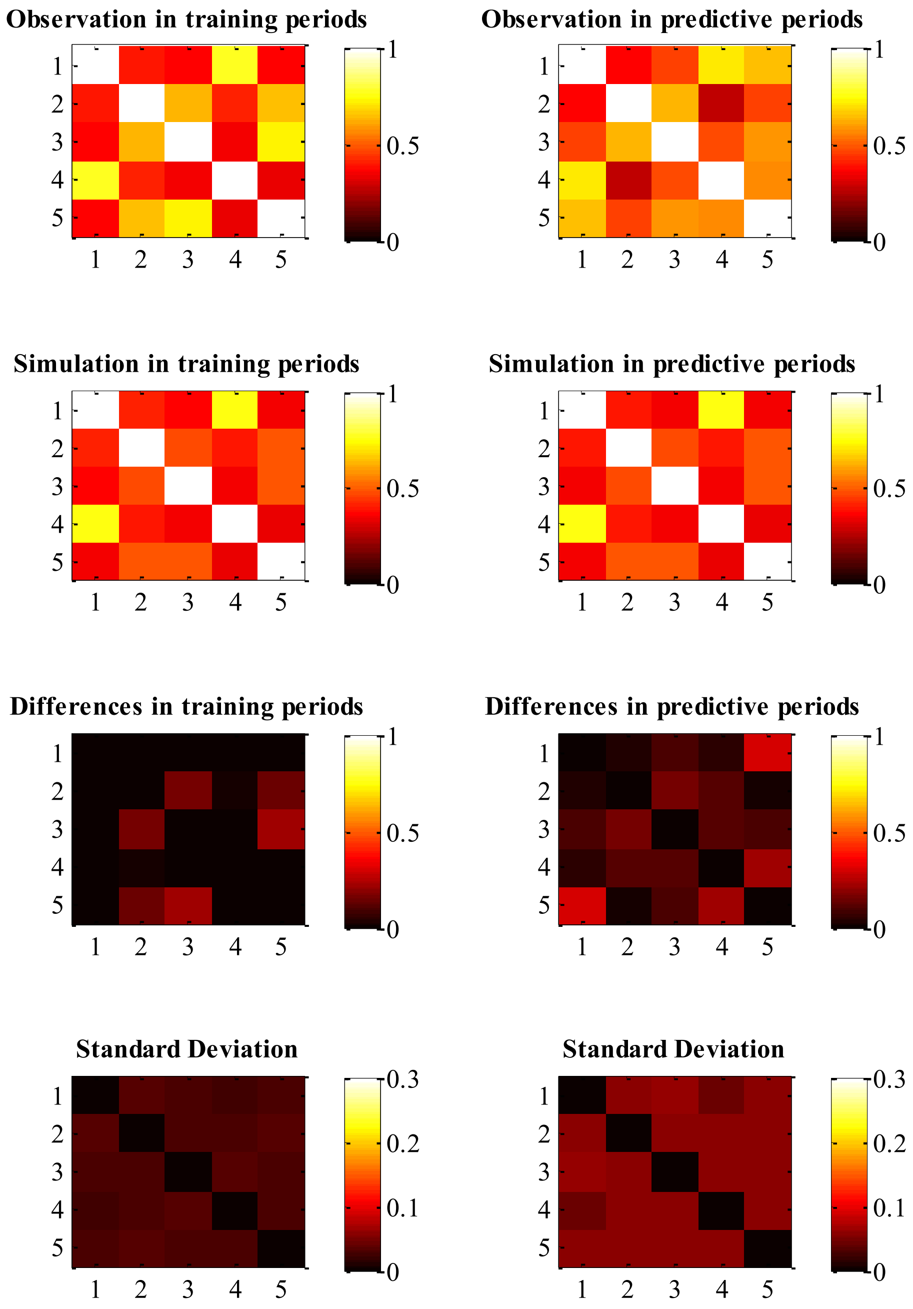


FIGURE 15 : SPATIAL CORRELATION OF AUTO-LOGISTIC MODELS WITH 3 STATES, DIFFERENCES BETWEEN OBSERVATION AND SIMULATION, AND ENSEMBLE STANDARD DEVIATION.

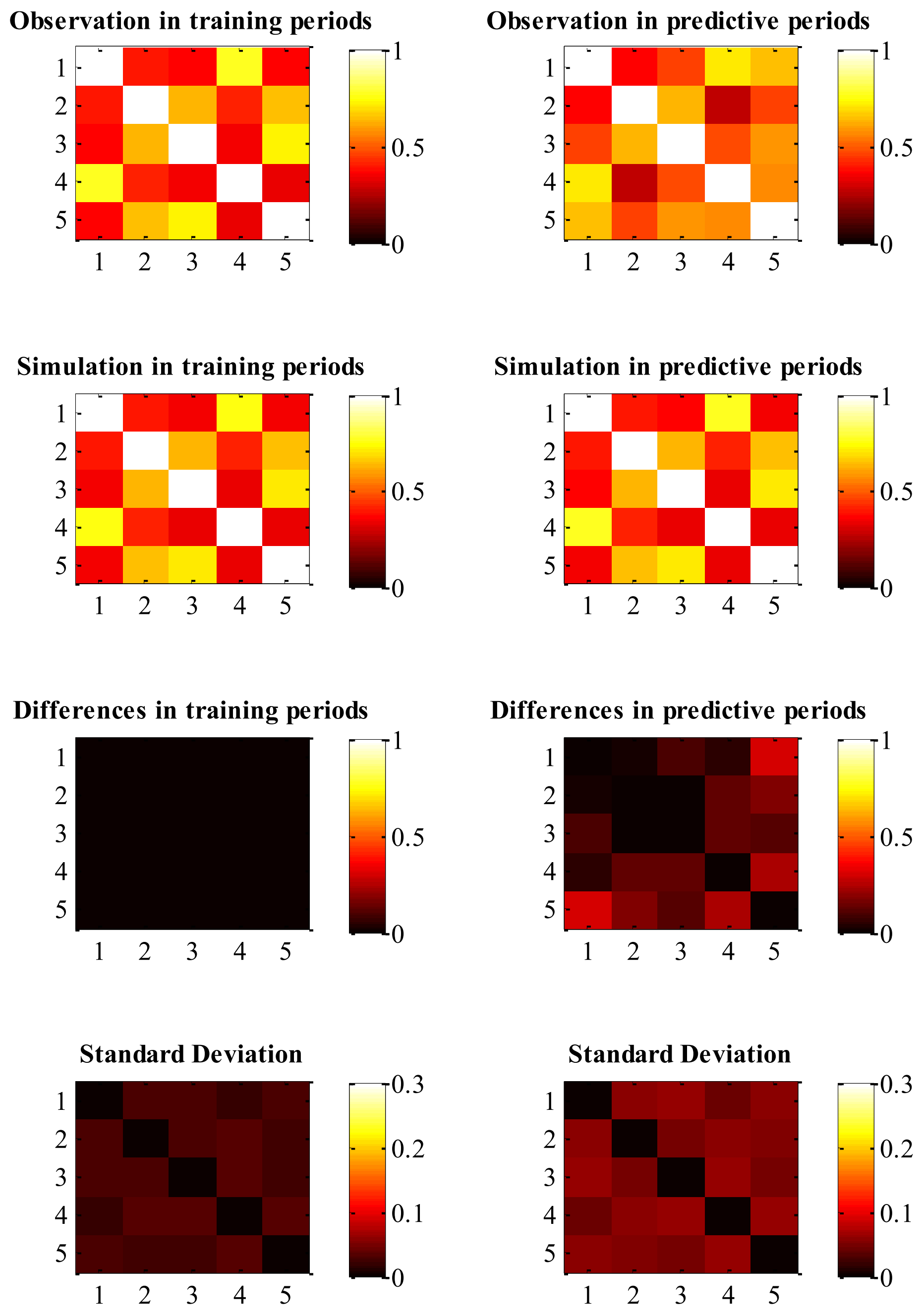


FIGURE 16 : SPATIAL CORRELATION OF CHOW-LIU TREE MODELS WITH 3 STATES, DIFFERENCES BETWEEN OBSERVATION AND SIMULATION, AND ENSEMBLE STANDARD DEVIATION.

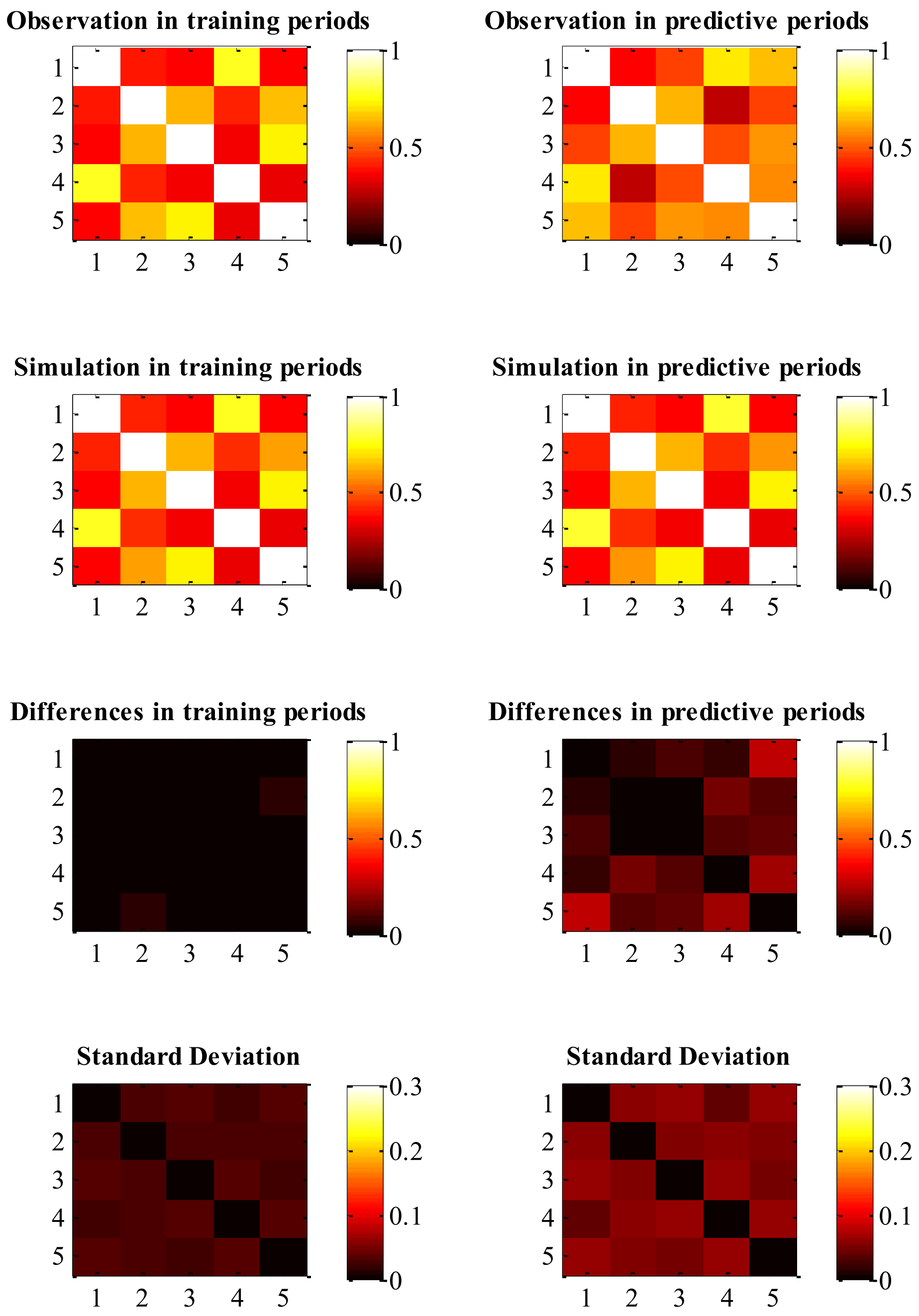


TABLE 19 : SIMULATED SPATIAL CORRELATION IN TRAINING PERIODS AND PREDICTIVE PERIODS WITH 3 STATES IN INDEPENDENT MODELS

Training	j				
	1.0000	0.4087	0.3629	0.7559	0.3530
	0.4087	1.0000	0.4787	0.3947	0.4984
i	0.3629	0.4787	1.0000	0.3496	0.4962
	0.7559	0.3947	0.3496	1.0000	0.3406
	0.3530	0.4984	0.4962	0.3406	1.0000

Prediction	j				
	1.0000	0.4054	0.3579	0.7607	0.3472
	0.4054	1.0000	0.4766	0.3946	0.4950
i	0.3579	0.4766	1.0000	0.3476	0.4940
	0.7607	0.3946	0.3476	1.0000	0.3366
	0.3472	0.4950	0.4940	0.3366	1.0000

TABLE 20 : SIMULATED SPATIAL CORRELATION IN TRAINING PERIODS AND PREDICTIVE PERIODS WITH 3 STATES IN AUTO-LOGISTIC MODELS

Training	j				
	1.0000	0.3918	0.3579	0.7652	0.3506
	0.3918	1.0000	0.6319	0.4081	0.6445
i	0.3579	0.6319	1.0000	0.3329	0.7157
	0.7652	0.4081	0.3329	1.0000	0.3310
	0.3506	0.6445	0.7157	0.3310	1.0000

Prediction	j				
	1.0000	0.3943	0.3606	0.7689	0.3508
	0.3943	1.0000	0.6296	0.4097	0.6411
i	0.3606	0.6296	1.0000	0.3325	0.7167
	0.7689	0.4097	0.3325	1.0000	0.3302
	0.3508	0.6411	0.7167	0.3302	1.0000

TABLE 21 : SIMULATED SPATIAL CORRELATION IN TRAINING PERIODS AND PREDICTIVE PERIODS WITH 3 STATES IN CHOW-LIU TREE MODELS

Training	j				
	1.0000	0.4111	0.3697	0.7867	0.3620
	0.4111	1.0000	0.6413	0.4176	0.6022
i	0.3697	0.6413	1.0000	0.3510	0.7223
	0.7867	0.4176	0.3510	1.0000	0.3366
	0.3620	0.6022	0.7223	0.3366	1.0000

Prediction	j				
	1.0000	0.4118	0.3689	0.7871	0.3615
	0.4118	1.0000	0.6411	0.4217	0.5954
i	0.3689	0.6411	1.0000	0.3537	0.7190
	0.7871	0.4217	0.3537	1.0000	0.3392
	0.3615	0.5954	0.7190	0.3392	1.0000

TABLE 22 : PARAMETERS FOR INDEPENDENT MODELS WITH 3 STATES

a

	j		
	50.0000	-48.7546	-50.0000
i	50.0000	-50.0000	41.5849
	21.2609	-48.0055	-16.0351

b

	i			
x1	0.1117	0.1068	-0.0613	
x2	-2.3992	-1.2207	-1.2845	j(1)
x3	2.7967	0.6540	1.4391	

	i			
x1	-0.1638	-0.1616	0.1602	
x2	1.1823	2.3145	1.1341	j(2)
x3	1.9667	0.4154	0.3217	

	i			
x1	0.0520	0.0548	-0.0989	
x2	1.2169	-0.8159	0.1509	j(3)
x3	-5.9723	-1.0695	-1.7782	

Alpha

		Station				
		1	2	3	4	5
	1	0.9450	0.5915	0.5381	0.9295	0.5191
States	2	0.0913	0.7406	0.7687	0.0996	0.8083
	3	0.0300	0.0218	0.0215	0.0410	0.0093

TABLE 23 : PARAMETERS FOR AUTO-LOGISTIC MODELS WITH 3 STATES

a	j		
	-4.4772	-11.4780	12.2543
i	-50.0000	50.0000	-50.0000
	49.5554	-24.6982	-42.2346

b	i			
x1	-0.2450	-0.1150	-0.8909	
x2	0.0717	0.8972	-2.7816	j(1)
x3	0.7477	4.7747	-0.0125	

	i			
x1	0.1785	0.1253	0.4658	
x2	0.3294	-2.7338	1.0166	j(2)
x3	1.3791	5.4489	2.0434	

	i			
x1	0.0666	-0.0102	0.4251	
x2	-0.4011	0.8617	1.7650	j(3)
x3	-2.1509	-10.2236	-2.0967	

TABLE 23 : CONTINUED

Alpha

		Station				
		1	2	3	4	5
States	1	-42.3388	0.6296	-0.3823	-3.8433	0.3101
	2	0.4267	-1.9218	-2.9232	0.1656	-22.6021
	3	-3.8646	-4.1276	-3.6756	-3.5186	-4.4287

Beta

	j				
	20.3156	1.1152	-14.6834	19.2483	
		0.1954	1.8516	0.1490	
i			-0.3483	1.6338	State 1
				0.7836	

	j				
	0.3227	1.2269	1.4767	20.0641	
		1.2816	0.8247	1.1962	
i			-0.0158	2.4077	State 2
				0.6997	

	j				
	-32.7714	-25.8753	3.6958	2.1912	
		3.1841	2.1925	-22.0503	
i			-0.4563	2.3716	State 3
				-21.8593	

TABLE 24 : PARAMETERS FOR CHOW-LIU TREE MODELS WITH 3 STATES

a	j		
	50.0000	-50.0000	-45.7114
i	0.4772	-20.0802	-28.8756
	50.0000	-45.6514	-50.0000

b	i			
x1	0.1129	-0.2502	0.1985	
x2	-2.3164	-0.5922	-2.3424	j(1)
x3	2.8226	1.3527	0.8157	

	i			
x1	0.0450	-0.2404	-0.1275	
x2	1.2886	0.2413	1.1781	j(2)
x3	-5.2673	-1.9083	-1.4535	

	i			
x1	-0.1579	0.4906	-0.0710	
x2	1.1485	0.3589	1.1737	j(3)
x3	2.0205	0.5057	0.6372	

TABLE 24 : CONTINUED

Alpha

		Station				
		1	2	3	4	5
States	1	0.9217	0.5739	0.5348	0.9348	0.5043
	2	0.0180	0.0233	0.0197	0.0305	0.0162
	3	0.0945	0.7480	0.7795	0.0709	0.8031

