AN ABSTRACT OF THE DISSERTATION OF

Travis Moore for the degree of Doctor of Philosophy in Computer Science presented on August 25, 2021.

Title: Efficient Algorithms for Robust Spatiotemporal Data Analysis

Abstract approved: ______________________________

Weng-Keen Wong

Many large-scale data analysis applications involve data that can vary over both time and space. Often the primary goal of analyzing spatiotemporal data is identifying trends, movements, and sudden changes with respect to time, location, or both. This can include a variety of applications in economics (housing prices, unemployment, job movement, etc), city planning (traffic, power consumption, resource allocation, etc), and ecology (migration patterns, species variety, habitat change, etc). Like many domains, one of the major challenges of spatiotemporal data is dealing with noise and missing or untrustworthy observations. These uncertainties make it difficult to ascertain the distinct roles that changes in time and location have on the data. To this end, I have developed two different approaches for dealing with data uncertainty in different spatiotemporal applications. The first approach, dubbed the Quantile Scan algorithm, makes use of quantile regression to more accurately identify anomalous regions in the data. The flexibility of this framework allows 'anomalies' to be defined with respect to any quantile of interest. I develop a version of the Quantile Scan algorithm for analyzing spatial, and spatiotemporal data. The second approach is a unique variation of Collective Graphical Models (CGMs) to incorporate multiple views of the data. This multiview model learns and leverages shared information between the views to better compensate for missing observations. Both the Quantile Scan and Multiview CGM algorithms improve accuracy and robustness on noisy data without sacrificing runtime. The speed and accuracy of these models is demonstrated on a variety of synthetic and real-world datasets, compared against existing algorithms.

Efficient Algorithms for Robust Spatiotemporal Data Analysis

by

Travis Moore

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented August 25, 2021
Commencement June 2022

ACKNOWLEDGEMENTS

This research was funded in part by NSF grant CCF-1521687.

I would like to thank our collaborators from eButterfly (Katy Prudic, Max Larrivee, Jeffrey Oliver and Jeremy Kerr) and eBird (Daniel Fink, Chris Wood, and Frank La Sorte), who helped in evaluating the performance of the algorithms presented.

A sincere thanks to the ARCS Foundation, for their scholar award from 2013 to 2016.

Finally, I would like to extend my personal thanks to my spouse and parents, for their unwavering support even during the most difficulty of times.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF TABLES

LIST OF ALGORITHMS

Chapter 1: Introduction

A common goal in many machine learning and data analysis tasks is the modeling and prediction of real world processes. These processes invariably change with respect to time and, if the analysis is at a large enough scale, over location. This spatiotemporal dynamic is evident in applications in: economics, such as analyzing differences in housing prices, unemployment, and job availability; city planning, where changes in traffic, power consumption, and resource allocation are of interest; and ecology, where season and habitat affect migration and species variety. In most of these domains, both space and time shape the dynamics of the data in different ways.

Within the vast umbrella of spatiotemporal analysis, this work focuses on the issue of dynamics: how values change over time and across different locations. As is the case in many other machine learning tasks, this goal is often impeded by noisy or missing observations within the data. The scale of this type of data, which is often collected over years and across multiple states or counties, makes this kind of uncertainty fairly common. This can be due to the cost limitations of data collection, inherent noise in the observation process, or the aggregation over individual processes. Accounting for this noise is imperative in order to properly ascertain the distinct influences that time and space have on the underlying systems.

In this work, I present two different paradigms for dealing with noisy spatiotemporal data within two different analysis frameworks. The first framework is the Spatial Scan Statistic Kulldorff [1997]. The Spatial Scan Statistic (SSS) is a sliding window search procedure that can be used over space or time aspects of a dataset to detect regions of significant change. Traditionally this significance is defined in terms of aggregate or mean values within the search windows. However, the use of measured quantiles instead of means has been shown to be more robust to data noise and outliers, particularly in regression analysis Rousseeuw and Leroy [1987]. To this end I derive the Quantile Spatial Scan Statistic (QSSS) and Quantile Snapshot Scan (Qsnap) algorithms, which perform spatial and spatiotemporal change detection over measured quantiles. Both algorithms make use of unique update procedures that reduce the normally intractable runtime cost of computing quantiles for each scan region by an order of magnitude. The robustness and versatility of both algorithms are demonstrated on real and synthetic datasets.

The second spatiotemporal framework is Collective Graphical Models (CGMs) Sheldon and Dietterich [2011]. CGMs model the group transitions of a population in a graphical model, based on an individual model but without individual observations. They are a natural fit for the application of population tracking Iwata et al. [2017], Iwata and Shimizu [2019], where individual observations are not available due to privacy constraints (with people) or sensing limitations (with animals). Without the individual data, CGMs reason based on aggregate statistics, making the model under-specified in general. To help combat this missing information, I develop a hierarchical multiview CGM model. This multiview CGM identifies and accounts for the shared information between two populations moving in the same space and time frame. The shared information is then leveraged as an additional factor to improve the predictive accuracy for both populations. The improved accuracy of this method is demonstrated on multiple synthetic and real-world datasets. I also show empirically that the runtime of the multiview algorithm is faster than the baseline for large graphs.

In the following three chapters I will present the QSSS, Qsnap, and the multiview CGM algorithms. The details of each algorithm are encapsulated within their chapters, including methodology, related work, and experiment results. A holistic conclusion will then follow in Chapter 5.

Chapter 2: Quantile Spatial Scan Statistic

2.1 Introduction

Spatial data analysis often involves finding spatial regions that are different from the surrounding area. For example, epidemiologists are interested in finding regions with an unusually high incidence of disease while criminologists are interested in identifying crime hotspots. The spatial scan statistic (SSS) [Kulldorff, 1997] is a widely used technique to discover unusual regions from a Bernoulli or Poisson point process. The SSS searches over a given set of regions, scoring each region according to how a quantity of interest (e.g. the disease rate) inside the region differs from outside the region. Finally, the SSS computes the p-value of the highest scoring region using a randomization test.

Many spatial data sets, however, are more complex than point processes, which focus on the spatial locations of the data. Real-world spatial data sets from domains such as citizen science biodiversity monitoring and real estate associate a response value with each point as well as a set of covariates (called features by machine learning researchers). For example, in a real estate data set, each data point has a location, a sale price, and associated features such as square footage, number of bedrooms, age, etc. Formally, we represent the ith data point of dataset $\boldsymbol{D}$ as a tuple $(Y_i, X_{i,1}, \ldots, X_{i,p}, L_{i,1}, \ldots, L_{i,d})$, where Y_i is a continuous response, $(X_{i,1}, \ldots, X_{i,p})$ are the p covariates and $(L_{i,1}, \ldots, L_{i,d})$ are spatial coordinates in d-dimensions; for simplicity, we assume $d = 2$. In later sections, we will refer to the data as $\boldsymbol{D} = \{\boldsymbol{Y}, \boldsymbol{X}, \boldsymbol{L}\}$ to represent the distinct aspects of response, covariates and locations.

We can follow the SSS framework to find unusual regions in this more complex setting. For each region, we fit a model that captures the relationship between the features and the response variable. Then, we use a scoring function to compare the models from inside the region versus outside the region, using a hypothesis test that compares the means of the models. While such an approach seems reasonable, there are two shortcomings. First, the approach is not robust as the mean is well known to be vulnerable to outliers and extreme values [Rousseeuw and Leroy, 1987]. Second, many real-world tasks compare spatial regions using other parts of their distributions besides the mean. For instance, a real-estate agent interested in high-end homes may want to compare regions based on the 90th percentile of the sale price distribution. To overcome both of these problems, we develop a novel method for comparing quantiles of spatial regions.

We modify the proposed SSS variant by fitting quantile regression models to the 'inside' and 'outside' regions. Unfortunately, this naive approach is computationally expensive; fitting a quantile regression requires a linear program and this step would be required in the inner loop of the algorithm. To make the algorithm efficient, we replace the traditional likelihood ratio test with the rank test, which is a non-parametric hypothesis test that avoids the need to fit quantile regressions to the data inside the region. However, performing a rank test from scratch every time we score a new region is also computationally expensive. Instead, we develop an incremental version of the rank test that allows the rank test from a smaller region to be updated when the region is grown to include more spatial data points. Lastly, we show how the algorithm can be adapted to perform one-sided hypothesis tests with only a minor increase in runtime[1].

The contributions of our work are as follows. First, we introduce the Quantile Spatial Scan Statistic (QSSS), which discovers unusual regions for continuous spatial data with covariates. The comparison between regions to determine unusualness is based on a comparison of the τ-th quantile of the response variable distributions. This algorithm is also robust to outliers, unlike an analogous algorithm that makes comparisons based on the mean of a region. Second, we show how to make the QSSS over an order of magnitude faster than a naive implementation by introducing an incremental update to the rank test. This update is exact and not an approximation. Third, we show how the rank test hypothesis of the algorithm can be adjusted to a one-sided version, with minimal impact on runtime. Finally, we evaluate the QSSS on simulated data and also show interesting results from case studies on three real-world datasets.

2.2 Background

We first present a brief introduction to the Spatial Scan Statistic, Quantile Regression and the Rank test for Quantile Regression as these techniques form the foundation for

[1]An earlier version of this work without the one sided test was published in UAI [Moore and Wong, 2018]

the QSSS. In addition, we discuss connections to existing work in these three areas.

2.2.1 The Spatial Scan Statistic

The Spatial Scan Statistic, introduced by Kulldorff [1997] is a widely used approach for finding anomalous regions. The algorithm considers a set of data points, each with a location and response value, and attempts to find the most anomalous contiguous region.

The original SSS used a scanning window in the shape of a circle to discover unusual regions. While in theory the search should be over all circular regions, in practice it is often limited to circles with centers determined by a fixed grid superimposed on the spatial area. For each region, a hypothesis test is calculated, scoring the region based on how different it is from the rest of the data. The likelihood ratio test, performed with an assumption on the distribution function of the data, is a common choice of hypothesis test. The SSS then returns the best scoring region.

Due to multiple hypothesis testing, we cannot interpret the score from the likelihood ratio test as a true p-value. Instead, we estimate the p-value through a randomization test. In each replication of the randomization test, we maintain the same underlying population as the original problem, but generate events assuming a uniform probability. Then, the search for the best scoring region is performed. The process is repeated for R replications to produce an empirical distribution which determines how likely it is to obtain the score of the best scoring region.

Many researchers have extended the original SSS approach, including using scanning windows that are arbitrarily shaped [Duczmal and Assuncao, 2004] and incorporating mobility patterns [Lan et al., 2014]. We point out that performing a quantile-based comparison results is a fundamentally different type of optimization problem and past work on speeding up the SSS (eg. [Neill and Moore, 2004, Neill, 2012]) is not readily applicable. Finally, Moore and Wong [2015] use the SSS to find species-rich hotspots, but they do not compare quantiles of distributions.

Another spatial scan variant that can perform comparisons on arbitrary quantiles of the underlying distribution is the Treatment Effect Spatial Scan (TESS) [McFowland et al., 2018]. In the TESS algorithm, the data is partitioned into two subsets denoted as ‘control’ and ‘treatment’. The algorithm learns a distribution model on the control set, and then uses the model to compute a p-value for every datapoint in the treatment set.

The original paper uses a discrete empirical distribution to fit these p-values, but any data appropriate model can be used.

With the treatment data condensed into a vector of p-values, the TESS algorithm uses the subset scan framework to find the subset of the treatment group where the p-values have the greatest deviation from the expected distribution. For a given subset $\boldsymbol{C}$ and quantile τ, the algorithm performs the likelihood ratio test

$$T = N(\boldsymbol{C}) KL\left(\frac{N_\tau(\boldsymbol{C})}{N(\boldsymbol{C})}, \tau\right) \tag{2.1}$$

where $N(\boldsymbol{C})$ is the number of points in $\boldsymbol{C}$, $N_\tau(\boldsymbol{C})$ is the number of points in $\boldsymbol{C}$ less than τ, and $KL()$ is the KL-divergence function. The authors suggest performing the test for a range of values of τ, and keeping the most significant result. The reduction of the treatment set to p-values makes the algorithm very fast to run, as these values can be computed independent of the subset search. The accuracy of the algorithm is tied to its ability to learn an accurate model from the control set.

The ability of TESS to detect significant regions at a given quantile makes it the closest related work to our QSSS algorithm and we compare against it in our experiments.

2.2.2 Quantile Regression

Suppose we have a continuous random variable Y with distribution function $F(Y) = P(Y \leq y)$. The τ-th quantile $q(\tau)$, with $0 < \tau < 1$, is defined as $q(\tau) = F^{-1}(\tau) = \inf_y\{F(y) \geq \tau\}$. For example, when $\tau = 0.5$, we get the median. Given a dataset $Y_1, \ldots, Y_n$, the τ-th sample quantile $\hat{q}(\tau)$, can be computed by solving the optimization problem:

$$\hat{q}(\tau) = \operatorname*{argmin}_q \sum_{i=1}^{n} \rho_\tau(Y_i - q) \tag{2.2}$$

where $\rho_\tau(r) = r(\tau - I(r < 0))$.

Quantile regression, introduced by Koenker and Bassett [1978], fits a regression to the conditional τ-th quantile of the response variable. Given a dataset $\boldsymbol{D} = \{(Y_1, \boldsymbol{X}_1), \ldots, (Y_n, \boldsymbol{X}_n)\}$ where Y_i is the response variable and $\boldsymbol{X_i}$ are the covariates, fitting a quantile regression involves solving:

$$\hat{\boldsymbol{\beta}}(\tau) = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \sum_{i=1}^{n} \rho_\tau(Y_i - \boldsymbol{X_i\beta}) \tag{2.3}$$

The solution $\hat{\boldsymbol{\beta}}(\tau)$ produces a conditional quantile function $Q_Y(\tau|\boldsymbol{X} = \boldsymbol{x}) = \boldsymbol{x'}\hat{\boldsymbol{\beta}}(\tau)$, similar to how a standard regression produces the conditional mean when the coefficients are multiplied with the covariate values.

Quantile regression is a useful tool for analyzing specific parts of a distribution. It can model the data extremes by setting τ close to either 1 or 0, or it can reduce the influence of these points by modeling τ close to 0.5.

Koenker and Machado [1999] introduce three methods for comparing two quantile regression models, based on Wald's test [Wald, 1943], the Likelihood Ratio test [Wilks, 1932], and the Rank test [Rao, 1948]. Mood's median test [Mood, 1950] can also be adapted to perform a fast, low power comparison at a given quantile. Any of these methods are still usable when the covariate set $\boldsymbol{X}$ is empty by using the quantiles of $\boldsymbol{Y}$. We use the Rank test, as it can be implemented without repeatedly re-estimating the quantile regression coefficients for each data subset, thereby reducing its computation time without sacrificing power. In the following section we explain the Rank test for quantile regression.

2.2.3 Rank Test for Quantile Regression

Let the regression model for the τth quantile have the form $\boldsymbol{Y} = \boldsymbol{X\beta_1} + \boldsymbol{\tilde{X}\beta_2}$ where each row $\boldsymbol{X}_i$ corresponds to a data point. For a given data subset $\boldsymbol{C} \subseteq \boldsymbol{D}$, $\boldsymbol{\tilde{X}_i} = \boldsymbol{X_i}$ if $\boldsymbol{X_i} \in \boldsymbol{C}$ and $\boldsymbol{\tilde{X}_i} = \boldsymbol{0}$ (we use boldface to indicate a vector of zeros) if $\boldsymbol{X_i} \notin \boldsymbol{C}$. This model will simultaneously fit a regression to $\boldsymbol{C}$ and $\boldsymbol{D} \setminus \boldsymbol{C}$. In the spatial scan context $\boldsymbol{C}$ is the region inside our circle and $\boldsymbol{D} \setminus \boldsymbol{C}$ is the region outside. The goal is then to test the null hypothesis $H_0 : \boldsymbol{\beta_2} = \boldsymbol{0}$ against the alternative $H_1 : \boldsymbol{\beta_2} \neq \boldsymbol{0}$ to see if the subset $\boldsymbol{C}$ is sufficiently different from the full distribution of $\boldsymbol{D}$.

One method for performing this hypothesis test is to use the score test [Rao, 1948], which takes the form $T = \boldsymbol{S'M}^{-1}\boldsymbol{S}$. Here $\boldsymbol{S}$ is a $p \times 1$ score vector and $\boldsymbol{M}$ is the $p \times p$ information matrix. For the hypothesis test defined above, $\boldsymbol{M} = n^{-1}(\boldsymbol{\tilde{X}} - \boldsymbol{H\tilde{X}})'(\boldsymbol{\tilde{X}} -$

$\boldsymbol{H\tilde{X}}$), where $\boldsymbol{H} = \boldsymbol{X}(\boldsymbol{X'X})^{-1}\boldsymbol{X'}$. $\boldsymbol{H\tilde{X}}$ is the projection of $\boldsymbol{\tilde{X}}$ onto the space spanned by $\boldsymbol{X}$, and subtracting it from $\boldsymbol{\tilde{X}}$ removes the influence of $\boldsymbol{\beta}_1$ from the test, allowing it to focus on $\boldsymbol{\beta}_2$.