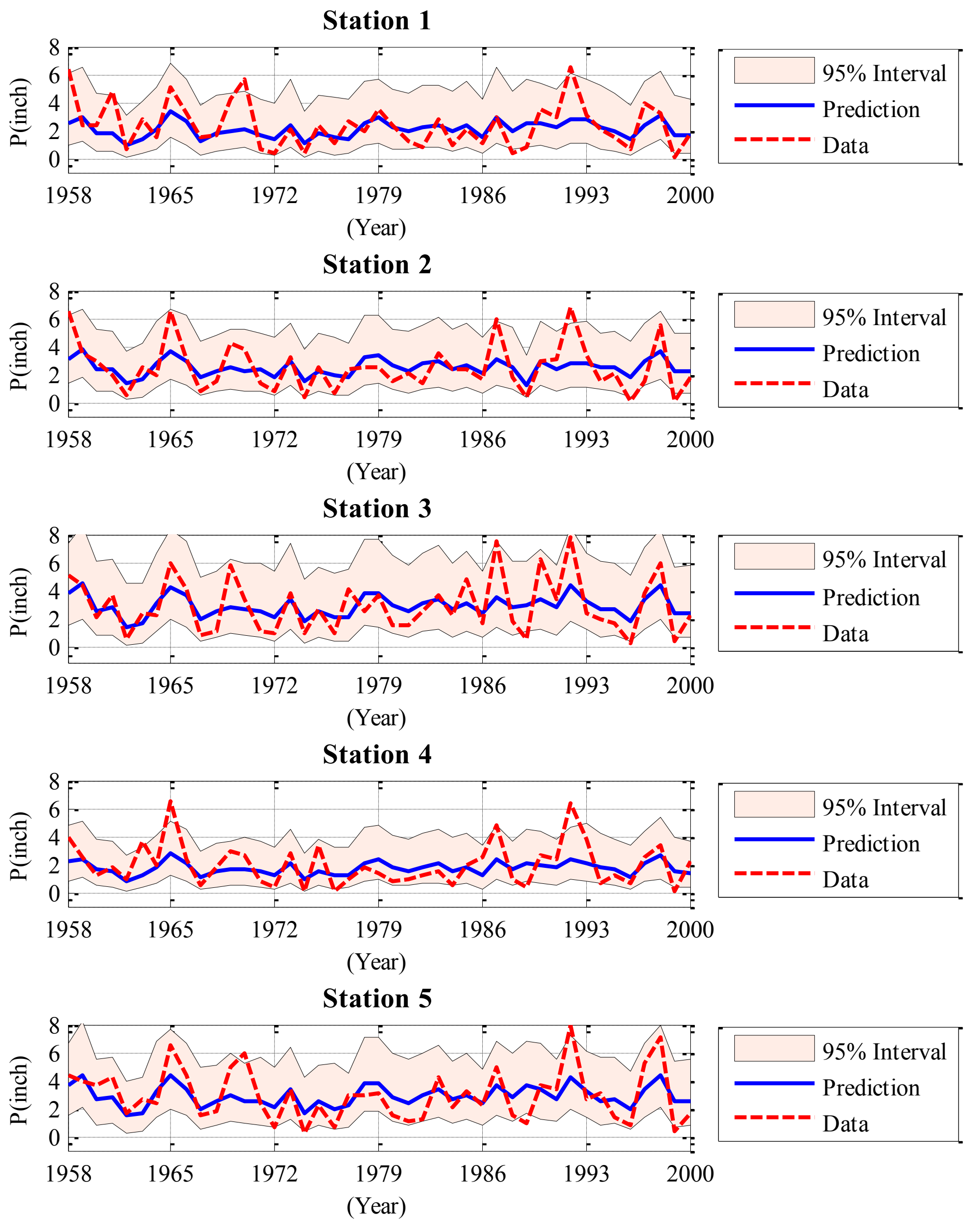
Beta

From	To	0 0	0 1	1 0	1 1	
1	2	0.0261	0.0522	0.4000	0.5217	State 1
2	3	0.3217	0.1043	0.1435	0.4304	
2	4	0.0478	0.3783	0.0174	0.5565	
3	5	0.3870	0.0783	0.1087	0.4261	

From	To	0 0	0 1	1 0	1 1	
1	5	0.9677	0.0144	0.0162	0.0018	State 2
5	2	0.9659	0.0180	0.0108	0.0054	
2	4	0.9479	0.0287	0.0215	0.0018	
4	3	0.9497	0.0197	0.0305	0.0000	

From	To	0 0	0 1	1 0	1 1	
1	2	0.2520	0.6535	0.0000	0.0945	State 3
1	5	0.1969	0.7087	0.0000	0.0945	
5	3	0.0787	0.1181	0.1417	0.6614	
3	4	0.1811	0.0394	0.7480	0.0315	

FIGURE 17 : SIMULATION RESULTS OF INDEPENDENT MODELS FOR RAINFALL AMOUNT WITH 4 STATES.



9. SUMMARY AND CONCLUSIONS

In this study, NHMM models are explored as a means of generating precipitation occurrence which depends on large scale atmospheric data. First, as compared to HMM, with respect to using large scale data as input data in NHMM for downscaling, it has an advantage of capturing nonstationary process in small scale precipitation. Also, the assumption of multivariate normal distribution or an auto-logistic model for the transition matrix in NHMM is more reasonable than simple assumptions presented at [Table 1]. Additionally, the spatial models in the NHMM are explored to find each characteristic of how they affect to whole NHMM models such as the simulation results and the computational time, and of which spatial model would be more efficient. The results in the case study show that the NHMM models have the ability to conduct both simulations and predictions. However, usually this method has problems when many spatial points are considered. In [Table 18], the correlation of results seems not to have much different among three spatial models, a model having many parameters tends to better ability to capture spatial patterns. With independent models, the BIC selection criterion for choosing the number of hidden states in NHMM tends to select models with many states to capture spatial dependency. If there are various spatial patterns with many spatial points, it will yield poorer results. Therefore, NHMM combined with spatial dependent models have an advantage for this problem by means of capturing spatial patterns. In this study, two spatial dependent models are explored and they both have great ability to capture spatial patterns. However, the two models have both advantages and disadvantages with regard to the number of parameters and computational time. The auto-logistic models have parameters explaining every pair so they give the most exact solutions in this comparison. On the other hand, as the number of points considered increases, model parameter estimation become intractable. For example, [*Hughes and Guttorp*, 1994] uses EM algorithm which can estimate transition matrix and emission matrix separately. Hughes also explores several parameter estimation methods for spatial models using MCMC and pseudo-likelihood, and discusses the limitation of maximum likelihood for a large network. In our case study, several runs were needed to find a feasible solution for auto-logistic models. The Chow-Liu tree (CLT) models have a different characteristic, and have a lower number of parameters than the auto-logistic models. The advantage of CLT models is to have own parameter estimation methods (Section 2.2.3), which makes much faster and stable to estimate dependent parameters than using a numerical optimization method for auto-logistic models. Obviously, the spatial correlation cannot be preserved like auto-logistic models but the result at [Figure 14][Figure 15] shows Chow-Liu tree models are adequate enough. The simulations for rainfall amount seem well generated by simple rainfall distribution for entire periods, but the distribution cannot capture rainfall amount enough for each years.

In conclusion, from this exploration, independent models with NHMM are recommended for very small network and maximum likelihood method is the most practical. If there are many points for downscaling, Chow-Liu tree models with the EM algorithm are recommended. If more exact solutions are needed, Auto-logistic models can be employed. However, as mentioned before, if many points are considered, The EM algorithm should be used to estimate parameters separately and global optimization methods should be used for emission matrix. A problem which the NHMM approach shares with other Markov models is capturing the rainfall amount for each year or month when simulated daily rainfall is aggregated. One potential future solution to this problem may be either by combining occurrence models with amount modes or by improving occurrence models alone.

REFERENCES

Bardossy, A., and E. J. Plate (1992), Space-time model for daily rainfall using atmospheric circulation patterns, *Water Resources Research*, *28*(5), 1247-1259.

Baum, L. E., T. Petrie, G. Soules, and N. Weiss (1970), A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains, *The Annals of Mathematical Statistics*, *41*(1), 164-171.

Besag, J. (1974), Spatial interaction and the statistical analysis of lattice systems, *Journal of the Royal Statistical Society. Series B (Methodological)*, 36, 192-236.

Besag, J. (1975), Statistical analysis of non-lattice data, *The Statistician*, 24, 179-195.

Bürger, G. (1996), Expanded downscaling for generating local weather scenarios, *Climate Research*, *7*, 111-128.

Chow, C., and C. Liu (1968), Approximating discrete probability distributions with dependence trees, *IEEE Transactions on Information Theory*, *14*(3), 462-467.

Crowley, T. (1990), Are there any satisfactory geologic analogs for a future greenhouse warming? *Journal of Climate*, *3*(11), 1282-1292.

Dempster, A. P., N. M. Laird, and D. B. Rubin (1977), Maximum likelihood from incomplete data via the EM algorithm, *Journal of the Royal Statistical Society. Series B (Methodological)*, 39, 1-38.

Flohn, H. (1979), Can climate history repeat itself? Possible climatic warming and the case of paleoclimatic warm phases, *Man's Impact on Climate. edited by W. Bach, Y. Pankrath, and W.W. Kellogg, Elsevier Scientific Publishing, Amsterdam,* 957-966.

Flohn, H., and R. Fantechi (1984), *The Climate of Europe, Past, Present, and Future: Natural and Man-Induced Climatic Changes, a European Perspective*, Springer, Dordrecht, 356.

Gabriel, K. R., and J. Neumann (1962), A Markov chain model for daily rainfall occurrence at Tel Aviv, *Quarterly Journal of the Royal Meteorological Society*, *88*(375), 90-95.

Giorgi, F., and L. O. Mearns (1991), Approaches to the simulation of regional climate change: a review, *Reviews of Geophysics*, *29*(2), 191-216.

Hay, L. E., G. J. McCabe Jr, D. M. Wolock, and M. A. Ayers (1991), Simulation of precipitation by weather type analysis, *Water Resources Research*, *27*(4), 493-501.

Hewitson, B. C., and R. G. Crane (1992a), Large scale atmospheric controls on local precipitation in tropical Mexico, *Geophysical Research Letters*, *19*(18), 1835-1838.

Hewitson, B. C., and R. G. Crane (1992b), Regional-scale climate prediction from the GISS GCM, *Global and Planetary Change*, *5*(3), 249-267.

Hewitson, B. C., and R. G. Crane (1996), Climate downscaling: techniques and application, *Climate Research*, *7*, 85-95.

Hughes, J. P., and P. Guttorp (1994), A class of stochastic models for relating synoptic atmospheric patterns to regional hydrologic phenomena, *Water Resources Research*, *30*(5), 1535-1546.

Hughes, J. P., D. P. Lettenmaier, and P. Guttorp (1993), A stochastic approach for assessing the effect of changes in synoptic circulation patterns on gauge precipitation, *Water Resources Research*, *29*(10), 3303-3315.

Hughes, J. P., P. Guttorp, and S. P. Charles (1999), A non-homogeneous hidden Markov model for precipitation occurrence, *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, *48*(1), 15-30.

Khalil, A. F., H. H. Kwon, U. Lall, and Y. H. Kaheil (2010), Predictive downscaling based on non-homogeneous hidden Markov models, *Hydrological Sciences Journal–Journal des Sciences Hydrologiques*, *55*(3), 333-350.

Kirshner, S., P. Smyth, and A. W. Robertson (2004), Conditional Chow-Liu Tree Structures for Modeling Discrete-valued Vector Time Series, *AUAI Press, Arlington, Virginia*, 317- 324.

Kwon, H. H., B. Sivakumar, Y. I. Moon, and B. S. Kim (2011), Assessment of change in design flood frequency under climate change using a multivariate downscaling model and a precipitation-runoff model, *Stochastic Environmental Research and Risk Assessment*, *25*(4), 567-581.

Rabiner, L., and B. Juang (1986), An introduction to hidden Markov models, *ASSP Magazine, IEEE*, *3*(1), 4-16.

Richardson, C. W., and D. A. Wright (1984), WGEN: A Model for Generating Daily Weather Variables By, *Agriculture, USDA-ARS*.

Robertson, A. W., S. Kirshner, and P. Smyth (2003), Hidden Markov models for modeling daily rainfall occurrence over Brazil, *Information and Computer Science, University of California, Irvine, Tech. Rep. UCIICS*, 03-27.

Robertson, A. W., S. Kirshner, and P. Smyth (2004), Downscaling of daily rainfall occurrence over northeast Brazil using a hidden Markov model, *Journal of Climate*, *17*(22), 4407-4424.

Stern, R. D., and R. Coe (1984), A model fitting analysis of daily rainfall data, *Journal of the Royal Statistical Society. Series A (General)*, 147(1), 1-34.

Ugray, Z., L. Lasdon, J. Plummer, F. Glover, J. Kelly, and R. Martí (2006), Scatter search and local NLP solvers: A multistart framework for global optimization, *McCombs Research Paper Series,* IROM-07-06.

Wilby, R. L., and T. M. L. Wigley (1997), Downscaling general circulation model output: a review of methods and limitations, *Progress in Physical Geography*, *21*(4), 530.

Zucchini, W., and P. Guttorp (1991), A hidden Markov model for space-time precipitation, *Water Resources Research*, *27*(8), 1917-1923.