The score vector is the gradient of the log-likelihood. In cases where the likelihood is not explicitly known, $\boldsymbol{S}$ can be approximated by a rank-score process. This assigns a ranking to each datapoint as a substitute for their probability, and changes the score test into a rank test. For applications to quantile regression, this ranking can be assigned with respect to the quantile of interest. Gutenbrunner and Jurecková [1992] and Machado and Silva [2002] use the following rank-score approximation for $\boldsymbol{S}$ for quantile regression.

$$\boldsymbol{S} = n^{-1/2}(\boldsymbol{\tilde{X}} - \boldsymbol{H\tilde{X}})'\boldsymbol{\hat{b}} \tag{2.4}$$

The $n \times 1$ vector $\boldsymbol{\hat{b}}$ is the rank-score for each datapoint based on $\boldsymbol{\beta}_1$, the quantile regression model under the null hypothesis. Each point is assigned a ranking value between τ and $\tau - 1$ depending on whether the point falls above, below, or on the regression plane defined by $\boldsymbol{\beta}_1$. More specifically, $\boldsymbol{\hat{b}}_i = \boldsymbol{\hat{a}}_i - (1-\tau)$ where $\boldsymbol{\hat{a}}_i = 1$ if $\boldsymbol{x_i\beta_1} > 0$, $\boldsymbol{\hat{a}}_i = 0$ if $\boldsymbol{x_i\beta_1} < 0$, and $0 \leq \boldsymbol{\hat{a}}_i \leq 1$ if $\boldsymbol{x_i\beta_1} = 0$, subject to the constraint $\boldsymbol{X'\hat{a}} = (1-\tau)\boldsymbol{X'1}$. The values $\boldsymbol{\hat{a}}$ are the dual solution of the quantile regression optimization, which can be computed in parallel with $\boldsymbol{\beta}_1$.

$\Psi^2 = \tau(1-\tau)$ is included to normalize the rank-score vector, giving us

$$T = \boldsymbol{S'M}^{-1}\boldsymbol{S}/\Psi^2 \tag{2.5}$$

The test statistic T follows a Chi-squared distribution under the null hypothesis with p degrees of freedom. The rank test has the same asymptotic power as the analogous Wald and likelihood ratio tests, but the rank test does not require an estimation of the parameters under H_1. This aspect of the rank test is critical for our algorithm as it greatly reduces the computation load for the algorithm's inner loop.

2.3 Related Work

Having presented the background material needed to understand the QSSS, along with techniques related to the background material, we now discuss other related work. A large body of work that is seemingly related to our task has focused on producing disease maps that illustrate how disease cases vary across space (eg. [Best et al., 2005]). Researchers have also investigated spatial quantile regression (eg. [Reich et al., 2011, Macmillan, 2013, King and Song, 2019, Almanjahie et al., 2019]). These modeling approaches generally produce a probabilistic surface, which results in a useful visualization but does not directly solve our goal of identifying specific unusual regions. Achieving this goal requires a human to inspect the probabilistic surface, manually segment it into unusual regions and rank these regions according to some unusualness criterion. This human intervention is not desirable when the spatial region is large and also if the goal is to create an automated monitoring system. Our QSSS algorithm essentially automates these steps in a computationally efficient manner.

2.4 Methodology

We start with a high level overview of our QSSS. Given a dataset $\boldsymbol{D} = \{\boldsymbol{Y}, \boldsymbol{X}, \boldsymbol{L}\}$, and a list of starting locations $\boldsymbol{P}$, the QSSS searches over circular areas in $\boldsymbol{L}$, beginning at each starting location in $\boldsymbol{P}$ and growing the regions one data point at a time, starting from some minimum number of points. The regions are grown as circles of increasing radius. Each time the region grows, we calculate its test statistic using our Incremental Rank test (Section 2.4.1). Once the region cannot be grown any larger, or reaches a maximum size, we move on to the next starting point in $\boldsymbol{P}$. After all starting points have been exhausted, an adjusted p-value is calculated for the region with the highest test statistic using a Gumbel correction (Section 2.4.3). We chose the Gumbel correction because it is much faster than the traditional randomization test. If the adjusted p-value is significant then the algorithm returns the region, otherwise it says that no significant region was found.

2.4.1 Faster Rank Test for QSSS

In the QSSS framework, the Rank test needs to be performed for every circular subset $\boldsymbol{C} \subseteq \boldsymbol{D}$. We can choose a set of starting points (either each data point or a grid formed over $\boldsymbol{L}$) for the regions and grow each one, recalculating our hypothesis test each time the region overlaps a new data point. The inclusion of a new data point i into the region will change the ith row of $\tilde{\boldsymbol{X}}$ from a row of zeros to the ith row of $\boldsymbol{X}$. Under the framework of the Rank test, $\boldsymbol{X}$, $\boldsymbol{H}$, and $\hat{\boldsymbol{b}}$ will be the same for every choice of region $\boldsymbol{C}$. Thus our only task is to update T as $\tilde{\boldsymbol{X}}$ changes.

The primary bottlenecks in updating T are in updating $\boldsymbol{S}$ and recomputing $\boldsymbol{M}^{-1}$. $\boldsymbol{M}^{-1}$ can be updated incrementally using applications of the Sherman-Morrison formula [Sherman and Morrison, 1950], but a more efficient update can be performed by leveraging the special structure of T. Note that we can re-write T as

$$T = \hat{\boldsymbol{b}}\boldsymbol{Z}(\boldsymbol{Z'Z})^{-1}\boldsymbol{Z'}\hat{\boldsymbol{b}}/\Psi^2 \tag{2.6}$$

where $\boldsymbol{Z} = \tilde{\boldsymbol{X}} - \boldsymbol{H}\tilde{\boldsymbol{X}}$. $\boldsymbol{Z}(\boldsymbol{Z'Z})^{-1}\boldsymbol{Z'}$ is by definition a projection matrix onto the space of $\boldsymbol{Z}$. If we let $\boldsymbol{U}$ be an n x p orthonormal column basis of $\boldsymbol{Z}$, then

$$T = \hat{\boldsymbol{b}}\boldsymbol{UU'}\hat{\boldsymbol{b}}/\Psi^2 \tag{2.7}$$

The inverse in Equation 2.6 is a normalization term. Since $\boldsymbol{U}$ is already normalized, the formulation of Equation 2.7 allows us to forgo the matrix inverse calculation. To utilize this efficient calculation of T, we will need to calculate an orthonormal basis $\boldsymbol{U}^t$ of $\boldsymbol{Z}^t$ for every iteration t. Rather than calculate this basis from scratch every iteration, we can dramatically improve runtime by performing incremental updates to $\boldsymbol{U}$ as $\tilde{\boldsymbol{X}}$ changes.

2.4.1.1 Incremental Orthogonalization of Rank Test

Our goal is to take an existing orthonormal basis at iteration t (i.e. $\boldsymbol{U}^t$), and calculate $\boldsymbol{U}^{t+1}$ based on the (small) change in $\tilde{\boldsymbol{X}}$ when a new data point is added to the inside region. This can be achieved by using a modified rank one QR update algorithm. To show this, we will prove that $\boldsymbol{Z}^{t+1}$ is a rank one update of $\boldsymbol{Z}^t$, illustrate an existing QR factorization rank one update algorithm from Golub and Loan [2012], and then show how that algorithm can be modified to run much faster for our problem.

$\boldsymbol{Z}$ Update: The first step is to show that the update of $\boldsymbol{Z}$ between iteration t and $t+1$ is a rank one update. Specifically:

Theorem 1

Let $\boldsymbol{Z}^t$ be the value of matrix $\boldsymbol{Z}$ at iteration t of the spatial scan expanding window search. At iteration $t+1$, when the ith data point of $\boldsymbol{X}$ is added to the inside region, then

$$\boldsymbol{Z}^{t+1} = \boldsymbol{Z}^t + \boldsymbol{A}_i \tag{2.8}$$

where $\boldsymbol{A}_i$ is a rank one matrix that can be decomposed into known vectors $\boldsymbol{A}_i = \boldsymbol{a}_i'\boldsymbol{b}_i$.

Proof: Let $\boldsymbol{K}_{[n\times n]}$ be a row selector matrix, where $K_{[j,j]} = 1$ if point j is in $\boldsymbol{C}$, and all other values are zero. For a current region $\boldsymbol{C}^t$ at iteration t, $\tilde{\boldsymbol{X}} = \boldsymbol{K}^t\boldsymbol{X}$. If we add point i to $\tilde{\boldsymbol{X}}$ during iteration $t+1$, then this is equivalent to changing element (i,i) of $\boldsymbol{K}^t$ from 0 to 1. We can express this change as a matrix sum $\boldsymbol{K}^{t+1} = \boldsymbol{K}^t + \boldsymbol{K}_i$ where $\boldsymbol{K}_i$ is zero except for element (i,i), which equals 1. This allows us to decompose the change in $\boldsymbol{Z}^{t+1}$ as follows:

$$\begin{aligned}
\boldsymbol{Z}^{t+1} &= (\boldsymbol{K}^t + \boldsymbol{K}_i)\boldsymbol{X} - \boldsymbol{H}(\boldsymbol{K}^t + \boldsymbol{K}_i)\boldsymbol{X} && (2.9)\\
&= \boldsymbol{Z}^t + \boldsymbol{K}_i\boldsymbol{X} - \boldsymbol{H}\boldsymbol{K}_i\boldsymbol{X} && (2.10)\\
&= \boldsymbol{Z}^t + (\boldsymbol{e}_i - \boldsymbol{H}_i)'\boldsymbol{X}_i && (2.11)
\end{aligned}$$

where $\boldsymbol{e}_i$ is the ith unit basis vector of size $1 \times n$. $\boldsymbol{H}$ is a symmetric matrix, so we use the row vector $\boldsymbol{H}_i$ to keep our notation consistent. In Equation 2.10, note that $\boldsymbol{Z}^t = \boldsymbol{K}^t\boldsymbol{X} - \boldsymbol{H}\boldsymbol{K}^t\boldsymbol{X}$. In Equation 2.11 we have reduced the update to $\boldsymbol{Z}^t$ to the product of a column and row vector i.e. $(\boldsymbol{e}_i - \boldsymbol{H}_i)'\boldsymbol{X}_i$. This means that the matrix added to $\boldsymbol{Z}^t$ has a rank of one. $\square$

We can use this special update structure in an algorithm to find $\boldsymbol{U}^{t+1}$ efficiently, by incrementally updating the QR factorization of $\boldsymbol{Z}$.

QR Rank One Update Algorithm: If the QR factorization of $\boldsymbol{Z}^t$ is known, where $\boldsymbol{R}$ is an upper triangular matrix and $\boldsymbol{Q} = \boldsymbol{U}^t$ is an orthonormal column basis, then the factorization for $\boldsymbol{Z}^{t+1}$ can be found with the rank one update algorithm detailed in section 12.5.1 of Golub and Loan [2012]. This algorithm lets us find the factorization $\boldsymbol{Z}^{t+1} = \boldsymbol{Q}^{t+1}\boldsymbol{R}^{t+1}$ using the previous factorization $\boldsymbol{Z}^t = \boldsymbol{Q}^t\boldsymbol{R}^t$, giving us $\boldsymbol{U}^{t+1} = \boldsymbol{Q}^{t+1}$

for our update to the test statistic T.

Let $\boldsymbol{v} = \boldsymbol{e}_i - \boldsymbol{H}_i$. We start by refactoring the update as

$$\boldsymbol{Z}^{t+1} = \boldsymbol{Q}^t\boldsymbol{R}^t + \boldsymbol{v'}\boldsymbol{X}_i = \boldsymbol{Q}^t(\boldsymbol{R}^t + \boldsymbol{w'}\boldsymbol{X}_i) \tag{2.12}$$

where $\boldsymbol{w'} = (\boldsymbol{Q}^t)^{-1}\boldsymbol{v'} = (\boldsymbol{Q}^t)'\boldsymbol{v'}$. Our goal is to turn $\boldsymbol{Z}^{t+1}$ into the product of an orthonormal matrix (which will be $\boldsymbol{Q}^{t+1}$) and an upper triangular matrix to be produced from $(\boldsymbol{R}^t + \boldsymbol{w'}\boldsymbol{X}_i)$. $\boldsymbol{R}^t$ is already upper triangular, leaving $\boldsymbol{w'}\boldsymbol{X}_i$ to be converted.

Givens rotations are a common tool used in QR factorization to convert matrices into a product of an orthonormal matrix and an upper triangular matrix. A Given's rotation is a matrix multiplication that results in the scaled difference of two rows of the multiplied matrix, similar to a linear algebra elimination step. Givens rotations have a special form that makes them particularly useful in triangularizaiton and QR factorization. They can be represented as a rotation matrix $\boldsymbol{G(i,j,\theta)}$, where $G_{[k,k]} = \cos\theta$ for $k = i, j$, $G_{[k,k]} = 1$ for $k \neq i, j$, $G_{[i,j]} = -G_{[j,i]} = -\sin\theta$, and all other entries are zero. θ, the angle of rotation, can be set such that the product of $\boldsymbol{G}$ and a given vector has a zero at index j, making it useful in zeroing out matrix elements to produce a triangular result. We can also see from this definition that Givens rotation matrices are orthonormal, and the inverse of a Givens rotation matrix is also its transpose. Thus, given a matrix $\boldsymbol{X}$, a set of Givens rotations $\boldsymbol{G} = \boldsymbol{G}_1 \ldots \boldsymbol{G}_m$ can be constructed such that $\boldsymbol{GX}$ is triangular. It then follows that $\boldsymbol{G'}$ is an orthonoral matrix, and $\boldsymbol{G}'(\boldsymbol{GX}) = \boldsymbol{X}$ is a QR factorization of $\boldsymbol{X}$.

We can compute a set of Givens rotation matrices $\boldsymbol{J}^1, ..., \boldsymbol{J}^{n-1}$ such that $(\boldsymbol{J}^1)'...(\boldsymbol{J}^{n-1})'\boldsymbol{w'} = ||\boldsymbol{w}||\boldsymbol{e'_1}$. This will ensure that $||\boldsymbol{w}||\boldsymbol{e'_1}\boldsymbol{X}_i$ is upper triangular, since only the first row of the product is non-zero. To maintain equality with the original formula, we must include the transpose of every Givens rotation we introduce. This results in

$$\boldsymbol{Z}^{t+1} = \boldsymbol{Q}^t\boldsymbol{J}^{n-1}...\boldsymbol{J}^1(\boldsymbol{J}^1)' \ldots (\boldsymbol{J}^{n-1})'(\boldsymbol{R}^t + \boldsymbol{w'}\boldsymbol{X}_i) \tag{2.13}$$

$$= \boldsymbol{Q}^t\boldsymbol{J}^{n-1}...\boldsymbol{J}^1(\boldsymbol{A} + ||\boldsymbol{w}||\boldsymbol{e}'_1\boldsymbol{X}_i) \tag{2.14}$$

where $\boldsymbol{A} = (\boldsymbol{J}^1)'...(\boldsymbol{J}^{p-1})'\boldsymbol{R}$, which is an upper Hessenberg matrix. Upper Hessenberg matrices are upper triangular matrices with one additional non-zero entry below the diagonal of each column. They can be turned into upper triangular matrices with a linear number of Givens rotations.

$$\boldsymbol{Z}^{t+1} = \boldsymbol{Q}^t \boldsymbol{J}^{n-1} ... \boldsymbol{J}^1 (\boldsymbol{A} + ||\boldsymbol{w}|| \boldsymbol{e}_1' \boldsymbol{X}_i) \tag{2.15}$$
$$= \boldsymbol{Q}^t \boldsymbol{J}^{n-1} ... \boldsymbol{J}^1 \tilde{\boldsymbol{A}} \tag{2.16}$$

$\tilde{\boldsymbol{A}}$ is also an upper Hessenberg matrix. As such, we can find another set of Givens rotation matrices $\boldsymbol{G}^1, ..., \boldsymbol{G}^{p-1}$ such that $(\boldsymbol{G}^{p-1})'...(\boldsymbol{G}^1)'\tilde{\boldsymbol{A}} = \tilde{\boldsymbol{R}}$, where $\tilde{\boldsymbol{R}}$ is an upper triangular matrix.

$$\boldsymbol{Z}^{t+1} = \boldsymbol{Q}^t \boldsymbol{J}^{n-1} ... \boldsymbol{J}^1 \tilde{\boldsymbol{A}} \tag{2.17}$$
$$= \boldsymbol{Q}^t \boldsymbol{J}^{n-1} ... \boldsymbol{J}^1 \boldsymbol{G}^1, ..., \boldsymbol{G}^{p-1} (\boldsymbol{G}^{p-1})' ... (\boldsymbol{G}^1)' \tilde{\boldsymbol{A}} \tag{2.18}$$
$$= \boldsymbol{Q}^t \boldsymbol{J}^{n-1} ... \boldsymbol{J}^1 \boldsymbol{G}^1, ..., \boldsymbol{G}^{p-1} \tilde{\boldsymbol{R}} \tag{2.19}$$

This completes the factorization update from Golub and Loan [2012], with $\boldsymbol{Q}^{t+1} = \boldsymbol{Q}^t \boldsymbol{J}^{n-1} ... \boldsymbol{J}^1 \boldsymbol{G}^1 ... \boldsymbol{G}^{p-1}$ and $\boldsymbol{R}^{t+1} = \tilde{\boldsymbol{R}}$.