APPENDIX A
[Matlab Code for NHMM]

```
clear all
close all
set(0, 'DefaultAxesFontName', 'Times New Roman','defaulttextfontsize',5)
%% Load Data

NumTimeValueR = 3; % Time Data Length [Y M D];
NumTimeValueW = 3; % Time Data Length [Y M D];
consideryear = [1958 1990; 1991 2000]; %[Parameter Training(from to); Prediction(from to)]

% Point Data (R)
filename = 'Texas_Precipitation_Month\Texasdata';

AlldataR = dlmread(filename);



% Weather Data (X)
filenameW ='RWdata\SOI.sigextract2.txt'; % ENSO
load RWdata\SST_data.mat % SST
filenameW2 ='RWdata\GCM_P_Daily_data'; % Large Scale Precipitation

AlldataW2 = dlmread(filenameW2); % Large Scale Precipitation
AlldataW = dlmread(filenameW); %SOI or ENSO [year month day data]

AlldataW=[AlldataW(AlldataW(:,1)>=consideryear(1,1)&AlldataW(:,1)<=consideryear(2,2),1:end),...
          mean(Final(Final(:,1)>=consideryear(1,1)&Final(:,1)<=consideryear(2,2),3:end),2),...
          AlldataW2(AlldataW2(:,1)>=consideryear(1,1)&AlldataW2(:,1)<=consideryear(2,2),4)];

%%



rowsizeR = size(AlldataR,2);
rowsizeW = size(AlldataW,2);
Wdim = rowsizeW-NumTimeValueW; % dimention of weather data
numberofsites = (rowsizeR-NumTimeValueR)/2; % number of rainfall gauge stations



AlldataR = AlldataR(AlldataR(:,1)>=consideryear(1,1) & AlldataR(:,1)<=consideryear(2,2),:);
AlldataW = AlldataW(AlldataW(:,1)>=consideryear(1,1) & AlldataW(:,1)<=consideryear(2,2),:);



considermonth = [([2])' ([2])']; % [From Feb. to Feb.]

% Data for Parameter Estimation
ParameterdataR = AlldataR(AlldataR(:,1)>=consideryear(1,1) & AlldataR(:,1)<=consideryear(1,2),:);
ParameterdataW = AlldataW(AlldataW(:,1)>=consideryear(1,1) & AlldataW(:,1)<=consideryear(1,2),:);

% Data for Validation
ModeldataR = AlldataR(AlldataR(:,1)>=consideryear(2,1) & AlldataR(:,1)<=consideryear(2,2),:);
ModeldataW = AlldataW(AlldataW(:,1)>=consideryear(2,1) & AlldataW(:,1)<=consideryear(2,2),:);
```

```
L = size(ParameterdataR,1); %toal length of data for parameter estimation

startdate=datenum(ParameterdataR(1,1:3)); %start date of data for parameter estimation
enddate=datenum(ParameterdataR(L,1:3)); %end date of data for parameter estimation



%% Option

numberofmonth = size(considermonth,1); % number of groups of months in a year

colsize = 1; % every parameters reshape to vector for optimization

spatialmodeltype =  3; % (1) independent (2) dependent (autologistic) (3) Chow-Liu

Thhold = 0;
Statesize = 7; % Number of State
%%
% Data Configuration
% Wx : Weather data{month,year}
% Rs : Rainfall Sequence
% Rr : Rainfall Amount
for m = 1:numberofmonth
   for y = consideryear(1,1):consideryear(1,2)
       Wx{m,y-consideryear(1,1)+1} =
ParameterdataW(and(ParameterdataW(:,1)==y,ParameterdataW(:,2)>=considermonth(m,1)&Parameterdata
W(:,2)<=considermonth(m,2)),NumTimeValueW+1:rowsizeW)';
       Rs{m,y-consideryear(1,1)+1} =
ParameterdataR(and(ParameterdataR(:,1)==y,ParameterdataR(:,2)>=considermonth(m,1)&ParameterdataR(:,
2)<=considermonth(m,2)),NumTimeValueR+numberofsites+1:NumTimeValueR+numberofsites*2)';
       Rr{m,y-consideryear(1,1)+1} =
ParameterdataR(and(ParameterdataR(:,1)==y,ParameterdataR(:,2)>=considermonth(m,1)&ParameterdataR(:,
2)<=considermonth(m,2)),NumTimeValueR+1:NumTimeValueR+numberofsites)';
   end
end

% set option
Prim_Cond = [Statesize Wdim spatialmodeltype numberofsites];

initial = false; independency = false; em = 0;
Scon_Cond = [initial independency 0 em];



disp(sprintf('--------------------------------------------'));
%% Generate Equations into M file function
disp(sprintf('Formulation...'));

MultivariateNorm(Prim_Cond);

disp(sprintf('Formulation...done'));
%% Parameter Estimation
disp(sprintf('Start Maximum Likelihood..'));
```

```
for i = 1:numberofmonth
    a{i} = ones(Statesize,Statesize)/Statesize;
    b{i} = zeros(Wdim,Statesize,Statesize);

    Alpha{i} = ones(Statesize,numberofsites)*0.5;

    if spatialmodeltype == 1
        RealBeta{i} = [];
    elseif spatialmodeltype == 2
        RealBeta{i} = zeros(Statesize,sum(0:numberofsites-1));
    elseif spatialmodeltype == 3
        RealBeta{i} = zeros(numberofsites-1,6,Statesize);
    end
end



for m = 1:numberofmonth
%% Estimate HMM parameters with independent models
    disp(sprintf('------------------------------------------Initial'));
    %option
    initial = true; independency = true; em = 0;
    Scon_Cond = [initial independency 0 em];
    %create initial parameters and boundary
    [initialvalue lb ub Con_size] =...
        NHMM_Boundary(a{m},b{m},Alpha{m},RealBeta{m},Prim_Cond,Scon_Cond);
    [cons conss] = NHMMconstraints(Prim_Cond,Scon_Cond);

    % likelihood function HMM
    Fun = @(x) MCMLinitial(x,m,Rs,Prim_Cond,Scon_Cond);

    t = cputime;



    % find global optimal point
    globalopt = 4;

    if globalopt == 1 %MultiStart Optimization
        otime = cputime;
        numberofmulti = 100; %the number of start point
        A = [];
        B = [];
        Aeq = cons;
        Beq = conss;
        MaxFunEvals_Data = 10000;
        MaxIter_Data = 10000;
        [parameters1(:,:,m),fval,exitflag,output,solutions] =...
            MultiSQP(Fun...
            ,initialvalue,A,B,Aeq,Beq,lb,ub,MaxFunEvals_Data,MaxIter_Data,numberofmulti);
        disp(sprintf('[MLT][m=%g] fval=%g', m,fval));
        otime = cputime - otime;
        disp(num2str(otime));
    elseif globalopt == 2 %Global Search Optimization
        otime = cputime;
        numberofmulti = [];
        A = [];
        B = [];
        Aeq = cons;
        Beq = conss;
        MaxFunEvals_Data = 10000;
        MaxIter_Data = 10000;
        [parameters1(:,:,m),fval,exitflag,output,solutions] =...
            GSSQP(Fun...
            ,initialvalue,A,B,Aeq,Beq,lb,ub,MaxFunEvals_Data,MaxIter_Data,numberofmulti);
        disp(sprintf('[GS][m=%g] fval=%g', m,fval));
        otime = cputime - otime;
        disp(num2str(otime));
    elseif globalopt == 3 %GA
        otime = cputime;
        nvars = size(initialvalue,2);
        A = [];
        B = [];
        Aeq = cons;
        Beq = conss;

        PopulationSize_Data = nvars*4;
        EliteCount_Data = 2;
        CrossoverFraction_Data=0.5;
        MigrationInterval_Data= 20;
        MigrationFraction_Data= 0.2;
        Generations_Data = 100;
        InitialPopulation_Data = initialvalue;
        PopInitRange_Data = [0;1];
        Tournamentsize = 4;
        Mutationrate = 0.01;

        Gadatum =
[PopulationSize_Data,EliteCount_Data,CrossoverFraction_Data,MigrationInterval_Data,MigrationFraction_
Data,Generations_Data,Tournamentsize,Mutationrate];



        [parameters1(:,:,m) fval] =...
            NHMMgaBit(Fun...
            ,nvars,A,B,Aeq,Beq,lb,ub,PopInitRange_Data,InitialPopulation_Data,Gadatum);
        disp(sprintf('[GA][m=%g] fval=%g', m,fval));
        otime = cputime - otime;
        disp(num2str(otime));
    elseif globalopt == 4 %SA
        otime = cputime;
        [parameters1(:,:,m),fval,exitflag,output] =...
            annealing(Fun,initialvalue,lb,ub,1e-6,inf,1000);
        disp(sprintf('[Annealing][m=%g] fval=%g', m,fval));
        otime = cputime - otime;
        disp(num2str(otime));
    elseif globalopt == 5 %MCMC
        otime = cputime;
        iter = 20000;
        ng = 100000;
        sig = 0.004;
        %acceptance rate [1% with 0.03] [3% with 0.02] [13% with 0.01] [35%
        % with 0.005] [81% with 0.001] [52% with 0.0035] [55% with 0.004]
        [itervals r rpercents] = MCMC(Fun,initialvalue,Con_size,iter,ng,sig);
        otime = cputime - otime;
        disp(num2str(otime));
    elseif globalopt == 6 %EM
        [guessTR,b1,guessE,beta1] = ParameterDecode(initialvalue, ...
            Con_size,Prim_Cond,Scon_Cond);
        seqs = Rs;
        parfor i = 1:100
            guessTR = rand(Statesize,Statesize);
            guessTR = guessTR./(guessTR*ones(Statesize,Statesize));
            guessE = rand(Statesize,numberofsites);
            [TR{i},E{i},logliks] = nhmmtrain(seqs,guessTR,guessE,500);
            loglk(i) = logliks(end);
        end
        maxv = find(loglk==max(loglk));
        parameters1(:,:,m) = [reshape(TR{maxv(1)},1,[]) reshape(E{maxv(1)},1,[])];
        disp(sprintf('[EM][m=%g] fval=%g', m,loglk(maxv(1))));
    else
        error('There is no such optimization tool');
    end



    sqpactive = true;
    if sqpactive == false % if we want to jump other procedures
            [a{m},b{m},Alpha{m},RealBeta{m}] = ParameterDecode(parameters1(:,:,m), ...
                Con_size,Prim_Cond,Scon_Cond);
    else
    % numerical optimiation (SQP)
        [a1,b1,alpha1,beta1] = ParameterDecode(parameters1(:,:,m), ...
            Con_size,Prim_Cond,Scon_Cond); % vector to matrix
        a1 = a1./(a1*ones(Statesize,Statesize)); % normalize
        [parameters1(:,:,m) lb ub Con_size] =...
            NHMM_Boundary(a1,b1,alpha1,beta1,Prim_Cond,Scon_Cond);
        [cons conss] = NHMMconstraints(Prim_Cond,Scon_Cond);
        A = [];
        B = [];
        Aeq = cons;
        Beq = conss;
        MaxFunEvals_Data = 10000;
        MaxIter_Data = 10000;
        [parameters2(:,:,m),fval,exitflag,output,lambda,grad,hessian] =...
            SQP2(Fun...
            ,parameters1(:,:,m),A,B,Aeq,Beq,lb,ub,MaxFunEvals_Data,MaxIter_Data);
        e = cputime-t;
        disp(sprintf('[SQP][m=%g] fval=%g t=%g', m,fval,e));



        [a2,b2,alpha2,beta2] = ParameterDecode(parameters2(:,:,m), ...
            Con_size,Prim_Cond,Scon_Cond);

        % transform into auto-logistic parameters
        buffer = log(a2./(1-a2));
        buffer(buffer>50) = 50;
        buffer(buffer<-50) = -50;
```

```
        a2 = buffer;

        Em_active = false;

        % transform into auto-logistic parameters
        if spatialmodeltype == 3 || Em_active == true
           [a{m},b{m},Alpha{m},RealBeta{m}] =...
              ParameterDecode(parameters2(:,:,m),Con_size,Prim_Cond,Scon_Cond);
           buffer = log(a{m}./(1-a{m}));
           buffer(buffer>50) = 50;
           buffer(buffer<-50) = -50;
           a{m} = buffer;
           break
        end



        %% Estimate NHMM parameters with independent models
        disp(sprintf('--------------------------------------------Independent'));

        initial = false; independency = true; em = 0;
        Scon_Cond = [initial independency 0 em];



        if spatialmodeltype == 2
                buffer = log(alpha2./(1-alpha2));
                buffer(buffer>100) = 50;
                buffer(buffer<-100) = -50;
                alpha2 = buffer;
        end

        [x0 lb ub Con_size] =...
           NHMM_Boundary(a2,b{m},alpha2,RealBeta{m},...
           Prim_Cond,Scon_Cond);
        [cons conss] = NHMMconstraints(Prim_Cond,Scon_Cond);

        A = [];
        B = [];
        Aeq = [];
        Beq = [];
        MaxFunEvals_Data = 10000;
        MaxIter_Data = 10000;
        [parameters3(:,:,m),fval,exitflag,output] =...
           SQP2(@(x)
MCML(x,m,Rs,Wx,Con_size,Prim_Cond,Scon_Cond),x0,A,B,Aeq,Beq,lb,ub,MaxFunEvals_Data,MaxIter_
Data);
        disp(sprintf('[SQP][m=%g] fval=%g', m,fval));

        [a{m},b{m},Alpha{m},b3] =...
           ParameterDecode(parameters3(:,:,m),Con_size,Prim_Cond,Scon_Cond);



    %% Estimate NHMM parameters with dependent models if this is dependent models
        disp(sprintf('--------------------------------------------Dependent'));
        initial = false; independency = false; em = 0;
        Scon_Cond = [initial independency 0 em];

        if spatialmodeltype == 2
           [Fixed_Parameter lb ub Con_size] =...
                 NHMM_Boundary(a{m},b{m},Alpha{m},RealBeta{m},...
                 Prim_Cond,Scon_Cond);
           [cons conss] = NHMMconstraints(Prim_Cond,Scon_Cond);

           A = [];
           B = [];
           Aeq = [];
           Beq = [];
           MaxFunEvals_Data = 10000;
           MaxIter_Data = 1000;
           [parameters4(:,:,m) fval] =...
              SQP2(@(x) MCML(x,m,Rs,Wx,Con_size,Prim_Cond,Scon_Cond),...
           Fixed_Parameter,A,B,Aeq,Beq,lb,ub,MaxFunEvals_Data,MaxIter_Data);

           disp(sprintf('[SQP][m=%g] fval=%g', m,fval));



           [a{m},b{m},Alpha{m},RealBeta{m}] =...
              ParameterDecode(parameters4(:,:,m), ...
              Con_size,Prim_Cond,Scon_Cond);


           disp(sprintf('[MCMLsingle] %g month was done...%g', m,fval));%2116.82
        end
    end
end

% Auto-Logistic parameters
m=1;
if spatialmodeltype==2
for s = 1:Statesize
    loc = triu(ones(numberofsites),1);
    loc(loc==1)=RealBeta{m}(s,:);
    beta(:,:,s) = loc;
    clear loc
end
end
disp(sprintf('--------------------------------------------'));

%
initial = false; independency = false; em = 0;
Scon_Cond = [initial independency 0 em];

% EM algorithm
if Em_active == true || spatialmodeltype == 3
    for m=1:numberofmonth
```

```
        disp(sprintf('------------------------------------------'));
        [guessa2{m},guessb2{m},guessAlpha2{m},guessBeta2{m}...
            ,logliks2{m},parameters5{m}] = ...
            EMalgorithm(m,Rs,Wx,a{m},b{m},Alpha{m},RealBeta{m},Prim_Cond,Scon_Cond,...
            1e-6,500);
        disp(sprintf('[EM MCEMsingle] %g month was done...', m));
    end
    guessa = guessa2; guessb = guessb2;
    guessAlpha = guessAlpha2; guessBeta = guessBeta2;
else
    guessa = a; guessb = b;
    guessAlpha = Alpha; guessBeta = RealBeta;
end

%%
% Rainfall data to structure
Scon_Cond(2)=true;
for m = 1:numberofmonth
    count = zeros(1,Statesize);
    DataTable2{m} = [];
    for y = 1:year(enddate)-year(startdate)+1
        mcurrentstate = 1;
        [currentState{m,y}]...
            = MCMEviterbi(Wx{m,y},Rs{m,y},guessa{m},guessb{m},guessAlpha{m},guessBeta{m},...
            mcurrentstate,Prim_Cond,Scon_Cond);
        for t = 1:size(currentState{m,y},2)
            s = currentState{m,y}(1,t);
            count(1,s) = count(1,s) + 1;
            DataTable{m,s}(count(1,s),:) = Rr{m,y}(:,t)';
        end
        DataTable2{m} = [DataTable2{m};Rr{m,y}'];
    end
end

% select distribution for rainfall amount
pmodel = 4;
if pmodel == 1; %exponential
    Fun = @(lamda,y) exppdf(y,lamda(1));
    cumFun = @(lamda,p) expcdf(p,lamda(1));
    lb=[exp(-100)];ub=[20];
    ini = [0.5];
elseif pmodel == 2; %gamma
    Fun = @(lamda,y) gampdf(y,lamda(1),lamda(2));
    cumFun = @(lamda,p) gamcdf(p,lamda(1),lamda(2));
    lb=[exp(-100) exp(-100)];ub=[20 20];
    ini = [1 2];
elseif pmodel == 3; %mixed exponential
    Fun = @(lamda,y) lamda(1)*exppdf(y,lamda(2))+(1-lamda(1))*exppdf(y,lamda(3));
    lb=[exp(-100) exp(-100) exp(-100)];ub=[1 20 20];
    ini = [0.3 3 4];
elseif pmodel == 4;%weibull
    Fun = @(lamda,y) wblpdf(y,lamda(1),lamda(2));
    cumFun = @(lamda,p) wblcdf(p,lamda(1),lamda(2));
    lb=[exp(-100) exp(-100)];ub=[20 20];
    ini = [1 1];
```

```
end

options = optimset;
options = optimset(options,'Display', 'off');
options = optimset(options,'MaxFunEvals', 1000);
options = optimset(options,'MaxIter', 1000);
options = optimset(options,'Algorithm', 'sqp');
options = optimset(options,'FinDiffType', 'central');

% parameter estimation for rainfall amount
fpa = [];
loglikelihood = [];
for sites = 1:numberofsites
   for m = 1:numberofmonth
      rdata2 = DataTable2{m}(DataTable2{m}(:,sites)>Thhold,sites)';
      numberofdata = size(rdata2,2);
      [fpa{sites,m} loglikelihood{sites,m}]=fmincon(@(parameters)
Objectivefun(parameters,Fun,[],rdata2),ini,[],[],[],[],lb,ub,[],options);
   end
end

%% Sampling
disp(sprintf('--------------------------------------sampling'));

numberofsampling = 1000;
per = 0.95; % percentile
uq=floor(numberofsampling*(per));
lq=ceil(numberofsampling*(1-per));
Pnumberofyear = consideryear(1,2)-consideryear(1,1)+1; % The number of year for training period
Mnumberofyear = consideryear(2,2)-consideryear(2,1)+1; % The number of year for predictive period

eachmonthlength = considermonth(:,2)-considermonth(:,1)+1;
monthlength  = sum(eachmonthlength);

% Rainfall occurrence
PRline = zeros(numberofsites,Pnumberofyear*monthlength,numberofsampling); % Data rainfall occurrence
in training period per month
PEline = zeros(numberofsites,Pnumberofyear*monthlength,numberofsampling); % Simulated rainfall
occurrence in training period per month
PRliney = zeros(numberofsites,Pnumberofyear,numberofsampling); % Data rainfall occurrence in training
period per year
PEliney = zeros(numberofsites,Pnumberofyear,numberofsampling); % Simulated rainfall occurrence in
training period per year
MRline = zeros(numberofsites,Mnumberofyear*monthlength,numberofsampling); % Data rainfall
occurrence in predictive period per month
MEline = zeros(numberofsites,Mnumberofyear*monthlength,numberofsampling); % Simulated rainfall
occurrence in predictive period per month
MRliney = zeros(numberofsites,Mnumberofyear,numberofsampling); % Data rainfall occurrence in
predictive period per year
MEliney = zeros(numberofsites,Mnumberofyear,numberofsampling); % Simulated rainfall occurrence in
predictive period per year
Pre = zeros(2,3,numberofsites,numberofsampling); % [Total number of occurrence][Average][Std] in
training period
Mod = zeros(2,3,numberofsites,numberofsampling); % [Total number of occurrence][Average][Std] in
```

```
predictive period

% Rainfall amount
PRline2 = zeros(numberofsites,Pnumberofyear*monthlength,numberofsampling);
PEline2 = zeros(numberofsites,Pnumberofyear*monthlength,numberofsampling);
PRliney2 = zeros(numberofsites,Pnumberofyear,numberofsampling);
PEliney2 = zeros(numberofsites,Pnumberofyear,numberofsampling);
MRline2 = zeros(numberofsites,Mnumberofyear*monthlength,numberofsampling);
MEline2 = zeros(numberofsites,Mnumberofyear*monthlength,numberofsampling);
MRliney2 = zeros(numberofsites,Mnumberofyear,numberofsampling);
MEliney2 = zeros(numberofsites,Mnumberofyear,numberofsampling);
Pre2 = zeros(2,3,numberofsites,numberofsampling);
Mod2 = zeros(2,3,numberofsites,numberofsampling);

P1 = zeros(numberofsites,100,numberofsampling); % Data Survival Function in training period
P2 = zeros(numberofsites,100,numberofsampling); % Simulated Survival Function in training period
M1 = zeros(numberofsites,100,numberofsampling); % Data Survival Function in predictive period
M2 = zeros(numberofsites,100,numberofsampling); % Simulated Survival Function in predictive period
PRcorr = zeros(numberofsites,numberofsites,numberofsampling); % Data spatial correlation in training
period
MRcorr = zeros(numberofsites,numberofsites,numberofsampling); % Data spatial correlation in predictive
period
PScorr = zeros(numberofsites,numberofsites,numberofsampling); % Simulated spatial correlation in training
period
MScorr = zeros(numberofsites,numberofsites,numberofsampling); % Simulated spatial correlation in
predictive period

PR2 = [];MR2 = [];PS2 = [];MS2 = [];
PRr2{numberofsampling} = [];MRr2{numberofsampling} = [];PSr2{numberofsampling} =
[];MSr2{numberofsampling} = [];
PR = [];MR = [];PS = [];MS = [];
PRr{numberofsampling} = [];MRr{numberofsampling} = [];PSr{numberofsampling} =
[];MSr{numberofsampling} = [];
parfor sampling = 1:numberofsampling
   [...
      PRline(:,:,sampling) PEline(:,:,sampling) PRliney(:,:,sampling) PEliney(:,:,sampling)...
      MRline(:,:,sampling) MEline(:,:,sampling) MRliney(:,:,sampling) MEliney(:,:,sampling)...
      Pre(:,:,:,sampling) Mod(:,:,:,sampling)...
      PRline2(:,:,sampling) PEline2(:,:,sampling) PRliney2(:,:,sampling) PEliney2(:,:,sampling)...
      MRline2(:,:,sampling) MEline2(:,:,sampling) MRliney2(:,:,sampling) MEliney2(:,:,sampling)...
      Pre2(:,:,:,sampling) Mod2(:,:,:,sampling)...
      P1(:,:,sampling) P2(:,:,sampling) M1(:,:,sampling) M2(:,:,sampling)...
      PRr2{sampling} MRr2{sampling} PSr2{sampling} MSr2{sampling}...
      PRr{sampling} MRr{sampling} PSr{sampling} MSr{sampling} Ramount(:,:,:,sampling)
PRamount(:,:,:,sampling) MRamount(:,:,:,sampling)...
      PRcorr(:,:,sampling) MRcorr(:,:,sampling) PScorr(:,:,sampling) MScorr(:,:,sampling)...
      PRbin(:,:,sampling) MRbin(:,:,sampling) PSbin(:,:,sampling) MSbin(:,:,sampling)...
   ] =
NHMMsampling(ParameterdataW,ModeldataW,ParameterdataR,ModeldataR,NumTimeValueR,NumTimeV
alueW,rowsizeW,rowsizeR,...
      consideryear,considermonth,Statesize,numberofsites,spatialmodeltype,independency,initial,...
      guessa,guessb,guessAlpha,guessBeta,Thhold,fpa,pmodel);
end
```

```
final = {sort(PRline,3) sort(MRline,3) sort(PRliney,3) sort(MRliney,3);...
        sort(PEline,3) sort(MEline,3) sort(PEliney,3) sort(MEliney,3)};
finalmean = {median(final{1,1},3) median(final{1,2},3) median(final{1,3},3) median(final{1,4},3);...
            median(final{2,1},3) median(final{2,2},3) median(final{2,3},3) median(final{2,4},3)};
final2 = {sort(PRline2,3) sort(MRline2,3) sort(PRliney2,3) sort(MRliney2,3);...
         sort(PEline2,3) sort(MEline2,3) sort(PEliney2,3) sort(MEliney2,3)};
finalmean2 = {median(final2{1,1},3) median(final2{1,2},3) median(final2{1,3},3) median(final2{1,4},3);...
            median(final2{2,1},3) median(final2{2,2},3) median(final2{2,3},3) median(final2{2,4},3)};
Survivalmean = {median(P1,3) median(M1,3);
               median(P2,3) median(M2,3)};
Survival = {sort(P1,3) sort(M1,3);
           sort(P2,3) sort(M2,3)};
endloc = find(sum([Survivalmean{1,1};Survivalmean{1,2};Survivalmean{2,1};Survivalmean{2,2}],1)~=0);
for l = 1:2
    for k = 1:2
        Survivalmean{l,k}(:,endloc(end)+1:end) = [];
        Survival{l,k}(:,endloc(end)+1:end,:) = [];
    end
end
%% Get datenumber

countp = 0;
countm = 0;
for y = consideryear(1,1):consideryear(1,2)
    for cm = 1:numberofmonth
        for m = considermonth(cm,1):considermonth(cm,2)
            countp = countp+1;
            Pmx(1,countp) = datenum(y,m,1);
        end
    end
    Pyx(1,y-consideryear(1,1)+1) = datenum(y,1,1);
end
for y = consideryear(2,1):consideryear(2,2)
    for cm = 1:numberofmonth
        for m = considermonth(cm,1):considermonth(cm,2)
            countm = countm+1;
            Mmx(1,countm) = datenum(y,m,1);
        end
    end
    Myx(1,y-consideryear(2,1)+1) = datenum(y,1,1);
end

%% Plot Simulation
%
figh=figure(3);
for s = 1:numberofsites
    lin = [Pmx Mmx];
    xline = 1:size(lin,2);
    tic  = 1:ceil(size(lin,2)/7):size(lin,2);
    lin2 = lin(tic);

    subplot(numberofsites,1,s);
    ciplot([final{2,1}(s,:,lq) final{2,2}(s,:,lq)]...
          ,[final{2,1}(s,:,uq) final{2,2}(s,:,uq)]...
          ,xline,[1 0.93 0.90]);
    hold on
    plot(xline,[finalmean{2,1}(s,:) finalmean{2,2}(s,:)],'-','LineWidth',2,'Color','blue');
    plot(xline,[finalmean{1,1}(s,:) finalmean{1,2}(s,:)],'--','LineWidth',2,'Color','red');
    axis([1 size(lin,2) -5 20])
    legend('95% Interval','Prediction','Data','Location','NorthEastOutside');
    h=xlabel('(Year)','Fontsize',9);
    pos=get(h,'Position');
    pos(1)=pos(1)+25;
    pos(2)=pos(2)+9.5;
    ylabel('N','FontSize',9)
    grid on
    title(['Station ' num2str(s)],'FontSize',10,'FontWeight','bold')
    set(gca,'XLim',[1 size(lin,2)])
    set(gca,'XTick',tic)
    set(gca,'XTickLabel',[datestr(lin2,'mmm yyyy')],'Fontsize',9)
end
set(figh,'OuterPosition',[1 1 750 800])



figh=figure(4);
for s = 1:numberofsites
    lin = [Pyx Myx];
    xline = 1:size(lin,2);
    tic  = 1:ceil(size(lin,2)/7):size(lin,2);
    lin2 = lin(tic);

    subplot(numberofsites,1,s);
    ciplot([final{2,3}(s,:,lq) final{2,4}(s,:,lq)]...
          ,[final{2,3}(s,:,uq) final{2,4}(s,:,uq)]...
          ,xline,[1 0.93 0.90]);
    hold on
    plot(xline,[finalmean{2,3}(s,:) finalmean{2,4}(s,:)],'-','LineWidth',2,'Color','blue');
    plot(xline,[finalmean{1,3}(s,:) finalmean{1,4}(s,:)],'--','LineWidth',2,'Color','red');
    plot([33.5;33.5],[0;15],'--bs','LineWidth',2);
    axis([1 size(lin,2) -5 20])
    legend('95% Interval','Simulation','Observation','Location','NorthEastOutside');
%     LEGEND BOXOFF
    h=xlabel('(Year)','Fontsize',9);
    pos=get(h,'Position');
    pos(1)=pos(1)+25;
    pos(2)=pos(2)+9.5;
    set(h,'Position',pos)
    ylabel('N','Fontsize',9)
    grid on
    title(['Station ' num2str(s)],'FontSize',10,'FontWeight','bold');
    set(gca,'XLim',[1 size(lin,2)])
    set(gca,'XTick',tic)
    set(gca,'XTickLabel',[datestr(lin2,'yyyy')],'Fontsize',9)
end
set(figh,'OuterPosition',[1 1 600 800])



figh=figure(5);
for s = 1:numberofsites
    lin = [Pmx Mmx];
    xline = 1:size(lin,2);
    tic  = 1:ceil(size(lin,2)/7):size(lin,2);
    lin2 = lin(tic);
    subplot(numberofsites,1,s);
    ciplot([final2{2,1}(s,:,lq) final2{2,2}(s,:,lq)]...
        ,[final2{2,1}(s,:,uq) final2{2,2}(s,:,uq)]...
        ,xline,[1 0.93 0.90]);
    hold on
    plot(xline,[finalmean2{2,1}(s,:) finalmean2{2,2}(s,:)],'-','LineWidth',2,'Color','blue');
    plot(xline,[finalmean2{1,1}(s,:) finalmean2{1,2}(s,:)],'--','LineWidth',2,'Color','red');
    axis([1 size(lin,2) -1 8])
    legend('95% Interval','Prediction','Data','Location','NorthEastOutside');
    h=xlabel('(Year)','Fontsize',9);
    pos=get(h,'Position');
    pos(1)=pos(1)+25;
    pos(2)=pos(2)+9.5;
    ylabel('P(inch)','FontSize',9)
    grid on
    title(['Station ' num2str(s)],'FontSize',10,'FontWeight','bold')
    set(gca,'XLim',[1 size(lin,2)])
    set(gca,'XTick',tic)
    set(gca,'XTickLabel',[datestr(lin2,'mmm yyyy')],'Fontsize',9)
end
set(figh,'OuterPosition',[1 1 750 800])



figh=figure(6);
for s = 1:numberofsites
    lin = [Pyx Myx];
    xline = 1:size(lin,2);
    tic  = 1:ceil(size(lin,2)/7):size(lin,2);
    lin2 = lin(tic);
    subplot(numberofsites,1,s);
    ciplot([final2{2,3}(s,:,lq) final2{2,4}(s,:,lq)]...
        ,[final2{2,3}(s,:,uq) final2{2,4}(s,:,uq)]...
        ,xline,[1 0.93 0.90]);
    hold on
    plot(xline,[finalmean2{2,3}(s,:) finalmean2{2,4}(s,:)],'-','LineWidth',2,'Color','blue');
    plot(xline,[finalmean2{1,3}(s,:) finalmean2{1,4}(s,:)],'--','LineWidth',2,'Color','red');
    axis([1 size(lin,2) -1 8])
    legend('95% Interval','Prediction','Data','Location','NorthEastOutside');
    h=xlabel('(Year)','Fontsize',9);
    pos=get(h,'Position');
    pos(1)=pos(1)+25;
    pos(2)=pos(2)+9.5;
    ylabel('P(inch)','FontSize',9)
    grid on
    title(['Station ' num2str(s)],'FontSize',10,'FontWeight','bold')
    set(gca,'XLim',[1 size(lin,2)])
    set(gca,'XTick',tic)
    set(gca,'XTickLabel',[datestr(lin2,'yyyy')],'Fontsize',9)
end
set(figh,'OuterPosition',[1 1 600 800])
%% plot survival function
figh=figure(7);
```

```
for s = 1:numberofsites
   subplot(ceil(numberofsites/2),2,s);
   ciplot(Survival{2,1}(s,:,lq)...
        ,Survival{2,1}(s,:,uq)...
        ,1:size(Survivalmean{1,1}(s,:),2),[0.95 0.93 0.90]);
   hold on
   plot(Survivalmean{2,1}(s,:),'-','LineWidth',2,'Color','blue');
   plot(Survivalmean{1,1}(s,:),'--','LineWidth',2,'Color','red');
   legend('95% Interval','Simulation','Observation','Location','NorthEast');
   h=xlabel('(days)','Fontsize',9);
   pos=get(h,'Position');
   pos(1)=pos(1)+25;
   pos(2)=pos(2)+9.5;
   ylabel('Survivor','Fontsize',9)
   grid on
   title(['Station ' num2str(s)],'FontSize',10,'FontWeight','bold')
   set(gca,'Fontsize',9)
end
set(figh,'OuterPosition',[1 1 600 800])

figh=figure(8);
for s = 1:numberofsites
   subplot(ceil(numberofsites/2),2,s);
   ciplot(Survival{2,2}(s,:,lq)...
        ,Survival{2,2}(s,:,uq)...
        ,1:size(Survivalmean{1,2}(s,:),2),[0.95 0.93 0.90]);
   hold on
   plot(Survivalmean{2,2}(s,:),'-','LineWidth',2,'Color','blue');
   plot(Survivalmean{1,2}(s,:),'--','LineWidth',2,'Color','red');
   legend('95% Interval','Prediction','Observation','Location','NorthEast');
   h=xlabel('(days)','Fontsize',9);
   pos=get(h,'Position');
   pos(1)=pos(1)+25;
   pos(2)=pos(2)+9.5;
   ylabel('Survivor','Fontsize',9)
   grid on
   title(['Station ' num2str(s)],'FontSize',10,'FontWeight','bold')
   set(gca,'Fontsize',9)
end
set(figh,'OuterPosition',[1 1 600 800])

%% plot summery statistic

figh=figure(9);
for s = 1:numberofsites
   subplot(ceil(numberofsites/2),2,s);
   xs{s}=[mean(Pre(1,1,s,:),4) mean(Pre(2,1,s,:),4);mean(Pre2(1,1,s,:),4) mean(Pre2(2,1,s,:),4)];
   xsa(s,:)=[mean(Pre(1,1,s,:),4) mean(Pre(2,1,s,:),4) mean(Pre2(1,1,s,:),4) mean(Pre2(2,1,s,:),4)];
   xsm(s,:)=[mean(Pre(1,2,s,:),4) mean(Pre(2,2,s,:),4) mean(Pre2(1,2,s,:),4) mean(Pre2(2,2,s,:),4)];
   xss(s,:)=[mean(Pre(1,3,s,:),4) mean(Pre(2,3,s,:),4) mean(Pre2(1,3,s,:),4) mean(Pre2(2,3,s,:),4)];
   x2=[reshape(Pre(2,1,s,:),[],1) reshape(Pre2(2,1,s,:),[],1)];
   hMulti = bar(1:2,xs{s});
   set(hMulti(1),'LineWidth',1,'FaceColor',[0 0 1])
   set(hMulti(2),'LineWidth',1,'FaceColor',[0.9 0.9 1])
   hold on
   boxplot(x2)
   axis([0.5 2.5 0 300])
   legend('Observation','Simulation','Location','NorthEast');
   ylabel('Total')
%    grid on
   title(['Total Count and Amount: Station ' num2str(s)],'FontSize',10,'FontWeight','bold')
   set(gca,'XLim',[0.5 2.5])
   set(gca,'XTick',[1 2])
   set(gca,'XTickLabel',['Count ';'Amount'],'FontSize',10)
end
set(figh,'OuterPosition',[1 1 750 800])

figh=figure(10);
for s = 1:numberofsites
   subplot(ceil(numberofsites/2),2,s);
   mxs{s}=[mean(Mod(1,1,s,:),4) mean(Mod(2,1,s,:),4);mean(Mod2(1,1,s,:),4) mean(Mod2(2,1,s,:),4)];
   mxsa(s,:)=[mean(Mod(1,1,s,:),4) mean(Mod(2,1,s,:),4) mean(Mod2(1,1,s,:),4) mean(Mod2(2,1,s,:),4)];
   mxsm(s,:)=[mean(Mod(1,2,s,:),4) mean(Mod(2,2,s,:),4) mean(Mod2(1,2,s,:),4) mean(Mod2(2,2,s,:),4)];
   mxss(s,:)=[mean(Mod(1,3,s,:),4) mean(Mod(2,3,s,:),4) mean(Mod2(1,3,s,:),4) mean(Mod2(2,3,s,:),4)];
   mx2=[reshape(Mod(2,1,s,:),[],1) reshape(Mod2(2,1,s,:),[],1)];
   hMulti = bar(1:2,mxs{s});
   set(hMulti(1),'LineWidth',1,'FaceColor',[0 0 1])
   set(hMulti(2),'LineWidth',1,'FaceColor',[0.9 0.9 1])
   hold on
   boxplot(mx2)
   axis([0.5 2.5 0 100])
   legend('Observation','Prediction','Location','NorthEast');
   ylabel('Total')
%    grid on
   title(['Total Count and Amount: Station ' num2str(s)],'FontSize',10,'FontWeight','bold')
   set(gca,'XLim',[0.5 2.5])
   set(gca,'XTick',[1 2])
   set(gca,'XTickLabel',['Count ';'Amount'],'FontSize',10)