Efficient QR Rank One Update Algorithm: The previous algorithm is not ideal in its current form, because creating the upper triangular matrix takes $O(n)$ Givens rotations, a result of $\boldsymbol{Q}$ being $n \times n$. This makes its complexity $O(n^2)$, which is too large to run as an inner loop procedure for larger datasets. However, the first p columns of $\boldsymbol{Q}$ and p rows of $\boldsymbol{R}$, denoted as $\boldsymbol{Q}_{[\cdot,1:p]}$ and $\boldsymbol{R}_{[1:p,\cdot]}$, are sufficient to reconstruct $\boldsymbol{Z}$, as $\boldsymbol{Z} = \boldsymbol{Q}_{[\cdot,1:p]} \boldsymbol{R}_{[1:p,\cdot]} = \boldsymbol{Q}\boldsymbol{R}$. Working with this reduced factorization would reduce the storage and number of Givens rotations required for the algorithm.

Unfortunately this representation is insufficient to perform the update. If we were to compute the vector $\boldsymbol{w}$ from Equation 2.12 with $\boldsymbol{Q}_{[\cdot,1:p]}$, then $\boldsymbol{w}' = \boldsymbol{Q}'_{[\cdot,1:p]} \boldsymbol{v}' = \boldsymbol{Q}'_{[\cdot,1:p]} \boldsymbol{e}_i' - \boldsymbol{Q}'_{[\cdot,1:p]} \boldsymbol{H}_i' = \boldsymbol{0}$. To see this, note that $\boldsymbol{Q}'_{[\cdot,1:p]} \boldsymbol{e}_i'$ is zero because the ith row of $\boldsymbol{Z}^t$ and $\boldsymbol{Q}$ is zero, since the ith data point has not been added to the inside region yet. $\boldsymbol{Q}'_{[\cdot,1:p]} \boldsymbol{H}_i'$ is also zero because $\boldsymbol{H}_i$ is perpendicular to $\boldsymbol{Z}^t$ and thus perpendicular to $\boldsymbol{Q}_{[\cdot,1:p]}$. With $\boldsymbol{w}' = \boldsymbol{0}$, Equation 2.12 becomes $\boldsymbol{Z}^{t+1} = \boldsymbol{Q}^t_{[\cdot,1:p]} \boldsymbol{R}^t_{[1:p,\cdot]}$, which completely ignores the update term. Intuitively speaking, we cannot update the column basis of $\boldsymbol{Z}^t$ by only considering that basis.

Fortunately, there is a way to summarize the influence of the last n-p columns of $\boldsymbol{Q}$, denoted $\boldsymbol{Q}_{[\cdot,(p+1):n]}$, into a single vector.

Theorem 2

Define the Givens rotations $(\boldsymbol{J}^1)' \dots (\boldsymbol{J}^{n-1})'$ from equation 2.13 such that $(\boldsymbol{J}^1)' \dots (\boldsymbol{J}^{n-1})'\boldsymbol{w}' = ||\boldsymbol{w}||\boldsymbol{e}_1$, where $\boldsymbol{w}' = \boldsymbol{Q}'\boldsymbol{v}'$. Let $\tilde{\boldsymbol{Q}}_{[\cdot,1:p]}$ be the row concatenation of $\boldsymbol{Q}_{[\cdot,1:p]}$ and the vector $\boldsymbol{q} = \boldsymbol{v}/||\boldsymbol{v}||$. Then

$$(\boldsymbol{J}^1)' \dots (\boldsymbol{J}^p)'\tilde{\boldsymbol{Q}}'_{[\cdot,1:p]}\boldsymbol{v}' = ||\boldsymbol{w}||\boldsymbol{e}_1 \tag{2.20}$$

Proof: Since $\boldsymbol{Q}$ is orthonormal, we can note that $||\boldsymbol{w}|| = ||\boldsymbol{v}||$. Givens rotations are length preserving when applied to vectors. When the Givens rotations zero out element j in $\boldsymbol{w}$, it changes element $j-1$ to $\sqrt{w_{j-1}^2 + w_j^2}$. Consequently, the result of rotations $\boldsymbol{J}'_{p+1} \dots \boldsymbol{J}'_{n-1}$ will set $w_{p+1} = \sqrt{w_{p+1}^2 + \dots + w_n^2} = \sqrt{\sum_{j=p+1}^n (\boldsymbol{Q}'_j\boldsymbol{v})^2)} = ||\boldsymbol{Q}_{[\cdot,(p+1):n]}\boldsymbol{v}||$. Because $\boldsymbol{Q}_{[\cdot,1:p]}$ is perpendicular to $\boldsymbol{v}$, the columns $\boldsymbol{Q}_{[\cdot,p+1]} \dots \boldsymbol{Q}_{[\cdot,n]}$ are an orthonormal basis of $\boldsymbol{v}$. Projecting $\boldsymbol{v}$ onto its own basis will preserve its length, giving us $||\boldsymbol{Q}_{[\cdot,(p+1):n]}\boldsymbol{v}|| = ||\boldsymbol{v}||$. $\boldsymbol{q}\boldsymbol{v}' = \boldsymbol{v}\boldsymbol{v}'/||\boldsymbol{v}|| = ||\boldsymbol{v}||$ as well, thus $\tilde{\boldsymbol{Q}}'_{[\cdot,1:p]}\boldsymbol{v}'$ is equivalent to the first $p+1$ rows of $\boldsymbol{J}'_{p+1} \dots \boldsymbol{J}'_{n-1}\boldsymbol{Q}'\boldsymbol{v}'$. Since the remaining rows of $\boldsymbol{J}'_{p+1} \dots \boldsymbol{J}'_{n-1}\boldsymbol{Q}'\boldsymbol{v}'$ are 0, we have $(\boldsymbol{J}^1)' \dots (\boldsymbol{J}^p)'\tilde{\boldsymbol{Q}}'_{[\cdot,1:p]}\boldsymbol{v}' = ||\boldsymbol{v}||\boldsymbol{e}_1$. $\square$

Theorem 2 allows us to summarize the Givens rotations $\boldsymbol{J}'_{p+1} \dots \boldsymbol{J}'_{n-1}$ with a single vector, meaning we don't need the full n x n forms of $\boldsymbol{Q}$ and $\boldsymbol{R}$. If we append $\boldsymbol{q}$ as a new column of $\boldsymbol{Q}_{[\cdot,1:p]}$ to produce $\tilde{\boldsymbol{Q}}_{[\cdot,1:p]}$ and a zero row to the bottom of $\boldsymbol{R}_{[1:p,\cdot]}$ to produce $\tilde{\boldsymbol{R}}_{[1:p,\cdot]}$ then we can run the algorithm with only $O(p)$ Givens rotations and still produce the same result. Since $\boldsymbol{q}$ is normalized and perpendicular to $\boldsymbol{Q}_{[\cdot,1:p]}$, $\tilde{\boldsymbol{Q}}_{[\cdot,1:p]}$ is still orthonormal.

Algorithm 1 shows the details of this modified QR update procedure. It creates the modified and slimmed down matricies $\tilde{\boldsymbol{Q}}^t_{[\cdot,1:p]}$ and $\tilde{\boldsymbol{R}}^t_{[1:p,\cdot]}$, and uses the techniques of Golub and Loan [2012] to find the factorization after the rank one matrix addition. The first p columns of $\tilde{\boldsymbol{Q}}^{t+1}_{[\cdot,1:p]}$ make our new orthonormal column basis $\boldsymbol{U}^{t+1}$ used to calculate our test statistic T.

Algorithm 2 shows the incremental rank test which calls rankOneUpdate. It takes

Algorithm 1 rankOneUpdate

```
Inputs: $\boldsymbol{Q}, \boldsymbol{R}, \boldsymbol{v}, \boldsymbol{u}$
# Add $\boldsymbol{q}$ as a new column basis to $\boldsymbol{Q}$
$\boldsymbol{q} = \boldsymbol{v}/||\boldsymbol{v}||$
$\boldsymbol{Q} = \text{Append\_Column}(\boldsymbol{Q}_{[\cdot,1:p]}, \boldsymbol{q})$
$\boldsymbol{R} = \text{Append\_Row}(\boldsymbol{R}_{[1:p,\cdot]}, \boldsymbol{0})$
$\boldsymbol{w}' = \boldsymbol{Q}'\boldsymbol{v}'$
# Use givens rotation to zero out $\boldsymbol{w}$
for $i = p - 1$ to 1 do
    $\boldsymbol{G} = \text{givens}(\boldsymbol{w}_i, \boldsymbol{w}_{i+1})$
    $\boldsymbol{Q}_{[\cdot,i:i+1]} = \boldsymbol{Q}_{[\cdot,i:i+1]}\boldsymbol{G}$
    $\boldsymbol{R}_{[i:i+1,\cdot]} = \boldsymbol{G}\boldsymbol{R}_{[i:i+1,\cdot]}$
    $\boldsymbol{w}_{[i:i+1]} = \boldsymbol{G}\boldsymbol{w}_{[i:i+1]}$
end for
$\boldsymbol{R}_{[1,\cdot]} = \boldsymbol{R}_{[1,\cdot]} + w_1\boldsymbol{u}$
# Use Givens rotations to make $\boldsymbol{R}$ upper triangular
for $i = 1$ to $p - 1$ do
    $\boldsymbol{G} = \text{givens}(\boldsymbol{R}_{[i,i]}, \boldsymbol{R}_{[i+1,i]})$
    $\boldsymbol{Q}_{[\cdot,i:i+1]} = \boldsymbol{Q}_{[\cdot,i:i+1]}\boldsymbol{G}$
    $\boldsymbol{R}_{[i:i+1,\cdot]} = \boldsymbol{G}\boldsymbol{R}_{[i:i+1,\cdot]}$
end for
# Return first $p$ columns of $\boldsymbol{Q}$, $p$ rows of $\boldsymbol{R}$
$\boldsymbol{Q} = \boldsymbol{Q}_{[\cdot,1:p]}$
$\boldsymbol{R} = \boldsymbol{R}_{[1:p,\cdot]}$
Return($\boldsymbol{Q}, \boldsymbol{R}$)
```

Algorithm 2 QSSS Incremental Rank Test

```
Inputs: $\boldsymbol{X}, \boldsymbol{H}, \hat{\boldsymbol{b}}, \boldsymbol{Q}, \boldsymbol{R}, \tau, i$
$\boldsymbol{v} = \boldsymbol{e_i} - \boldsymbol{H}_i$
$\boldsymbol{Q}, \boldsymbol{R}$ =rankOneUpdate($\boldsymbol{Q}, \boldsymbol{R}, \boldsymbol{v}, \boldsymbol{X}_i$)
$T = \hat{\boldsymbol{b}}'\boldsymbol{Q}\boldsymbol{Q}'\hat{\boldsymbol{b}}/(\tau(1-\tau))$
Return($T, \boldsymbol{Q}, \boldsymbol{R}$)
```

the index i of the datapoint being added to the region, along with the QR factorization for the previous iteration as inputs.

Note that our incremental rank test is not an approximation as it computes the test statistic (Equation 2.5) exactly.

2.4.1.2 Update Runtime

With our compact representation for $\tilde{\boldsymbol{Q}}_{[\cdot,1:p]}$ and $\tilde{\boldsymbol{R}}_{[1:p,\cdot]}$, the rank one update to our QR factorization takes $O(np)$ time. Each Givens rotation is an $O(n)$ operation, and we perform $O(p)$ of them in total. Once $\boldsymbol{U}^{t+1}$ is found, T^{t+1} can be calculated in $O(np)$ time by computing $\hat{\boldsymbol{b}}\boldsymbol{U}^{t+1} = \boldsymbol{u}$, and then finding $T^{t+1} = \boldsymbol{u}\boldsymbol{u}'$. Thus the entire update to T can be performed in $O(np)$ time when a single point is added to $\tilde{\boldsymbol{X}}$.

2.4.2 One Sided Hypothesis Tests

The rank test for quantile regression defined in Equation 2.5 is a two sided model test. This can be seen from the form of the alternative hypothesis $H_1 : \boldsymbol{\beta}_2 \neq \boldsymbol{0}$. In some applications, it may be beneficial to be able to consider the one sided alternative tests $H_{1+} : \boldsymbol{\beta}_2 \geq \boldsymbol{0}$ and $H_{1-} : \boldsymbol{\beta}_2 \leq \boldsymbol{0}$. These would let us consider alternative models where the response value is strictly increasing or decreasing from the null across all parameters. A general hypothesis test would also be desirable, where each parameter β_i of $\boldsymbol{\beta}_2$ could independently be set as less than zero, greater than zero, or not equal to zero. Having this kind of freedom would allow us to restrict the algorithm to find specific types of model changes in unusual regions.

Silvapulle and Silvapulle [1995] give one way of setting up a restricted alternative hypothesis for the general score test. To illustrate, if we consider the alternative $H_{1+} : \boldsymbol{\beta}_2 \geq \boldsymbol{0}$, then the adjusted score test becomes

$$T_+ = \boldsymbol{S}'\boldsymbol{M}^{-1}\boldsymbol{S} - \inf\{(\boldsymbol{V} - \boldsymbol{\delta})'\boldsymbol{M}(\boldsymbol{V} - \boldsymbol{\delta}) : \boldsymbol{\delta} \geq \boldsymbol{0}\} \tag{2.21}$$

where $\boldsymbol{V} = \boldsymbol{M}^{-1}\boldsymbol{S}$. The added correction term on the right removes the influence of parameters that would decrease in the alternative model, restricting their change to be at least zero. The introduced variable $\boldsymbol{\delta}$ is the same size as $\boldsymbol{\beta}_2$, and the constraints on $\boldsymbol{\delta}$ mimic the restrictions applied to $\boldsymbol{\beta}_2$. If $\boldsymbol{\beta}_2$ is unbounded, such as in our two-sided test, then $\boldsymbol{\delta}$ is unbounded and the correction term reduces to zero, giving us our original hypothesis test.

Performing this correction changes the large sample distribution of T_+ from a chi-squared to a chi-bar-squared distribution. In practice, the specifics of this distribution can be ignored, as significance for the algorithm can be determined using randomization test corrections on the value of T_+.

Solving for the correction term in the one-sided hypothesis test requires finding a solution to a quadratic program with vector constraints. Finding this solution for every candidate region would introduce a major bottleneck in the algorithm runtime. Below, we show how the correction can be calculated efficiently with only a slight increases in the total runtime.

2.4.2.1 Fast One Sided Test Computation

The QP in Equation 2.21 can equivalently be written as $\inf\{\boldsymbol{\gamma}'\boldsymbol{M}\boldsymbol{\gamma} : \boldsymbol{\gamma} \leq \boldsymbol{V}\}$, using $\boldsymbol{\gamma} = \boldsymbol{V} - \boldsymbol{\delta}$ as the vector to solve for. For our rank test, we can further expand this using our form of the information matrix and score vector. Recall from previous sections that $\boldsymbol{S} = \boldsymbol{Z}'\hat{\boldsymbol{b}}$ and $\boldsymbol{M} = \boldsymbol{Z}'\boldsymbol{Z}\Psi$, where $\boldsymbol{Z} = \tilde{\boldsymbol{X}} - \boldsymbol{H}\tilde{\boldsymbol{X}}$, and Ψ is a constant.

$$\boldsymbol{\gamma}'\boldsymbol{M}\boldsymbol{\gamma} = \boldsymbol{\gamma}'(\boldsymbol{Z}'\boldsymbol{Z}\Psi)\boldsymbol{\gamma} \tag{2.22}$$

$$\boldsymbol{V} = \boldsymbol{M}^{-1}\boldsymbol{S} = (\boldsymbol{Z}'\boldsymbol{Z}\Psi)^{-1}\boldsymbol{Z}'\hat{\boldsymbol{b}} \tag{2.23}$$