end
set(figh,'OuterPosition',[1 1 750 800])

%% plot spatial correlation
clims = [0 1];
figh=figure(11);
subplot(4,2,1);imagesc(mean(PRcorr,3),clims);
colormap(hot);
colorbar
title(['Observation in training periods'],'FontSize',10,'FontWeight','bold')
% set(gca,'XLim',[1 numberofsites],'YLim',[1 numberofsites])
set(gca,'XTick',1:numberofsites,'FontSize',10,'YTick',1:numberofsites)

std(PScorr,0,3);
subplot(4,2,3);imagesc(mean(PScorr,3),clims);
colormap(hot);
colorbar
title(['Simulation in training periods'],'FontSize',10,'FontWeight','bold')
% set(gca,'XLim',[1 numberofsites],'YLim',[1 numberofsites])
set(gca,'XTick',1:numberofsites,'FontSize',10,'YTick',1:numberofsites)

std(PScorr,0,3);
```

```
subplot(4,2,5);imagesc(abs(mean(PRcorr,3)-mean(PScorr,3)),clims);
colormap(hot);
colorbar
title(['Differences in training periods'],'FontSize',10,'FontWeight','bold')
% set(gca,'XLim',[1 numberofsites],'YLim',[1 numberofsites])
set(gca,'XTick',1:numberofsites,'FontSize',10,'YTick',1:numberofsites)

subplot(4,2,2);imagesc(mean(MRcorr,3),clims);
colormap(hot);
colorbar
title(['Observation in predictive periods'],'FontSize',10,'FontWeight','bold')
% set(gca,'XLim',[1 numberofsites],'YLim',[1 numberofsites])
set(gca,'XTick',1:numberofsites,'FontSize',10,'YTick',1:numberofsites)

std(MScorr,0,3);
subplot(4,2,4);imagesc(mean(MScorr,3),clims);
colormap(hot);
colorbar
title(['Simulation in predictive periods'],'FontSize',10,'FontWeight','bold')
% set(gca,'XLim',[1 numberofsites],'YLim',[1 numberofsites])
set(gca,'XTick',1:numberofsites,'FontSize',10,'YTick',1:numberofsites)

std(PScorr,0,3);
subplot(4,2,6);imagesc(abs(mean(MRcorr,3)-mean(MScorr,3)),clims);
colormap(hot);
colorbar
title(['Differences in predictive periods'],'FontSize',10,'FontWeight','bold')
% set(gca,'XLim',[1 numberofsites],'YLim',[1 numberofsites])
set(gca,'XTick',1:numberofsites,'FontSize',10,'YTick',1:numberofsites)

% set(figh,'OuterPosition',[1 1 700 800])
%% plot spatial correlation
clims = [0 0.3];
% figh=figure(12);
subplot(4,2,7);imagesc(std(PScorr,0,3),clims);
colormap(hot);
colorbar
title(['Standard Deviation'],'FontSize',10,'FontWeight','bold')
% set(gca,'XLim',[1 numberofsites],'YLim',[1 numberofsites])
set(gca,'XTick',1:numberofsites,'FontSize',10,'YTick',1:numberofsites)

subplot(4,2,8);imagesc(std(MScorr,0,3),clims);
colormap(hot);
colorbar
title(['Standard Deviation'],'FontSize',10,'FontWeight','bold')
% set(gca,'XLim',[1 numberofsites],'YLim',[1 numberofsites])
set(gca,'XTick',1:numberofsites,'FontSize',10,'YTick',1:numberofsites)
set(figh,'OuterPosition',[1 1 600 1200])
```

```
function [] = MultivariateNorm(Prim_Cond)
Statesize = Prim_Cond(1);
Wdim = Prim_Cond(2);
```

```
spatialmodeltype = Prim_Cond(3);
numberofsites = Prim_Cond(4);
%% Multivariate Normal F(x a b)
x = sym('x',[Wdim 1]); % Weather data vector at time t

b_demo = sym('b',[Wdim Statesize*Statesize]); % average for x according to state changes
syms b; % (x, i, j)
b = reshape(b_demo,Wdim,Statesize,Statesize);

%% Transition Matrix (a)

a = sym('a',[Statesize Statesize]);
A = a./(a*ones(Statesize,Statesize));
%% (A0) Non-homogeneous Transition Matrix

onematrix = ones(Statesize,Statesize);
Ao(Statesize,Statesize) = sym(0);
for i = 1:Statesize
   for j = 1:Statesize
      Ao(i,j) = exp(a(i,j)+transpose(b(:,i,j))*x);
   end
end




Ao = Ao./(Ao*onematrix); % Normalize



%% Differentiation Ao and log(Ao)

Diff_A_b(Wdim,Statesize,Statesize,Statesize,Statesize) = sym(0);

for k = 1:Wdim
   for i = 1:Statesize
      for j = 1:Statesize
         for l = 1:Statesize
         for m = 1:Statesize
         Diff_A_b(k,i,j,l,m) = Diff_A_b(k,i,j,l,m)+diff(Ao(l,m),b(k,i,j),1);
         end
         end
      end
   end
end



Diff_A_h(Statesize,Statesize,Statesize,Statesize) = sym(0);
Diff_A_a(Statesize,Statesize,Statesize,Statesize) = sym(0);
for i = 1:Statesize
   for j = 1:Statesize
      for l = 1:Statesize
      for m = 1:Statesize
      Diff_A_a(i,j,l,m) = Diff_A_a(i,j,l,m)+diff(Ao(l,m),a(i,j),1);
      Diff_A_h(i,j,l,m) = Diff_A_h(i,j,l,m)+diff(A(l,m),a(i,j),1);
```

```
        end
        end
    end
end

%% Make Function File

matlabFunction(Ao,'file', 'f_Ao','vars', {a b x} );
matlabFunction(Diff_A_a, Diff_A_b,'file', 'diffAmg','vars', {a b x} );
matlabFunction(Diff_A_h,'file', 'diffAh','vars', {a} );
%% Emission Matrix for Binary (B)
R = sym('R', [numberofsites 1]); % Rainfall [0 or 1]
alpha = sym('alpha', [Statesize numberofsites]); % Independent Parameter



realbetasize = sum(0:numberofsites-1);
realbeta = sym('beta', [Statesize realbetasize]);% Dependent Parameter

% Auto-Logistic
beta(numberofsites,numberofsites,Statesize) = sym(0);
for s = 1:Statesize
    loc = sym(triu(ones(numberofsites),1));
    loc(loc==1)=realbeta(s,:);
    beta(:,:,s) = loc;
    clear loc
end

B2(Statesize,numberofsites) = sym(0);
for s = 1:Statesize
    for i = 1:numberofsites
        B2(s,i) = exp(alpha(s,i)*R(i) + beta(i,:,s)*R(i)*R)...
            /(1+exp(alpha(s,i) + beta(i,:,s)*R));
    end
end

n = 2^numberofsites;

binarytable_ch = dec2bin((0:n-1)',numberofsites);
binarytable_in = double(binarytable_ch-'0');

Buffertable(n,Statesize) = sym(0);
MPLE_B22(Statesize,1) = sym(0);
for s = 1:Statesize
    MPLE_B22(s,1) = exp(alpha(s,:)*R + transpose(R)*(beta(:,:,s)*R));
    for i = 1:n
        Buffertable(i,s) = exp(alpha(s,:)*binarytable_in(i,:)' +
binarytable_in(i,:)*(beta(:,:,s)*binarytable_in(i,:)'));
    end
    MPLE_B22(s,1) = MPLE_B22(s,1)/sum(Buffertable(:,s),1);
end

% Independent
B1(Statesize,numberofsites) = sym(0);
for s = 1:Statesize
```

```
    for i = 1:numberofsites
        B1(s,i) = (1-alpha(s,i))^(1-R(i))*(alpha(s,i))^R(i);
    end
end



MPLE_B1 = prod(B1,2);
MPLE_B2 = MPLE_B22;

diff_MPLE_B1_alpha(Statesize,numberofsites) = sym(0);
diff_MPLE_B1_beta(Statesize,realbetasize) = sym(0);
for s = 1:Statesize
    for i = 1:numberofsites
        diff_MPLE_B1_alpha(s,i) = diff_MPLE_B1_alpha(s,i)+diff(MPLE_B1(s,1),alpha(s,i),1);
    end
    for j = realbetasize
        diff_MPLE_B1_beta(s,j) = diff_MPLE_B1_beta(s,j)+diff(MPLE_B1(s,1),realbeta(s,j),1);
    end
end

diff_MPLE_B2_alpha(Statesize,numberofsites) = sym(0);
diff_MPLE_B2_beta(Statesize,realbetasize) = sym(0);

for s = 1:Statesize
    for i = 1:numberofsites
        diff_MPLE_B2_alpha(s,i) = diff_MPLE_B2_alpha(s,i)+diff(MPLE_B2(s,1),alpha(s,i),1);
    end
    for j = 1:realbetasize
        diff_MPLE_B2_beta(s,j) = diff_MPLE_B2_beta(s,j)+diff(MPLE_B2(s,1),realbeta(s,j),1);
    end
end

% for simulation
if spatialmodeltype == 1 || spatialmodeltype == 3
    B = B1;
    MPLE_B = MPLE_B1;
elseif spatialmodeltype == 2
    B = B2;
    MPLE_B = MPLE_B22;
end



%% Make Function File for B ...
matlabFunction(B,'file', 'f_B','vars', {alpha realbeta R} );
matlabFunction(MPLE_B,'file', 'f_MPLE_B','vars', {alpha realbeta R} );
matlabFunction(B1,'file', 'f_B1','vars', {alpha realbeta R} );
matlabFunction(MPLE_B1,'file', 'f_MPLE_B1','vars', {alpha realbeta R} );
matlabFunction(B2,'file', 'f_B2','vars', {alpha realbeta R} );
matlabFunction(MPLE_B2,'file', 'f_MPLE_B2','vars', {alpha realbeta R} );



matlabFunction(diff_MPLE_B1_alpha,diff_MPLE_B1_beta,'file', 'f_diff_MPLE_B1_ab','vars', {alpha
realbeta R} );
matlabFunction(diff_MPLE_B2_alpha,diff_MPLE_B2_beta,'file', 'f_diff_MPLE_B2_ab','vars', {alpha
realbeta R} );

end
```

```
function [x0 lb ub Con_size] = NHMM_Boundary(a,b,Alpha,Beta,Prim_Cond,Scon_Cond)
Statesize = Prim_Cond(1);
Wdim = Prim_Cond(2);
spatialmodeltype = Prim_Cond(3);
numberofsites = Prim_Cond(4);
initial = Scon_Cond(1);
independency = Scon_Cond(2);

%low/up boundary of transition matrix
if initial == true
    alb = ones(Statesize,Statesize)*exp(-500);
    aub = ones(Statesize,Statesize);
else
    alb = ones(Statesize,Statesize)*(-50);
    aub = ones(Statesize,Statesize)*(50);
end

if initial == true
    blb=zeros(Wdim,Statesize,Statesize);
    bub=zeros(Wdim,Statesize,Statesize);
else
    blb = ones(Wdim,Statesize,Statesize)*(-50);
    bub = ones(Wdim,Statesize,Statesize)*(50);
end



%low/up boundary of emission parameters
if spatialmodeltype == 1
    Betaub = [];
    Betalb = [];
elseif spatialmodeltype == 2
    Betalb = -ones(Statesize,sum(0:numberofsites-1))*50;
    Betaub = ones(Statesize,sum(0:numberofsites-1))*50;
    if initial == true || independency == true
        Betaub = zeros(Statesize,sum(0:numberofsites-1));
        Betalb = zeros(Statesize,sum(0:numberofsites-1));
    end
elseif spatialmodeltype == 3
    Betaub = [];
    Betalb = [];
else
    error('Invalid Spatial Model Type');
end



if spatialmodeltype == 1 || initial == true || spatialmodeltype == 3
    Alphalb = ones(Statesize,numberofsites)*exp(-500);
    Alphaub = ones(Statesize,numberofsites);
```

```
elseif spatialmodeltype == 2
    Alphalb = -ones(Statesize,numberofsites)*50;
    Alphaub = ones(Statesize,numberofsites)*50;
else
    error('Invalid Spatial Model Type');
end

% Change Matrix dimension to vector
[lb Con_size] =...
    ParameterEncode(alb,blb,Alphalb,Betalb,Prim_Cond,Scon_Cond);
[ub Con_size] =...
    ParameterEncode(aub,bub,Alphaub,Betaub,Prim_Cond,Scon_Cond);



[x0 Con_size] =...
    ParameterEncode(a,b,Alpha,Beta,Prim_Cond,Scon_Cond);

end
```

```
function [Parameter Con_size] = ParameterEncode(a,b,Alpha,Beta,Prim_Cond,Scon_Cond)
%All matrix of parameters to vector

spatialmodeltype = Prim_Cond(3);
initial = Scon_Cond(1);
independency = Scon_Cond(2);
em = Scon_Cond(4);
colsize = 1;

Converteda = reshape(a,colsize,[]);
Con_a_size = size(Converteda,2);

if initial == true
    Convertedb = [];
    Con_b_size = 0;
else
    Convertedb = reshape(b,colsize,[]);
    Con_b_size = size(Convertedb,2);
end

ConvertedAlpha = reshape(Alpha,colsize,[]);
Con_A_size = size(ConvertedAlpha,2);

if spatialmodeltype==1 || initial==true || independency == true
    ConvertedBeta = [];
    Con_B_size = 0;
elseif spatialmodeltype==2 || spatialmodeltype==3
    ConvertedBeta = reshape(Beta,colsize,[]);
    Con_B_size = size(ConvertedBeta,2);
end
```

```
if em == 0;
    if initial == false
        Parameter = [Converteda Convertedb ConvertedAlpha ConvertedBeta];
        Con_size = [Con_a_size Con_b_size Con_A_size Con_B_size];
    else
        Parameter = [Converteda ConvertedAlpha];
        Con_size = [Con_a_size 0 Con_A_size 0];
    end
elseif em == 1
    Parameter = [Converteda Convertedb];
    Con_size = [Con_a_size Con_b_size 0 0];
elseif em == 2
    if spatialmodeltype==1
        Parameter = [ConvertedAlpha];
        Con_size = [0 0 Con_A_size 0];
    else
        Parameter = [ConvertedAlpha ConvertedBeta];
        Con_size = [0 0 Con_A_size Con_B_size];
    end
end
```

```
function [cons conss] = NHMMconstraints(Prim_Cond,Scon_Cond)
Statesize = Prim_Cond(1);
Wdim = Prim_Cond(2);
spatialmodeltype = Prim_Cond(3);
numberofsites = Prim_Cond(4);
initial = Scon_Cond(1);
independency = Scon_Cond(2);
muzero = Scon_Cond(3);
em = Scon_Cond(4);

%% constraints
% cons = zeros(Statesize+(initial-1)*(Wdim+Statesize),size(lb,2));
% conss = zeros(Statesize+(initial-1)*(Wdim+Statesize),1);
if initial == true
    numberofcons = 0;
    trans=zeros(Statesize,Statesize);
    alp=zeros(Statesize,numberofsites);

    for s = 1:Statesize
        numberofcons = numberofcons + 1;

        trans(s,:) = 1;
        [constraints Con_size] =...
            ParameterEncode(trans,[],alp,[],...
            Prim_Cond,Scon_Cond);
        cons(numberofcons,:) = constraints(1,:);
        conss(numberofcons,1) = 0;
        trans(s,:) = 0;
    end
elseif initial == false
    if independency == true || spatialmodeltype == 1
```

```
        numberofcons = 0;
        trans=zeros(Statesize,Statesize);
        b = zeros(Wdim,Statesize,Statesize);
        alp=zeros(Statesize,numberofsites);
        for s = 1:Statesize
            numberofcons = numberofcons + 1;

            trans(s,:) = 1;
            [constraints Con_size] =...
                ParameterEncode(trans,b,alp,[],...
                Prim_Cond,Scon_Cond);
            cons(numberofcons,:) = constraints(1,:);
            conss(numberofcons,1) = 0;
            trans(s,:) = 0;
        end
        for w = 1:Wdim
            for s = 1:Statesize
                numberofcons = numberofcons + 1;

                b(w,s,:) = 1;
                [constraints Con_size] =...
                    ParameterEncode(trans,b,alp,[],...
                    Prim_Cond,Scon_Cond);
                cons(numberofcons,:) = constraints(1,:);
                conss(numberofcons,1) = 0;
                b(w,s,:) = 0;
            end
        end
    else
        numberofcons = 0;
        trans=zeros(Statesize,Statesize);
        b = zeros(Wdim,Statesize,Statesize);
        alp=zeros(Statesize,numberofsites);
        if spatialmodeltype == 1
            RealBeta = [];
        elseif spatialmodeltype == 2
            RealBeta = zeros(Statesize,sum(0:numberofsites-1));
        elseif spatialmodeltype == 3
            RealBeta = zeros(numberofsites-1,6,Statesize);
        end
        for s = 1:Statesize
            numberofcons = numberofcons + 1;

            trans(s,:) = 1;
            [constraints Con_size] =...
                ParameterEncode(trans,b,alp,RealBeta,...
                Prim_Cond,Scon_Cond);
            cons(numberofcons,:) = constraints(1,:);
            conss(numberofcons,1) = 0;
            trans(s,:) = 0;
        end
        for w = 1:Wdim
            for s = 1:Statesize
                numberofcons = numberofcons + 1;
                b(w,s,:) = 1;
                [constraints Con_size] =...
                    ParameterEncode(trans,b,alp,RealBeta,...
                    Prim_Cond,Scon_Cond);
                cons(numberofcons,:) = constraints(1,:);
                conss(numberofcons,1) = 0;
                b(w,s,:) = 0;
            end
        end
    end
end
```

```
function [ML DL] = MCMLinitial(Parameter,m,Rs,Prim_Cond,Scon_Cond)
% Maximize Likelihood
Statesize = Prim_Cond(1);
numberofsites = Prim_Cond(4);
initial = Scon_Cond(1);
[numberofmonths numberofyears] = size(Rs);
%%
A = reshape(Parameter(1:Statesize^2),Statesize,[]);
A = A./(A*ones(Statesize,Statesize));
B = reshape(Parameter(Statesize^2+1:end),Statesize,[]);
%%

n = 2^numberofsites;
binarytable_ch = dec2bin((0:n-1)',numberofsites);
binarytable_in = double(binarytable_ch-'0');

% Joint distribution of spatial model
B_distribution =...
    BinaryTable(1,B,[],1,initial);
Diff_a_distribution = zeros(Statesize,numberofsites,n);
for i = 1:n
    Diff_a_distribution(:,:,i) = f_diff_MPLE_B1_ab(B,[],binarytable_in(i,:)');
end
%%

ForDiffalpha = zeros(size(Diff_a_distribution(:,:,1))); % transition
Fordiffa = zeros(Statesize,Statesize,Statesize,Statesize); % emission






logliks = 0;
for y = 1:numberofyears
    seq = Rs{m,y};
    % decode HMM (calculate likelihood and forward(fs) and backward(bs))
    [pStates,pSeq,fs,bs,sc,bb] = NHMMdecodeinitial(seq,A,B_distribution,Statesize, numberofsites);