During our incremental rank test, the factorization $\boldsymbol{QR} = \boldsymbol{Z}$ will be known for the current value of $\boldsymbol{Z}$. Performing this substitution for the previous two equations gives

$$\boldsymbol{\gamma}'(\boldsymbol{Z}'\boldsymbol{Z}\Psi)\boldsymbol{\gamma} = \Psi\boldsymbol{\gamma}'(\boldsymbol{R}'\boldsymbol{Q}'\boldsymbol{Q}\boldsymbol{R})\boldsymbol{\gamma} = \Psi(\boldsymbol{R}\boldsymbol{\gamma})'(\boldsymbol{R}\boldsymbol{\gamma}) = \Psi\sum_{i=1}^{p} r_i^2 \tag{2.24}$$

$$(\boldsymbol{Z}'\boldsymbol{Z}\Psi)^{-1}\boldsymbol{Z}'\hat{\boldsymbol{b}} = (\boldsymbol{R}'\boldsymbol{R}\Psi)^{-1}\boldsymbol{R}'\boldsymbol{Q}'\hat{\boldsymbol{b}} = \Psi^{-1}\boldsymbol{R}^{-1}\boldsymbol{Q}'\hat{\boldsymbol{b}} \tag{2.25}$$

Our quadratic program is equivalent to the least squares optimization of the vector $\boldsymbol{r} = \boldsymbol{R}\boldsymbol{\gamma}$, subject to the constraint $\boldsymbol{\gamma} \leq \Psi^{-1}\boldsymbol{R}^{-1}\boldsymbol{Q}'\hat{\boldsymbol{b}}$. Since $\boldsymbol{R}$ is square upper triangular, its inverse can be found quickly, making this form of the constraints relatively simple to compute. To solve the simplified optimization, we can easily find the gradient and second gradient with respect to $\boldsymbol{\gamma}$.

$$\frac{\partial \Psi(\boldsymbol{R}\boldsymbol{\gamma})'(\boldsymbol{R}\boldsymbol{\gamma})}{\partial \gamma_j} = \frac{\partial}{\partial \gamma_j} \Psi \sum_{i=1}^{p} \left(\sum_{k=i}^{p} R_{i,k}\gamma_k \right)^2 = 2\Psi \sum_{i=1}^{j} r_i R_{i,j} \tag{2.26}$$

$$\frac{\partial^2 \Psi(\boldsymbol{R}\boldsymbol{\gamma})'(\boldsymbol{R}\boldsymbol{\gamma})}{\partial \gamma_j^2} = 2\Psi \sum_{i=1}^{j} R_{i,j}^2 \tag{2.27}$$

The resultant summations go from 1 to j because $\boldsymbol{R}$ is upper triangular, and thus has zeros below the diagonal. We can see from the second gradient that the optimization is convex, making gradient methods guaranteed to converge. With the gradient and second gradient known, our optimization can be solved using a boundary constrained Newton's method with the update

$$\gamma_j^{t+1} = \gamma_j^t - \frac{\sum_{i=1}^{j} r_j R_{i,j}}{\sum_{i=1}^{j} R_{i,j}^2} \tag{2.28}$$

and with the constraint $\boldsymbol{\gamma} \leq \Psi^{-1}\boldsymbol{R}^{-1}\boldsymbol{Q}'\hat{\boldsymbol{b}}$. This derivation is for the alternative hypothesis $\boldsymbol{\beta}_2 \geq 0$. For $\boldsymbol{\beta}_2 \leq 0$, we simply flip the signs on the constraint and solve with $\boldsymbol{\gamma} \geq \Psi^{-1}\boldsymbol{R}^{-1}\boldsymbol{Q}'\hat{\boldsymbol{b}}$. In general, we can decide the direction of inequality for each term in $\boldsymbol{\beta}_2$ and adjust the constraint on γ_j accordingly. If we want the jth term of $\boldsymbol{\beta}_2$ to have a two sided alternative, then we simply leave γ_j unconstrained.

2.4.2.2 One-Sided Test Runtime Analysis

In order to change the QSSS algorithm to account for a one-sided hypothesis test, the minimizing value of $\boldsymbol{\gamma}$ must be calculated for each candidate region in the scan algorithm. Since our regions grow point by point from an initial circle, we can greatly speed up the convergence of the optimization by warmstarting it with the value of $\boldsymbol{\gamma}$ from the previous region. Each iteration of Newton's method takes $O(p^2)$ time to compute.

For each new region, we must also compute the constraint boundary $\Psi^{-1}\boldsymbol{R}^{-1}\boldsymbol{Q}'\hat{\boldsymbol{b}}$. $\boldsymbol{R}$ is upper triangular, and its inverse can be found through back-substitution in $O(p^2)$ time. The full product then takes $O(np)$ time to compute. Computing the adjusted test statistic $T - \Psi(\boldsymbol{R}\boldsymbol{\gamma})^{-1}(\boldsymbol{R}\boldsymbol{\gamma})$ can also be done in $O(p^2)$ time.

If we let k be the number of iterations required for Newton's method to converge, the total runtime overhead of the one sided test for each region is $O(np + kp^2)$. As long as warmstarting keeps the value of k relatively low, this will not change the $O(np)$ asymptotic runtime of the QSSS update.

It should also be noted that in some cases the one-sided test statistic will be the same as the two-sided alternative. For example, for the alternative test of $\boldsymbol{\beta}_2 \geq 0$, if it occurs that $\boldsymbol{V} \geq \mathbf{0}$, then $\boldsymbol{V}$ is a valid assignment for $\boldsymbol{\delta}$. $\boldsymbol{\delta} = \boldsymbol{V}$ causes the correction term to drop to zero (and is thus the minimizing assignment to $\boldsymbol{\delta}$), making the one-sided test value equal to the two-sided value. Thus, if $\boldsymbol{V} \geq \mathbf{0}$ then the one-sided corrections do not need to be applied. By comparing $\boldsymbol{V}$ against $\mathbf{0}$ at each iteration using the one-sided test of interest, we can prune unnecessary corrections to the test statistic and further improve the runtime.

2.4.3 Multiple Hypothesis Test Correction

To account for the multiple hypothesis test problem, we perform a correction using the method in Abrams et al. [2010]. We generate 1000 simulations of the data under the null hypothesis. The maximum test statistic from each of these simulations are used to fit the parameters μ, γ of a Gumbel distribution. We calculate the adjusted p-value of a region with test statistic T as $1 - g(T|\mu, \gamma)$, where g is the CDF of the Gumbel distribution. This tells us the rarity of drawing a value at least as large as T from the distribution of maximum test statistics. In all of our applications we report the most significant region found by QSSS, provided that the adjusted p-value of the region is less than 0.05. Otherwise no significantly different region is found.

2.5 Results

2.5.1 Synthetic Data Experiments

We begin by demonstrating the speedup from our incremental formulation of the Rank test and its one-sided variant, along with a comparison of the Rank test to other possible choices of hypothesis tests. We use synthetic data to evaluate these two criteria, as it is easy to generate in large quantities, and it can contain a verifiable ground truth[2].

2.5.1.1 Simulator

The purpose of our simulator is to inject data points in spatial regions where the data distribution is altered at a specific percentile. We start with a default distribution, then modify a specific range of the distribution for a random spatial region. This acts as the target region for the algorithm to identify.

In our data simulator, values for $\boldsymbol{X}$ and $\boldsymbol{L}$ are generated uniformly at random across defined minimum and maximum values. The response values $\boldsymbol{Y}$ are then calculated as

$$y_i = \boldsymbol{x}_i \boldsymbol{\beta} + \epsilon \tag{2.29}$$

where $\epsilon \sim Norm(0, \sigma)$ is normally distributed noise term. By uniformly generating quantile values $\boldsymbol{q} \in [0, 1]^n$, we can specify the value of the error term for data point i as $\epsilon(q_i)$.

For each dataset generated, the simulator selects a random circular subset of the data $\boldsymbol{C}$, where $\boldsymbol{C}$ contains a specified number of points. This area is the 'target' region for the algorithms to locate. A random offset parameter $\boldsymbol{\delta}$ is generated, with $|\boldsymbol{\delta}|$ kept constant for each experiment. Exactly three values in $\boldsymbol{\delta}$ are non-zero, which are randomly selected when $p > 3$. For every point $i \in \boldsymbol{C}$, if $|q_i - \tau| < 0.1$ then y_i is calculated from the following formula instead:

$$y_i = \boldsymbol{x}_i(\boldsymbol{\beta} + \boldsymbol{\delta}) + \epsilon(q_i) \tag{2.30}$$

This modification effectively changes the parameterization of the model inside $\boldsymbol{C}$ within the quantile range $[\tau - 0.1, \tau + 0.1]$. Specifically, three of the p covariates are changed, corresponding to the non-zero entries of $\boldsymbol{\delta}$. We evaluate each algorithm based on the proportion of data points in the best reported region that are in $\boldsymbol{C}$.