    logliks = pSeq + logliks;
    % for gradient
    if nargout  > 1

        logf = log(fs);
        logb = log(bs);
        seq = [zeros(numberofsites,1) seq];
        tsize = size(seq,2);



        diffAlpha=zeros(Statesize,numberofsites,tsize);



        for t = 2:tsize
            diffAlpha(:,:,t)=NHMMmissingdataSQP(seq(:,t),Diff_a_distribution);   % Marginal out missing data
        end

        diffa = repmat(diffAh(A),[1 1 1 1 tsize]);
        logGE = log(bb);
        logA = repmat(log(A),[1 1 tsize]);
        scales = log(sc);



        for k = 1:Statesize
            for l = 1:Statesize
                buffer = exp(reshape(logf(k,1,1:tsize-1)...
                    +logGE(l,1,2:tsize)+logb(l,1,2:tsize)-scales(1,1,2:tsize),1,[]));
                for u = 1:Statesize
                    for v = 1:Statesize
                        Fordiffa(u,v,k,l) = Fordiffa(u,v,k,l) + sum(reshape(diffa(u,v,k,l,2:tsize),1,[]).*buffer);
                    end
                end

                 buffer2 = exp(reshape(logf(k,1,1:tsize-1)...
                    +logA(k,l,2:tsize)+logb(l,1,2:tsize)-scales(1,1,2:tsize),1,[]));
                for s = 1:numberofsites
                    ForDiffalpha(l,s) = ForDiffalpha(l,s) + sum(reshape(diffAlpha(l,s,2:tsize),1,[]).*buffer2);
                end
            end
        end
    end
end



ML = -real(logliks);
if nargout  > 1
    [Parameter] =...
    [reshape(sum(sum(Fordiffa,4),3),1,[]) reshape(sum(sum(ForDiffalpha,4),3),1,[])];
    DL = -real(Parameter)';
end
```

```
function p = BinaryTable(spatialmodeltype,alpha,beta,independency,initial)
```

```
%Estimate Joint Distribution
    [Statesize numberofsites] = size(alpha);

    n = 2^numberofsites;

    p = zeros(n,Statesize);

    binarytable_ch = dec2bin((0:n-1)',numberofsites);
    binarytable_in = double(binarytable_ch-'0');

    if spatialmodeltype == 1 || initial == true || and(spatialmodeltype == 3,independency==true)
        for i = 1:n
            p(i,:) = f_MPLE_B1(alpha,beta,binarytable_in(i,:)')';
        end
    elseif spatialmodeltype == 2
        for i = 1:n
            if independency==true
                beta = zeros(Statesize,sum(0:numberofsites-1));
            end
            p(i,:) = f_MPLE_B2(alpha,beta,binarytable_in(i,:)')';
        end
    elseif spatialmodeltype == 3
        BG = 2.^(2-1:-1:0);
        for i = 1:n
            seq = binarytable_in(i,:)';
            Ptable = f_B1(alpha,[],seq);
            for s = 1:Statesize
                p(i,s) = Ptable(s,beta(1,1,s));
                for k = 1:size(beta,1)
                    if Ptable(s,beta(k,1,s)) == 0
                        p(i,s) = p(i,s)*0;
                    else
                        p(i,s) = p(i,s)*beta(k,BG*[seq(beta(k,1,s),1) seq(beta(k,2,s),1)]'+3,s)/Ptable(s,beta(k,1,s));
                    end
                end
            end
        end
    end
end
```

```
function [pStates,pSeq,fs,bs,s,Bs] = NHMMdecodeinitial(seq,A,B_distribution,Statesize, numberofsites)
%% Estimate Forward and Backward and Likelihood

seq = [ones(numberofsites,1), seq];

L = length(seq);

Bs(Statesize,1,size(seq,2)) = 0;



fs = zeros(Statesize,1,L);
```

```
bs = ones(Statesize,1,L);
s = zeros(1,1,L);

%we introduce a scaling factor
fs(1,1,1) = 1;
s(1) = 1;




Bs(:,:,1) = NHMMmissingdata(seq(:,1),B_distribution);

for count = 2:L

    Bs(:,:,count) = NHMMmissingdata(seq(:,count),B_distribution);
    e = Bs(:,:,count);
    for state = 1:Statesize
        fs(state,1,count) = e(state) .* (fs(:,1,count-1)' *A(:,state));
    end
    % scale factor normalizes sum(fs,count) to be 1.
    s(count) =  sum(fs(:,1,count),1);
    fs(:,1,count) =  fs(:,1,count)./s(count);
end

% so once again use the scale factor

for count = L-1:-1:1

    e = Bs(:,:,count+1);
    for state = 1:Statesize
      bs(state,1,count) = (1/s(count+1)) * sum( A(state,:)'.* bs(:,1,count+1) .* e);
    end
end

pSeq = sum(log(s));
pStates = [];
```

```
function [a,b,Alpha,Beta] = ParameterDecode(Parameter,Con_size,Prim_Cond,Scon_Cond)
%All vector of parameters to matrix

Statesize = Prim_Cond(1);
Wdim = Prim_Cond(2);
spatialmodeltype = Prim_Cond(3);
numberofsites = Prim_Cond(4);
initial = Scon_Cond(1);
independency = Scon_Cond(2);
em = Scon_Cond(4);
% colsize = 1;

Cum_size = cumsum(Con_size,2);
if em == 0
    a = reshape(Parameter(1,1:Cum_size(1)),Statesize,[]);
    if initial == false
        b = reshape(Parameter(1,Cum_size(1)+1:Cum_size(2)),Wdim,Statesize,Statesize);
    else
        b = zeros(Wdim,Statesize,Statesize);
    end



    Alpha = reshape(Parameter(1,Cum_size(2)+1:Cum_size(3)),Statesize,numberofsites);
    if spatialmodeltype ~= 3
        if independency == true || initial == true || spatialmodeltype == 1
            Beta = zeros(Statesize,sum(0:numberofsites-1));
        else
            Beta = reshape(Parameter(1,Cum_size(3)+1:Cum_size(4))...
                    ,Statesize,[]);
        end
    else
        if independency == true || initial == true
            Beta = [];
        else
            Beta = reshape(Parameter(1,Cum_size(3)+1:Cum_size(4)),numberofsites-1,6,Statesize);
        end
    end



elseif em == 1
    a = reshape(Parameter(1,1:Cum_size(1)),Statesize,[]);
    b = reshape(Parameter(1,Cum_size(1)+1:Cum_size(2)),Wdim,Statesize,Statesize);
    Alpha = [];
    Beta = [];
elseif em == 2
    Alpha = reshape(Parameter(1,Cum_size(2)+1:Cum_size(3)),Statesize,numberofsites);
    if spatialmodeltype ~= 3
        if independency == true || initial == true
            Beta = zeros(Statesize,sum(0:numberofsites-1));
        else
            Beta = reshape(Parameter(1,Cum_size(3)+1:Cum_size(4)),Statesize,[]);
        end
    else
        if independency == true || initial == true
            Beta = zeros(numberofsites-1,6,Statesize);
        else
            Beta = reshape(Parameter(1,Cum_size(3)+1:Cum_size(4)),numberofsites-1,6,Statesize);
        end
    end
    a = [];
    b = [];
end
```

```
function [ML DL] = MCML(Parameter,m,Rs,Wx,Con_size,Prim_Cond,Scon_Cond)
% Maximize Likelihood
```

```
Statesize = Prim_Cond(1);
Wdim = Prim_Cond(2);
spatialmodeltype = Prim_Cond(3);
numberofsites = Prim_Cond(4);
initial = Scon_Cond(1);
independency = Scon_Cond(2);


[numberofmonths numberofyears] = size(Rs);
%%
[a,b,Alpha,Beta] = ParameterDecode(Parameter,Con_size,Prim_Cond,Scon_Cond);

%%
B_distribution =...
   BinaryTable(spatialmodeltype,Alpha,Beta,independency,initial);
[Diff_a_distribution Diff_b_distribution] =...
   BinaryTableDiff(spatialmodeltype,Alpha,Beta,independency,initial);

[bsize1 bsize2] = size(Beta);


ForDiffalpha = zeros(size(Diff_a_distribution(:,:,1)));
ForDiffbeta = zeros(size(Diff_b_distribution(:,:,1)));

Fordiffa = zeros(Statesize,Statesize,Statesize,Statesize);
Fordiffb = zeros(Wdim,Statesize,Statesize,Statesize,Statesize);



logliks = 0;
for y = 1:numberofyears
   X = Wx{m,y};
   seq = Rs{m,y};
   [pStates,pSeq, fs, bs, sc, ao, bb] = NHMMdecode(X,seq,a,b,B_distribution,Statesize, numberofsites,
Wdim);

   logliks = pSeq + logliks;

   if nargout  > 1

      logf = log(fs);
      logb = log(bs);
      seq = [zeros(numberofsites,1) seq];
      X = [zeros(Wdim,1) X];
      tsize = size(X,2);

      diffa=zeros(Statesize,Statesize,Statesize,Statesize,tsize);
      diffb=zeros(Wdim,Statesize,Statesize,Statesize,Statesize,tsize);

      diffAlpha=zeros(Statesize,numberofsites,tsize);
      diffBeta=zeros(Statesize,bsize2,tsize);
```

```
        for t = 2:tsize
            [diffa(:,:,:,:,t) diffb(:,:,:,:,:,t)] = ...
                diffAmg(a,b,X(:,t));

            diffAlpha(:,:,t)=NHMMmissingdataSQP(seq(:,t),Diff_a_distribution);
            if spatialmodeltype == 2
                diffBeta(:,:,t)=NHMMmissingdataSQP(seq(:,t),Diff_b_distribution);
            end
        end

        logGE = log(bb);
        logA = log(ao);
        scales = log(sc);



        for k = 1:Statesize
            for l = 1:Statesize
                buffer = exp(reshape(logf(k,1,1:tsize-1)...
                    +logGE(l,1,2:tsize)+logb(l,1,2:tsize)-scales(1,1,2:tsize),1,[]));
                for u = 1:Statesize
                    for v = 1:Statesize
                        Fordiffa(u,v,k,l) = Fordiffa(u,v,k,l) + sum(reshape(diffa(u,v,k,l,2:tsize),1,[]).*buffer);
                        for i = 1:Wdim
                            Fordiffb(i,u,v,k,l) = Fordiffb(i,u,v,k,l) + sum(reshape(diffb(i,u,v,k,l,2:tsize),1,[]).*buffer);
                        end
                    end
                end

                 buffer2 = exp(reshape(logf(k,1,1:tsize-1)...
                    +logA(k,l,2:tsize)+logb(l,1,2:tsize)-scales(1,1,2:tsize),1,[]));

                for s = 1:numberofsites
                    ForDiffalpha(l,s) = ForDiffalpha(l,s) + sum(reshape(diffAlpha(l,s,2:tsize),1,[]).*buffer2);
                end
                if spatialmodeltype==2
                    for g = 1:bsize2
                        ForDiffbeta(l,g) = ForDiffbeta(l,g) + sum(reshape(diffBeta(l,g,2:tsize),1,[]).*buffer2);
                    end
                end
            end
        end
    end
end



ML = -real(logliks);
if nargout  > 1
    [Parameter] =...

ParameterEncode(sum(sum(Fordiffa,4),3),sum(sum(Fordiffb,5),4),sum(sum(ForDiffalpha,4),3),sum(sum(For
Diffbeta,4),3),...
    Prim_Cond,Scon_Cond);
    DL = -real(Parameter)';
```

```
end
```

```
function [a b] = BinaryTableDiff(spatialmodeltype,alpha,beta,independency,initial)
   [Statesize numberofsites] = size(alpha);
   realbetasize = sum(0:numberofsites-1);
   n = 2^numberofsites;



   binarytable_ch = dec2bin((0:n-1)',numberofsites);
   binarytable_in = double(binarytable_ch-'0');

   if spatialmodeltype == 1 || initial == true || spatialmodeltype == 3
      a = zeros(Statesize,numberofsites,n);
      b = zeros(Statesize,realbetasize,n);
      for i = 1:n
         [a(:,:,i) b(:,:,i)] = f_diff_MPLE_B1_ab(alpha,beta,binarytable_in(i,:)');
      end
   elseif spatialmodeltype == 2
      a = zeros(Statesize,numberofsites,n);
      b = zeros(Statesize,realbetasize,n);
      for i = 1:n
         if independency==true
            beta = zeros(Statesize,sum(0:numberofsites-1));
         end
         [a(:,:,i) b(:,:,i)] = f_diff_MPLE_B2_ab(alpha,beta,binarytable_in(i,:)');
      end
   else
      error('BTD');
   end
end
```

```
function [guessa,guessb,guessAlpha,guessBeta,logliks,f_parameters] =...
   EMalgorithm(m,Rs,Wx,guessa,guessb,guessAlpha,guessBeta,Prim_Cond,Scon_Cond,tol,maxiter)
%%
Statesize = Prim_Cond(1);
Wdim = Prim_Cond(2);
spatialmodeltype = Prim_Cond(3);
numberofsites = Prim_Cond(4);
initial = Scon_Cond(1);
independency = Scon_Cond(2);



[numberofmonths numberofyears] = size(Rs);

disp('guessa');
fprintf([repmat('%g ',1,size(guessa,2)),'\n'],guessa');
disp('guessAlpha');
fprintf([repmat('%g ',1,size(guessAlpha,2)),'\n'],guessAlpha');
```

```
trtol = tol;
etol = tol;
%% Initialize for modeltype3
if spatialmodeltype == 3
   clear currentState
   Scon_Cond(2)=true;
   for y = 1:numberofyears
      mcurrentstate = 1;
      [currentState{y}]...
         = MCMEviterbi(Wx{m,y},Rs{m,y},guessa,guessb,guessAlpha,guessBeta,...
         mcurrentstate,Prim_Cond,Scon_Cond);
   end
   disp('Independent Alpha before');
   fprintf([repmat('%g ',1,size(guessAlpha,2)),'\n'],guessAlpha');
   Scon_Cond(2)=false;
   [Tree guessAlpha MSTreeEdges guessBeta MI_table]=...
      TreeStructure(currentState,Rs,Prim_Cond,m);

   disp('Independent Alpha after');
   fprintf([repmat('%g ',1,size(guessAlpha,2)),'\n'],guessAlpha');
   disp('-------------------------------------------');
end

converged = false;
loglik = -inf; % loglik is the log likelihood of all sequences given the TR and E

for iteration = 1:maxiter
   disp('-------------------------------------------');
   oldLL = loglik;



   oldGuessa = guessa;
   oldGuessb = guessb;
   oldGuessAlpha = guessAlpha;
   oldGuessBeta = guessBeta;

%%
   Scon_Cond(4) = 0; %all parameter
   [Parameter0 lb0 ub0 Con_size0] =...
         NHMM_Boundary(guessa,guessb,guessAlpha,guessBeta,...
         Prim_Cond,Scon_Cond);
   [cons0 conss0] = NHMMconstraints(Prim_Cond,Scon_Cond);
   Scon_Cond(4) = 1; %state parameter
   [Parameter1 lb1 ub1 Con_size1] =...
         NHMM_Boundary(guessa,guessb,guessAlpha,guessBeta,...
         Prim_Cond,Scon_Cond);
   [cons1 conss1] = NHMMconstraints(Prim_Cond,Scon_Cond);
   Scon_Cond(4) = 2; %emission parameter
   [Parameter2 lb2 ub2 Con_size2] =...
         NHMM_Boundary(guessa,guessb,guessAlpha,guessBeta,...
         Prim_Cond,Scon_Cond);
   [cons2 conss2] = NHMMconstraints(Prim_Cond,Scon_Cond);
```

```
  %% Viterbi
  clear currentState
  clear logfs logbs logscales logAA logBB
  loglikelihood = 0;
  for y = 1:numberofyears
    mcurrentstate = 1;
    [currentState{y}]...
      = MCMEviterbi(Wx{m,y},Rs{m,y},guessa,guessb,guessAlpha,guessBeta,...
      mcurrentstate,Prim_Cond,Scon_Cond);

    B_distribution =...
      BinaryTable(spatialmodeltype,guessAlpha,guessBeta,independency,initial);
    [pStates,logPseq, fs, bs, scales, AA, BB] =...
      NHMMdecode(Wx{m,y},Rs{m,y},guessa,guessb,...
      B_distribution,Statesize, numberofsites, Wdim);
    loglikelihood = loglikelihood+logPseq;
    ch = [sum(sum(fs<0|fs==nan|fs==inf)),...
        sum(sum(bs<0|bs==nan|bs==inf)),...
        sum(sum(scales<0|scales==nan|scales==inf)),...
        sum(sum(sum(AA<0|AA==nan|AA==inf))),...
        sum(sum(sum(BB<0|BB==nan|BB==inf)))];
      if sum(ch) ~= 0
        disp('ch');
      end
    logfs{y} = log(fs);
    logbs{y} = log(bs);
    logscales{y} = log(scales);
    logAA{y} = log(AA);
    logBB{y} = log(BB);

  end
  if iteration==1
    disp(fprintf('Startwith loglikelihood = %g',loglikelihood));
  end



%%

    %% States (1)

    A = [];
    B = [];
    Aeq = [];
    Beq = [];
    MaxFunEvals_Data = 10000;
    MaxIter_Data = 1000;

    Scon_Cond(4) = 1;
    [Parameter1 fval] =...
      SQP2(@(x) MCEMdouble2(x,Parameter0,m,Wx,Rs,...
    logfs,logbs,logscales,logAA,logBB,...
    Con_size1,Con_size0,Prim_Cond,Scon_Cond),...
    Parameter1,A,B,Aeq,Beq,lb1,ub1,MaxFunEvals_Data,MaxIter_Data);
    disp('**Done For States');
    %% Emission (2)
    Scon_Cond(4) = 2;
    if spatialmodeltype == 3
        [Tree guessAlpha MSTreeEdges guessBeta MI_table]=...
            TreeStructure(currentState,Rs,Prim_Cond,m);
        [Parameter2] =...
            ParameterEncode([],[],guessAlpha,guessBeta,...
            Prim_Cond,Scon_Cond);

        disp('CL guessAlpha');
        fprintf([repmat('%g ',1,size(guessAlpha,2)),'\n'],guessAlpha');
        disp('CL guessBeta');
        fprintf([repmat('%g ',1,size(reshape(guessBeta,numberofsites-
1,[]),2)),'\n'],reshape(guessBeta,numberofsites-1,[])');
    else
        A = [];
        B = [];
        Aeq = [];
        Beq = [];
        MaxFunEvals_Data = 10000;
        MaxIter_Data = 1000;

        [Parameter2 fval] =...
            SQP2(@(x) MCEMoccurence2(x,Parameter0,m,Wx,Rs,...
            logfs,logbs,logscales,logAA,logBB,...
            Con_size2,Con_size0,Prim_Cond,Scon_Cond),...
            Parameter2,A,B,Aeq,Beq,lb2,ub2,MaxFunEvals_Data,MaxIter_Data);
    end
    disp('**Done For Emis');
    %% Convert Emis
    Scon_Cond(4) = 2;
    [g1,g2, guessAlpha,guessBeta] =...
        ParameterDecode(Parameter2,Con_size2,Prim_Cond,Scon_Cond);
    %% Convert States
    Scon_Cond(4) = 1;
    [guessa,guessb,g3,g4] =...
        ParameterDecode(Parameter1,Con_size1,Prim_Cond,Scon_Cond);



    %%
    Scon_Cond(4) = 0;
    [f_parameters] =...
        NHMM_Boundary(guessa,guessb,guessAlpha,guessBeta,...
        Prim_Cond,Scon_Cond);
  loglik = 0;
  for y = 1:numberofyears
    B_distribution =...
        BinaryTable(spatialmodeltype,guessAlpha,guessBeta,independency,initial);
    [pStates,logPseq, fs, bs, scales, AA, BB] =...
        NHMMdecode(Wx{m,y},Rs{m,y},guessa,guessb,...
        B_distribution,Statesize, numberofsites, Wdim);
    loglik = loglik+logPseq;
  end
    %%
    disp('Final guessa');
    fprintf([repmat('%g ',1,size(guessa,2)),'\n'],guessa');
    if spatialmodeltype ~= 3
        disp('Final guessAlpha');
        fprintf([repmat('%g ',1,size(guessAlpha,2)),'\n'],guessAlpha');
    end
    disp(sprintf('[EM][m=%g] fval=%g loglik=%g oldLL=%g...', m,fval,loglik,oldLL));
    disp('-------------------------------------------');
    logliks(iteration,1) = loglik;
    if (abs(loglik-oldLL)/(1+abs(oldLL))) < tol
        if norm(guessa - oldGuessa,inf)/Statesize < trtol
            if norm(reshape(guessb - oldGuessb,Statesize,[]),inf)/(Statesize+Wdim) < trtol
                if norm(reshape(guessBeta - oldGuessBeta,Statesize,[]),inf)/(Statesize+numberofsites) < trtol
                    if norm(guessAlpha - oldGuessAlpha,inf)/(Statesize+numberofsites) < etol
                      converged = true;
                      break
                    end
                end
            end
        end
    end


end


if ~converged
    warning('NoConvergence',...
      'not converged with tolerance %f in %d iterations.',tol,maxiter);
end

end
```

```
function [currentState, logP, finalState]...
    = MCMEviterbi(X,seq,a,b,alpha,beta,mcurrentstate,Prim_Cond,Scon_Cond)

spatialmodeltype = Prim_Cond(3);

initial = Scon_Cond(1);
independency = Scon_Cond(2);


numStates = size(a,1);

B_distribution =...
    BinaryTable(spatialmodeltype,alpha,beta,independency,initial);

L = size(seq,2);
```

```
currentState = zeros(1,L);

if L == 0
   return
end


pTR = zeros(numStates,L);

%assumption is that model is in state mcurrentstate
v = -Inf(numStates,1);
v(mcurrentstate,1)=0;
vOld = v;



for count = 1:L
   logA = log(f_Ao(a,b,X(:,count)));
   for state = 1:numStates % (j) states(next)
      % for each state we calculate
      bestVal = -inf;
      bestPTR = 0;
      % use tr loop to avoid lots of calls to max
      val = zeros(1,numStates);
      for inner = 1:numStates % (i) i states(now) -> j [best way]

         val(1,inner) = vOld(inner) + logA(inner,state);
         if val(1,inner) >= bestVal
            bestVal = val(1,inner);
            bestPTR = inner;
         end
      end

      %% (Random Part I)PTR
      test=find(val(1,:)==bestVal);
      testsize=size(test);
      if 1~=testsize
         rnd = rand(1);
         selectstate = 0;
         for i = 1:testsize
            selectstate = selectstate+1/testsize;
            if selectstate <= rnd
               bestPTR = test(i);
            end
         end
      end

      %%
      % save the best transition information for later backtracking
      pTR(state,count) = bestPTR;
```

```
        % update v
        e = NHMMmissingdata(seq(:,count),B_distribution);
        v(state) = log(e(state,1)) + bestVal;

    end

    vOld = v;
end

% decide which of the final states is post probable
[logP, finalState] = max(v); % [ final value and final state ]
%%
% Now back trace through the model and getting states
currentState(L) = finalState;

for count = L-1:-1:1
    currentState(count) = pTR(currentState(count+1),count+1);
    if currentState(count) == 0
        error('Zero');
    end
end
```

```
function [Tree Ind_P MSTreeEdges EdgePtables MI_table]=TreeStructure(currentState,Rs,...
    Prim_Cond,m)

Statesize = Prim_Cond(1);
numberofsites = Prim_Cond(4);

numberofyears = size(Rs,2);
count = zeros(1,Statesize);
DataTable{Statesize}=[];
for y = 1:numberofyears
    Allstates= currentState{y};
    seq = Rs{m,y};
    for t = 1:size(seq,2)
        s = Allstates(1,t);
        if isempty(find(seq(:,t)<0))
            count(1,s) = count(1,s) + 1;
            DataTable{s}(count(1,s),:) = seq(:,t)';
        end
    end
end

Tree=zeros(numberofsites,numberofsites,Statesize);
Ind_P=zeros(Statesize,numberofsites);
MSTreeEdges=zeros(numberofsites-1,3,Statesize);
EdgePtables=[ones(numberofsites-1,2,Statesize) zeros(numberofsites-1,4,Statesize)];

MI_table=zeros(numberofsites*(numberofsites-1)/2,3,Statesize);

parfor state = 1:Statesize
```

```
    if ~isempty(DataTable{state})
        [Tree(:,:,state) Ind_P(state,:) MSTreeEdges(:,:,state) EdgePtables(:,:,state) MI_table(:,:,state)] =...
            ChowLiuTree(DataTable{state});
    end
end
```

```
function [Tree Ind_P MSTreeEdges EdgePtables MI_table] = ChowLiuTree(binarydata)

Order = 2;
row = size(binarydata,2);
col = size(binarydata,1);



%% Make Joint Distribution 3D
Joint_P = zeros(row,row,Order^2);
binarytable_ch = dec2bin((0:Order^2-1)',Order);
binarytable = double(binarytable_ch-'0');

for i = 1:row
    for j = i:row
        count = 0;
        for k = 1:Order^2
            Joint_P(i,j,k) = ...
                sum((binarydata(:,i) == binarytable(k,1)&...
                binarydata(:,j) == binarytable(k,2)),1);
            count = count+Joint_P(i,j,k);
        end
        Joint_P(i,j,:) = Joint_P(i,j,:)/count;
    end
    Ind_P(1,i)=Joint_P(i,i,4);
end

%% Mutual Information
MI = zeros(row,row);
MI_table = zeros(row*(row-1)/2,3);

count = 0;
for i = 1:row
    for j = i+1:row
        for k = 1:Order^2
            Buffer = dec2bin(k-1,2);
            if Buffer(1) == '1'
                p1 = Ind_P(i);
            else
                p1 = 1-Ind_P(i);
            end
            if Buffer(2) == '1'
                p2 = Ind_P(j);
            else
                p2 = 1-Ind_P(j);
            end
            if Joint_P(i,j,k)/(p1*p2) > 1 && Joint_P(i,j,k) > 0
            MI(i,j) = MI(i,j)+...
                Joint_P(i,j,k)*log(Joint_P(i,j,k)/(p1*p2));
            end
        end

        count = count + 1;
        MI_table(count,1)=MI(i,j);
        MI_table(count,2)=i;
        MI_table(count,3)=j;

    end
end
%% MWST (Maximum Weight Spanning Tree)

[Tree MSTreeEdges] =  MWST (MI_table,row);
MSTreeEdges = sortrows(MSTreeEdges,2);
for k = 1:size(MSTreeEdges,1)
    for i = 1:2^Order
        EdgePtables(k,i) = Joint_P(MSTreeEdges(k,2),MSTreeEdges(k,3),i);
    end
end

EdgePtables = [MSTreeEdges(:,2:3) EdgePtables];
EdgePtables = Tracking(EdgePtables);
% view(biograph( Tree ))
end
```

```
function [Tree MSTreeEdges] =  MWST (MI_table,row)
% The function takes CostMatrix as input and returns the maximum spanning tree T
% Uses Kruskal's Algorithm
% Extract the edge weights from the cost matrix
% Sort the edges in a non decreasing order of weights
% This algorithm is revised by Lowell Guangdi at 2009/06/11.
n = row; %Number of vertices

SortedEdgeWeights = sortrows(MI_table,1);
% First column of SortedEdgeWeights are the weights
% Second and third column are the vertices that the edges connect
m = size(SortedEdgeWeights,1); % number of edges

% We use the Disjoint sets data structures to detect cycle while adding new
% edges. Union by Rank with path compression is implemented here.

% Assign parent pointers to each vertex. Initially each vertex points to
% itself. Now we have a conceptual forest of n trees representing n disjoint
% sets
global ParentPointer ;
ParentPointer(1:n) = 1:n;

% Assign a rank to each vertex (root of each tree). Initially all vertices
% have the rank zero.
```

```
TreeRank(1:n) = 0;

% Visit each edge in the sorted edges array
% If the two end vertices of the edge are in different sets (no cycle), add
% the edge to the set of edges in maximum spanning tree
MSTreeEdges = 0;
MSTreeEdgesCounter = 0; i = m;
while ((MSTreeEdgesCounter < (n-1)) && (i>=1))
%Find the roots of the trees that the selected edge's two vertices
%belong to. Also perform path compression.

   temproot = SortedEdgeWeights(i,2);
   root1 = FIND_PathCompression(temproot);

   temproot = SortedEdgeWeights(i,3);
   root2 = FIND_PathCompression(temproot);

   if (root1 ~= root2)
      MSTreeEdgesCounter = MSTreeEdgesCounter + 1;
      MSTreeEdges(MSTreeEdgesCounter,1:3) = SortedEdgeWeights(i,:);
      if (TreeRank(root1)>TreeRank(root2))
         ParentPointer(root2)=root1;
      else
         if (TreeRank(root1)==TreeRank(root2))
           TreeRank(root2)=TreeRank(root2) + 1;
         end
         ParentPointer(root1)=root2;
      end
   end
   i = i - 1;
end

MSTreeEdgesCounter = 0;
Tree = 0;
Tree(1:n,1:n)=0;
while (MSTreeEdgesCounter < (n-1))
   MSTreeEdgesCounter = MSTreeEdgesCounter + 1;
   Tree(MSTreeEdges(MSTreeEdgesCounter,2),MSTreeEdges(MSTreeEdgesCounter,3))=1;
   Tree(MSTreeEdges(MSTreeEdgesCounter,3),MSTreeEdges(MSTreeEdgesCounter,2))=1;
end

end

function [parent] = FIND_PathCompression(temproot)

global ParentPointer;
ParentPointer(temproot);
if (ParentPointer(temproot)~=temproot)
   ParentPointer(temproot) = FIND_PathCompression(ParentPointer(temproot));
end
parent = ParentPointer(temproot);
end
```

```
function [ML DL] = MCEMdouble2(x0,Fixed_Parameter,m,Wx,Rs,...
    logfs,logbs,logscales,logAA,logBB,Con_size1,Con_size0,Prim_Cond,Scon_Cond)
% EM for Transition Matrix
Statesize = Prim_Cond(1);
Wdim = Prim_Cond(2);
spatialmodeltype = Prim_Cond(3);
numberofsites = Prim_Cond(4);
initial = Scon_Cond(1);
independency = Scon_Cond(2);

[numberofmonths numberofyears] = size(Rs);



%%

Scon_Cond(4) = 1;
[a2,b2,Alpha2,Beta2] = ParameterDecode(x0,Con_size1,Prim_Cond,Scon_Cond);

%%



ForA = zeros(Statesize,Statesize);



Fordiffa = zeros(Statesize,Statesize,Statesize,Statesize);
Fordiffb = zeros(Wdim,Statesize,Statesize,Statesize,Statesize);



for y = 1:numberofyears

    seq = [zeros(numberofsites,1) Rs{m,y}];
    X = [zeros(size(Wx{m,y},1),1) Wx{m,y}];
    logf = logfs{y};
    logb = logbs{y};
    scales = logscales{y};
    logA = logAA{y};
    logGE = logBB{y};
    tsize = size(X,2);



    diffa=zeros(Statesize,Statesize,Statesize,Statesize,tsize);
    diffb=zeros(Wdim,Statesize,Statesize,Statesize,Statesize,tsize);

    AA2 = zeros(Statesize,Statesize,tsize);
    for t = 2:size(seq,2)
        AA2(:,:,t) = f_Ao(a2,b2,X(:,t));
        if nargout  > 1
        [diffa(:,:,:,:,t) diffb(:,:,:,:,:,t)]=...
            diffAmg(a2,b2,X(:,t));
        end
    end
    logA2 = log(AA2);

    for k = 1:Statesize
        for l = 1:Statesize
            buffer = reshape(exp(logf(k,1,1:tsize-1)...
                +logA(k,l,2:tsize)+logGE(l,1,2:tsize)+logb(l,1,2:tsize)-scales(1,1,2:tsize)),1,[]);
            ForA(k,l) = ForA(k,l) + sum(reshape(logA2(k,l,2:tsize),1,[]).*buffer(1,:));
            if nargout  > 1
                for u = 1:Statesize
                    for v = 1:Statesize
                        Fordiffa(u,v,k,l) = Fordiffa(u,v,k,l) +
sum((reshape(diffa(u,v,k,l,2:tsize),1,[]))./reshape(AA2(k,l,2:tsize),1,[]).*buffer(1,:));
                        for i = 1:Wdim
                            Fordiffb(i,u,v,k,l) = Fordiffb(i,u,v,k,l) +
sum((reshape(diffb(i,u,v,k,l,2:tsize),1,[]))./reshape(AA2(k,l,2:tsize),1,[]).*buffer(1,:));
                        end
                    end
                end
            end
        end
    end


end

ML = -sum(sum(ForA));

if nargout  > 1
    Scon_Cond(4) = 1;
    [Parameter] =...
    ParameterEncode(sum(sum(Fordiffa,4),3),sum(sum(Fordiffb,5),4),[],[],...
    Prim_Cond,Scon_Cond);
    DL = -Parameter';

end
```

```
function [ML G] = MCEMoccurence2(x0,Fixed_Parameter,m,Wx,Rs,...
    logfs,logbs,logscales,logAA,logBB,Con_size2,Con_size0,Prim_Cond,Scon_Cond)
Statesize = Prim_Cond(1);
Wdim = Prim_Cond(2);
spatialmodeltype = Prim_Cond(3);
numberofsites = Prim_Cond(4);
initial = Scon_Cond(1);
independency = Scon_Cond(2);
 [numberofmonths numberofyears] = size(Rs);

%%

Scon_Cond(4) = 2;
```

```
[a2,b2,Alpha2,Beta2] = ParameterDecode(x0,Con_size2,Prim_Cond,Scon_Cond);
[bsize1 bsize2] = size(Beta2);
%%

B_distribution2 =...
    BinaryTable(spatialmodeltype,Alpha2,Beta2,independency,initial);
[Diff_a_distribution Diff_b_distribution] =...
    BinaryTableDiff(spatialmodeltype,Alpha2,Beta2,independency,initial);

ForB = zeros(Statesize,1);
ForDiffa = zeros(Statesize,numberofsites);
ForDiffb = zeros(size(Beta2));

for y = 1:numberofyears

    seq = [zeros(numberofsites,1) Rs{m,y}];

    logf = logfs{y};
    logb = logbs{y};


    for t = 2:size(seq,2)
        diffAlpha=NHMMmissingdataSQP(seq(:,t),Diff_a_distribution);
        diffBeta=NHMMmissingdataSQP(seq(:,t),Diff_b_distribution);
        B = NHMMmissingdata(seq(:,t),B_distribution2);
        logB = log(B);
        for l = 1:Statesize
            buffer2 = exp(reshape(logf(l,1,t)+logb(l,1,t),1,[]));
            ForB(l,1) = ForB(l,1)+sum(reshape(logB(l,1),1,[]).*buffer2);
            if nargout  > 1
                for s = 1:numberofsites
                    ForDiffa(l,s) = ForDiffa(l,s) + sum(reshape(diffAlpha(l,s)/B(l,1),1,[]).*buffer2);
                end
                if spatialmodeltype == 2
                    for g = 1:bsize2
                        ForDiffb(l,g) = ForDiffb(l,g) + sum(reshape(diffBeta(l,g)/B(l,1),1,[]).*buffer2);
                    end
                end
            end
        end
    end
end

ML = -sum(sum(ForB));
if nargout  > 1
    Scon_Cond(4) = 2;
    [Parameter] =...
    ParameterEncode([],[],ForDiffa,ForDiffb,...
    Prim_Cond,Scon_Cond);
    G = -Parameter';
end
```

```
function [PRline PEline PRliney PEliney MRline MEline MRliney MEliney Pre Mod...
      PRline2 PEline2 PRliney2 PEliney2 MRline2 MEline2 MRliney2 MEliney2 Pre2 Mod2...
      P1 P2 M1 M2 PRr2 MRr2 PSr2 MSr2 PRr MRr PSr MSr Ramount PRamount MRamount...
      PRcorr MRcorr PScorr MScorr PRbin MRbin PSbin MSbin] =...

NHMMsampling(ParameterdataW,ModeldataW,ParameterdataR,ModeldataR,NumTimeValueR,NumTimeV
alueW,rowsizeW,rowsizeR,...
   consideryear,considermonth,Statesize,numberofsites,spatialmodeltype,independency,initial,...
   guessa,guessb,guessAlpha,guessBeta,Thhold,fpa,pmodel)
numberofmonth = size(considermonth,1);
Pnumberofyear = consideryear(1,2)-consideryear(1,1)+1;
Mnumberofyear = consideryear(2,2)-consideryear(2,1)+1;

% disp(sprintf('--------------------------------------sampling'));
Pstartdate=datenum(consideryear(1,1),1,1);
Penddate=datenum(consideryear(1,2),12,31);
Palldate = Penddate-Pstartdate+1;
Pdatematrix = zeros(Palldate,3);
Pdatematrix(1:Palldate,1:3)=[year(Pstartdate:Penddate)' month(Pstartdate:Penddate)'
day(Pstartdate:Penddate)']; %put year month date
Presultseq = zeros(Palldate,3+numberofsites);
Presultseq(1:Palldate,1:3) = Pdatematrix;
Presultamount = Presultseq;
% Presultstate=Presultseq;
Ploc=zeros(size(Pdatematrix,1),1);
%% for Prediction
Mstartdate=datenum(consideryear(2,1),1,1);
Menddate=datenum(consideryear(2,2),12,31);
Malldate = Menddate-Mstartdate+1;
Mdatematrix = zeros(Malldate,3);
Mdatematrix(1:Malldate,1:3)=[year(Mstartdate:Menddate)' month(Mstartdate:Menddate)'
day(Mstartdate:Menddate)']; %put year month date
Mresultseq = zeros(Malldate,3+numberofsites);
Mresultseq(1:Malldate,1:3) = Mdatematrix;
Mresultamount = Mresultseq;
Mloc=zeros(size(Mdatematrix,1),1);
%%
fs = ones(1,Statesize);
for y = consideryear(1,1):consideryear(1,2)
   for m = 1:numberofmonth
      [seq,fs,s,largeW,largeG]=
NHMMgenerate_general(ParameterdataW(and(ParameterdataW(:,1)==y,ParameterdataW(:,2)>=considermo
nth(m,1)&ParameterdataW(:,2)<=considermonth(m,2)),NumTimeValueW+1:rowsizeW)',...

guessa{m},guessb{m},guessAlpha{m},guessBeta{m},fs,numberofsites,spatialmodeltype,independency,initia
l);

Presultseq(and(Presultseq(:,1)==y,Presultseq(:,2)>=considermonth(m,1)&Presultseq(:,2)<=considermonth(m
,2)),4:3+numberofsites)=seq'-1;

Ploc(and(Presultseq(:,1)==y,Presultseq(:,2)>=considermonth(m,1)&Presultseq(:,2)<=considermonth(m,2)),1)
= 1;
```

```
        fv = zeros(size(seq));
        for s = 1:numberofsites
            for i = 1:size(seq,2)
                if seq(s,i) == 2
                    rn = 0.2+0.78*rand();
                    rn = rand();
                    if pmodel == 1
                        fv(s,i) = expinv(rn,fpa{s,m}(1));
                    elseif pmodel == 2
                        fv(s,i) = gaminv(rn,fpa{s,m}(1),fpa{s,m}(2));
                    elseif pmodel == 3
                        fv(s,i) = MixedExponentialRand( fpa{s,m}(2:3)' , [fpa{s,m}(1) 1-fpa{s,m}(1)]' );
                    elseif pmodel == 4
                        fv(s,i) = wblinv(rn,fpa{s,m}(1),fpa{s,m}(2));
                    else
                        error('pmodel');
                    end
                elseif seq(s,i) == 1
                    fv(s,i) = 0;
                else
                    error('fv');
                end
            end
        end

Presultamount(and(Presultseq(:,1)==y,Presultseq(:,2)>=considermonth(m,1)&Presultseq(:,2)<=considermont
h(m,2)),4:3+numberofsites)=fv';
    end
end
%%
fs = ones(1,Statesize);
for y = consideryear(2,1):consideryear(2,2)
    for m = 1:numberofmonth
        [seq,fs,s,largeW,largeG]=
NHMMgenerate_general(ModeldataW(and(ModeldataW(:,1)==y,ModeldataW(:,2)>=considermonth(m,1)&
ModeldataW(:,2)<=considermonth(m,2)),NumTimeValueW+1:rowsizeW)',...

guessa{m},guessb{m},guessAlpha{m},guessBeta{m},fs,numberofsites,spatialmodeltype,independency,initia
l);

Mresultseq(and(Mresultseq(:,1)==y,Mresultseq(:,2)>=considermonth(m,1)&Mresultseq(:,2)<=considermont
h(m,2)),4:3+numberofsites)=seq'-1;

Mloc(and(Mresultseq(:,1)==y,Mresultseq(:,2)>=considermonth(m,1)&Mresultseq(:,2)<=considermonth(m,2)
),1) = 1;
        fv = zeros(size(seq));
        for s = 1:numberofsites
            for i = 1:size(seq,2)
                if seq(s,i) == 2
                    rn = 0.2+0.78*rand();
                    rn = rand();
                    if pmodel == 1
                        fv(s,i) = expinv(rn,fpa{s,m}(1));
                    elseif pmodel == 2
                        fv(s,i) = gaminv(rn,fpa{s,m}(1),fpa{s,m}(2));
                    elseif pmodel == 3
                        fv(s,i) = MixedExponentialRand( fpa{s,m}(2:3)' , [fpa{s,m}(1) 1-fpa{s,m}(1)]' );
                    elseif pmodel == 4
                        fv(s,i) = wblinv(rn,fpa{s,m}(1),fpa{s,m}(2));
                    else
                        error('pmodel');
                    end
                elseif seq(s,i) == 1
                    fv(s,i) = 0;
                else
                    error('fv');
                end
            end
        end

Mresultamount(and(Mresultseq(:,1)==y,Mresultseq(:,2)>=considermonth(m,1)&Mresultseq(:,2)<=considerm
onth(m,2)),4:3+numberofsites)=fv';
    end
end

PR = ParameterdataR(:,NumTimeValueR+numberofsites+1:end);
MR = ModeldataR(:,NumTimeValueR+numberofsites+1:end);
PS = Presultseq(:,NumTimeValueR+1:end);
MS = Mresultseq(:,NumTimeValueR+1:end);

PR2 = ParameterdataR(:,NumTimeValueR+1:NumTimeValueR+numberofsites);
MR2 = ModeldataR(:,NumTimeValueR+1:NumTimeValueR+numberofsites);
PS2 = Presultamount(:,NumTimeValueR+1:end);
MS2 = Mresultamount(:,NumTimeValueR+1:end);

for sites = NumTimeValueR+1:NumTimeValueR+numberofsites

    tarP = ~~Ploc&ParameterdataR(:,sites+numberofsites)>=0;
    tarM = ~~Mloc&ModeldataR(:,sites+numberofsites)>=0;

    Pre(:,:,sites-NumTimeValueR) = [sum(ParameterdataR(tarP,sites+numberofsites),1) ...
                        mean(ParameterdataR(tarP,sites+numberofsites),1) ...
                        std(ParameterdataR(tarP,sites+numberofsites),1);
                        sum(Presultseq(tarP,sites),1)...
                        mean(Presultseq(tarP,sites),1)...
                        std(Presultseq(tarP,sites),1)];

    Mod(:,:,sites-NumTimeValueR) = [sum(ModeldataR(tarM,sites+numberofsites),1) ...
                        mean(ModeldataR(tarM,sites+numberofsites),1) ...
                        std(ModeldataR(tarM,sites+numberofsites),1);
                        sum(Mresultseq(tarM,sites),1)...
                        mean(Mresultseq(tarM,sites),1)...
                        std(Mresultseq(tarM,sites),1)];

    Pre2(:,:,sites-NumTimeValueR) = [sum(ParameterdataR(tarP,sites),1) ...
                        mean(ParameterdataR(tarP,sites),1) ...
                        std(ParameterdataR(tarP,sites),1);
                        sum(Presultamount(tarP,sites),1)...
                        mean(Presultamount(tarP,sites),1)...
                        std(Presultamount(tarP,sites),1)];
    Mod2(:,:,sites-NumTimeValueR) = [sum(ModeldataR(tarM,sites),1) ...
                                mean(ModeldataR(tarM,sites),1) ...
                                std(ModeldataR(tarM,sites),1);
                                sum(Mresultamount(tarM,sites),1)...
                                mean(Mresultamount(tarM,sites),1)...
                                std(Mresultamount(tarM,sites),1)];

    s = sites-NumTimeValueR;
    PR(~tarP,s) = 0;
    MR(~tarM,s) = 0;
    PS(~tarP,s) = 0;
    MS(~tarM,s) = 0;
    [P1(s,:) P2(s,:)]=Surviver(PR(:,s)',PS(:,s)');
    [M1(s,:) M2(s,:)]=Surviver(MR(:,s)',MS(:,s)');
    P1(s,:) = P1(s,:)/P1(s,1);
    P2(s,:) = P2(s,:)/P2(s,1);
    M1(s,:) = M1(s,:)/M1(s,1);
    M2(s,:) = M2(s,:)/M2(s,1);

    PRr2{s}(:,1) = PR2(tarP,s);
    MRr2{s}(:,1) = MR2(tarM,s);
    PSr2{s}(:,1) = PS2(tarP,s);
    MSr2{s}(:,1) = MS2(tarM,s);

    PRr{s}(:,1) = PR(tarP,s);
    MRr{s}(:,1) = MR(tarM,s);
    PSr{s}(:,1) = PS(tarP,s);
    MSr{s}(:,1) = MS(tarM,s);
%
    locP(:,s) = tarP;
    locM(:,s) = tarM;
end

for i = 1:numberofsites
    for j = 1:numberofsites
        PRcorr(i,j) = corr(PR(sum(locP,2)==numberofsites,i),PR(sum(locP,2)==numberofsites,j));
        MRcorr(i,j) = corr(MR(sum(locM,2)==numberofsites,i),MR(sum(locM,2)==numberofsites,j));
        PScorr(i,j) = corr(PS(sum(locP,2)==numberofsites,i),PS(sum(locP,2)==numberofsites,j));
        MScorr(i,j) = corr(MS(sum(locM,2)==numberofsites,i),MS(sum(locM,2)==numberofsites,j));
        if isnan(PRcorr(i,j))
            PRcorr(i,j) = 0;
        end
        if isnan(MRcorr(i,j))
            MRcorr(i,j) = 0;
        end
        if isnan(PScorr(i,j))
            PScorr(i,j) = 0;
        end
        if isnan(MScorr(i,j))
            MScorr(i,j) = 0;
        end
    end
end
```

```
PRbin = PR(sum(locP,2)==numberofsites,:);
MRbin = MR(sum(locM,2)==numberofsites,:);
PSbin = PS(sum(locP,2)==numberofsites,:);
MSbin = MS(sum(locM,2)==numberofsites,:);

eachmonthlength = considermonth(:,2)-considermonth(:,1)+1;
monthlength  = sum(eachmonthlength);

MRmonthly = zeros(Mnumberofyear,monthlength,numberofsites);
MEmonthly = zeros(Mnumberofyear,monthlength,numberofsites);
PRmonthly = zeros(Pnumberofyear,monthlength,numberofsites);
PEmonthly = zeros(Pnumberofyear,monthlength,numberofsites);
PRline = zeros(numberofsites,Pnumberofyear*monthlength);
PEline = zeros(numberofsites,Pnumberofyear*monthlength);
PRliney = zeros(numberofsites,Pnumberofyear);
PEliney = zeros(numberofsites,Pnumberofyear);
MRline = zeros(numberofsites,Mnumberofyear*monthlength);
MEline = zeros(numberofsites,Mnumberofyear*monthlength);
MRliney = zeros(numberofsites,Mnumberofyear);
MEliney = zeros(numberofsites,Mnumberofyear);

MRmonthly2 = zeros(Mnumberofyear,monthlength,numberofsites);
MEmonthly2 = zeros(Mnumberofyear,monthlength,numberofsites);
PRmonthly2 = zeros(Pnumberofyear,monthlength,numberofsites);
PEmonthly2 = zeros(Pnumberofyear,monthlength,numberofsites);
PRline2 = zeros(numberofsites,Pnumberofyear*monthlength);
PEline2 = zeros(numberofsites,Pnumberofyear*monthlength);
PRliney2 = zeros(numberofsites,Pnumberofyear);
PEliney2 = zeros(numberofsites,Pnumberofyear);
MRline2 = zeros(numberofsites,Mnumberofyear*monthlength);
MEline2 = zeros(numberofsites,Mnumberofyear*monthlength);
MRliney2 = zeros(numberofsites,Mnumberofyear);
MEliney2 = zeros(numberofsites,Mnumberofyear);



for y = consideryear(1,1):consideryear(1,2)
    yy = y-consideryear(1,1)+1;
    for sites = NumTimeValueR+1:NumTimeValueR+numberofsites
        i = sites-NumTimeValueR;
        for j = 1:numberofmonth
            startmloc = sum(eachmonthlength(1:j-1));
            for m = considermonth(j,1):considermonth(j,2)
                mloc = startmloc+m-considermonth(j,1)+1;

dploc=~~Ploc&ParameterdataR(:,sites+numberofsites)>=0&ParameterdataR(:,1)==y&ParameterdataR(:,2)=
=m;
    %
                PRmonthly(yy,mloc,i) = sum(ParameterdataR(~~dploc,sites+numberofsites),1);
                PEmonthly(yy,mloc,i) = sum(Presultseq(~~dploc,sites),1);
                PRline(i,yy*monthlength-monthlength+mloc) = PRmonthly(yy,mloc,i);
                PEline(i,yy*monthlength-monthlength+mloc) = PEmonthly(yy,mloc,i);

                PRmonthly2(yy,mloc,i) = sum(ParameterdataR(~~dploc,sites),1);
            PEmonthly2(yy,mloc,i) = sum(Presultamount(~~dploc,sites),1);
            PRline2(i,yy*monthlength-monthlength+mloc) = PRmonthly2(yy,mloc,i);
            PEline2(i,yy*monthlength-monthlength+mloc) = PEmonthly2(yy,mloc,i);
         end
      end
   PRliney(i,yy) = sum(PRmonthly(yy,:,i),2);
   PEliney(i,yy) = sum(PEmonthly(yy,:,i),2);

   PRliney2(i,yy) = sum(PRmonthly2(yy,:,i),2);
   PEliney2(i,yy) = sum(PEmonthly2(yy,:,i),2);
   end
end

for y = consideryear(2,1):consideryear(2,2)
   yy = y-consideryear(2,1)+1;
   for sites = NumTimeValueR+1:NumTimeValueR+numberofsites
      i = sites-NumTimeValueR;
      for j = 1:numberofmonth
         startmloc = sum(eachmonthlength(1:j-1));
         for m = considermonth(j,1):considermonth(j,2)
            mloc = startmloc+m-considermonth(j,1)+1;

dmloc=~~Mloc&ModeldataR(:,sites+numberofsites)>=0&ModeldataR(:,1)==y&ModeldataR(:,2)==m;
            MRmonthly(yy,mloc,i) = sum(ModeldataR(~~dmloc,sites+numberofsites),1);
            MEmonthly(yy,mloc,i) = sum(Mresultseq(~~dmloc,sites),1);
            MRline(i,yy*monthlength-monthlength+mloc) = MRmonthly(yy,mloc,i);
            MEline(i,yy*monthlength-monthlength+mloc) = MEmonthly(yy,mloc,i);

            MRmonthly2(yy,mloc,i) = sum(ModeldataR(~~dmloc,sites),1);
            MEmonthly2(yy,mloc,i) = sum(Mresultamount(~~dmloc,sites),1);
            MRline2(i,yy*monthlength-monthlength+mloc) = MRmonthly2(yy,mloc,i);
            MEline2(i,yy*monthlength-monthlength+mloc) = MEmonthly2(yy,mloc,i);
         end
      end
      MRliney(i,yy) = sum(MRmonthly(yy,:,i),2);
      MEliney(i,yy) = sum(MEmonthly(yy,:,i),2);

      MRliney2(i,yy) = sum(MRmonthly2(yy,:,i),2);
      MEliney2(i,yy) = sum(MEmonthly2(yy,:,i),2);
   end
end
for m = 1:numberofmonth

PRamount(:,:,m)=Presultamount(Presultamount(:,2)>=considermonth(m,1)&Presultamount(:,2)<=considerm
onth(m,2),4:end);

MRamount(:,:,m)=Mresultamount(Mresultamount(:,2)>=considermonth(m,1)&Mresultamount(:,2)<=conside
rmonth(m,2),4:end);

Ramount(:,:,m)=[Presultamount(Presultamount(:,2)>=considermonth(m,1)&Presultamount(:,2)<=considerm
onth(m,2),4:end);

Mresultamount(Mresultamount(:,2)>=considermonth(m,1)&Mresultamount(:,2)<=considermonth(m,2),4:end
)];
end
```

```
function [seq,gsc,s,largeW,largeG]=
NHMMgenerate_general(X,a,b,Alpha,Beta,fs,numberofsites,spatialmodeltype,independency,initial)
L = size(X,2);
seq = zeros(numberofsites,L);
s = zeros(1,L);
largeW=[];largeG=[];
numStates = size(a,1);
checkTr = size(a,2);
if checkTr ~= numStates
   error('stats:hmmgenerate:BadTransitions',...
        'TRANSITION matrix must be square.');
end

   n = 2^numberofsites;




binarytable_ch = dec2bin((0:n-1)',numberofsites);
binarytable_in = double(binarytable_ch-'0');

randvalsS = rand(1,L);

zeroBeta = zeros(size(Beta));
e = f_B(Alpha,zeroBeta,ones(numberofsites,1));
IndB = zeros(numStates,numberofsites,2);
IndB(:,:,2) = e(:,:);
IndB(:,:,1) = 1-IndB(:,:,2);
IndB = cumsum(IndB,3)./repmat(sum(IndB,3),[1 1 2]);
BinaryG = 2.^(numberofsites-1:-1:0);
B_distribution =...
   BinaryTable(spatialmodeltype,Alpha,Beta,0,initial);
CumsumB_distribution = cumsum(B_distribution,1);
B_distribution2 =...
   BinaryTable(spatialmodeltype,Alpha,zeroBeta,1,initial);
%
if spatialmodeltype ~= 1 && independency == false
   M = 500;

   largeW = zeros(numStates,M);
   randset = rand(numStates,numberofsites,M);

   largeG = repmat(IndB(:,:,1),[1 1 M]) < randset;
   loc = sum(repmat(BinaryG,[numStates 1 M]).*largeG,2)+1;
   for ss = 1:numStates
      largeW(ss,:) = (B_distribution(loc(ss,:,:),ss)./B_distribution2(loc(ss,:,:),ss))';
      largeW(ss,:) = largeW(ss,:)/sum(largeW(ss,:),2);
      largeW(ss,:) = cumsum(largeW(ss,:),2);
   end
else
```

```
    largeW=[];largeG=[];
end

gs = ones(1,numStates);


for count = 1:L
    A = f_Ao(a,b,X(:,count));

    gs(1,:) = fs(1,:)*A(:,:);
    gs(1,:) = fs(1,:)*A(:,:);
    gsc = cumsum(gs(1,:),2);
    gsc = gsc/gsc(1,numStates);

    s(count) = 1;
    for i = 1:numStates
        if gsc(i)>=randvalsS(count)
            s(count)=i;
            break
        end
    end


    fsc = e(s(count),:);

    seq(:,count) = 1;
    if spatialmodeltype == 1 || independency == true
        for i = 1:numberofsites
            if fsc(1,i)>=rand();
                seq(i,count)=2;
            end
        end
    elseif spatialmodeltype ~= 1 && independency == false
        for site = 1:numberofsites
            argmin = find(largeW(s(count),:) > rand());
            seq(site,count) = largeG(s(count),site,argmin(1,1))'+1;
        end
    end

    emit = B_distribution(BinaryG*(seq(:,count)-1)+1,:)';

    fs(1,:) = gs(1,:).*emit(:,1)';
end
```