[2]Matlab code for our experiments and algorithms can be found at https://github.com/moortrav/QSSS

2.5.1.2 Comparison against Other Baselines

TESS is the only other recent method we are aware of that can be easily modified to solve exactly the same problem as the QSSS. We compare against three additional baseline algorithms, named Mean, SSS-Moods, and LR. Mean fits standard least squares regressions to the inside and outside region, and compares them using the likelihood ratio test. This algorithm compares the means, not quantiles, of the distributions. SSS-Moods fits a τth quantile regression with coefficients $\boldsymbol{\beta}$ to the entire data set. For each test region, this baseline calculates Mood's test at τ, which compares the number of points above and below the $\boldsymbol{\beta}$ plane from both inside and outside of the region. LR uses the more powerful but computationally expensive likelihood ratio test from Koenker and Machado [1999] for quantile regression. This LR test forms a Chi-squared statistic from the residuals of the quantile regressions fit to the null and alternative models. Each of these baselines uses the same region search as our QSSS.

To modify TESS to be applicable to our setup, we replace the subset scan with the same search over spatial regions used in the other algorithms. We compute the baseline p-values by fitting a non-parametric distribution to the entire data set. Specifically, we first fit a quantile regression to the data at the τth quantile and then calculate the residuals for each point. Each point is then assigned a p-value based on the percentage of points with a lower residual. For example, if the jth residual is larger than 90% of the other points, then point j is given a p-value of 0.9. This approach is a straightforward and meaningful way of converting the multivariate data into a univariate distribution, which the TESS algorithm was originally designed for. We found that using these non-parametric p-values produce better results than attempting to fit a parametric model, such as the normal distribution.

Using our simulator, we produced 30 randomly generated data sets for each experiment setting. We keep $n = 2000$ and look at $p = 5$, 10, 15, and 20. 500 of the points in each dataset form a circular target region. All the points in this region between the $(\tau - 0.1)$ and $(\tau + 0.1)$ quantiles are generated from the linear model $\boldsymbol{Y} = \boldsymbol{X}(\boldsymbol{\beta} + \boldsymbol{\delta})$, while the rest of the data is generated from the model $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta}$. We investigate setting τ equal to 0.1, 0.3, 0.5, 0.7, and 0.9. Our SSS-Moods, LR, Rank, and TESS tests search for regions that differ at the τth percentile, while the mean test only looks for differences in the mean.

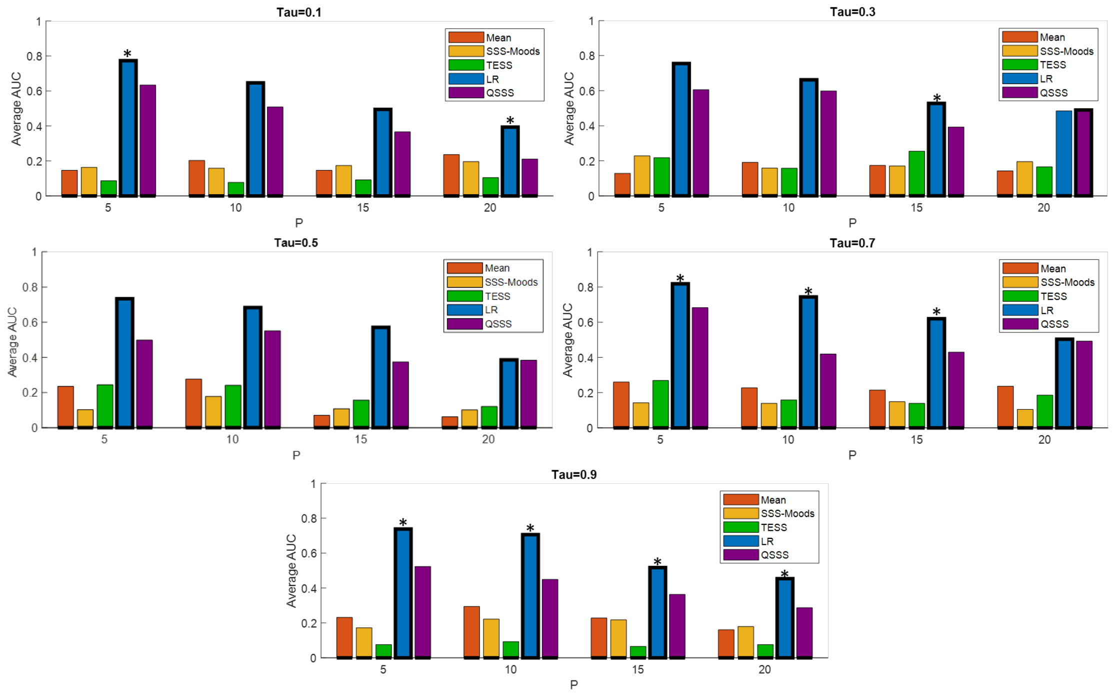


Figure 2.1: Average AUC of each algorithm for each experiment setup. Bold denotes the best performing algorithm, while * indicates algorithms that are significantly better than all others (Wilcoxon signed-rank test, $\alpha = 0.05$).

For each algorithm, we look at the accuracy of the most significant region found for each dataset. We compute the AUC of each algorithm, using points in the target regions as true positives. Specifically, we compute the true positive rate ($tpr = \frac{\text{True Positives Detected}}{\text{Total True Positives}}$) and the false positive rate ($fpr = \frac{\text{False Positives Detected}}{\text{Total True Negatives}}$) for each dataset. This gives a single point in the ROC curve, where the curve is equal to 0 to the left of fpr, and equal to tpr to the right of fpr. We compute the area under this curve (AUC) as $tpr(1 - fpr)$, and then report the average AUC of each algorithm over the 30 datasets generated for each experiment setup.

Figure 2.1 shows the average AUC of each algorithm across all experiments. The likelihood ratio algorithm is a clear winner in terms of performance, with QSSS a definitive second place. We expected these two algorithms to be the best, since they have the best statistical power for our simulated data of the five variants. Our Mean, SSS-Moods, and TESS algorithms all performed equally poorly in the experiments, as each has its own issue in this setting. A mean-based test is a poor choice for detecting quantile specific changes, and the Moods test lacks power. TESS prefers a separate control set to learn a distribution model, which cannot be obtained in a setting where the anomalous regions are unknown. Using the entire dataset incorporates these anomalous regions into the model, making them harder to identify.

2.5.1.3 Timing Results

Using our simulated data, we compare the runtime of our incremental version of the Rank test to its naive (non-incremental) formulation. For each algorithm we calculated the Rank test statistic T, starting from a base radius, then expanding to include 100 new points. In Figure 2.2 (left) we show the average time, in milliseconds, that each algorithm took to calculate T when a new point was added for increasing values of n. Our experiments show that while the naive algorithm is quadratic in n, our incremental version is linear.

Figure 2.2 also shows the average total runtime of our incremental QSSS versus each baseline algorithm as p increases with n fixed at 2000, and as n increases with p fixed at 10. Because they reduce the multivariate data down to univariate values and perform simple hypothesis tests, TESS and SSS-Moods are extremely fast, though at a huge cost to accuracy. Likelihood ratio is easily the slowest of the algorithms, and quickly becomes intractable to use as n increases to even moderate size. With the incremental speedups, our QSSS algorithm has a comparable runtime to the Mean scan algorithm, scaling similarly as p and n increase. This makes our algorithm far more usable on larger datasets than the likelihood ratio test, with only a small trade-off in performance.

2.5.1.4 One Sided Test Evaluation

To compare the power of our one-sided QSSS, we use the same data simulator as the previous experiments. We set n to 2000, p to 5, and test over a range of τ values. In these experiments, the parameter offset $\boldsymbol{\delta}$ is set so that $\boldsymbol{\delta} \geq \mathbf{0}$. We compare the performance of QSSS using the default two-sided test, to QSSS using the one-sided alternative $\boldsymbol{\beta_2} \geq \mathbf{0}$.

Figure 2.3 shows the average AUC of the two algorithms. The one-sided variant is strictly better than the two-sided QSSS in every experiment, and significantly so in three

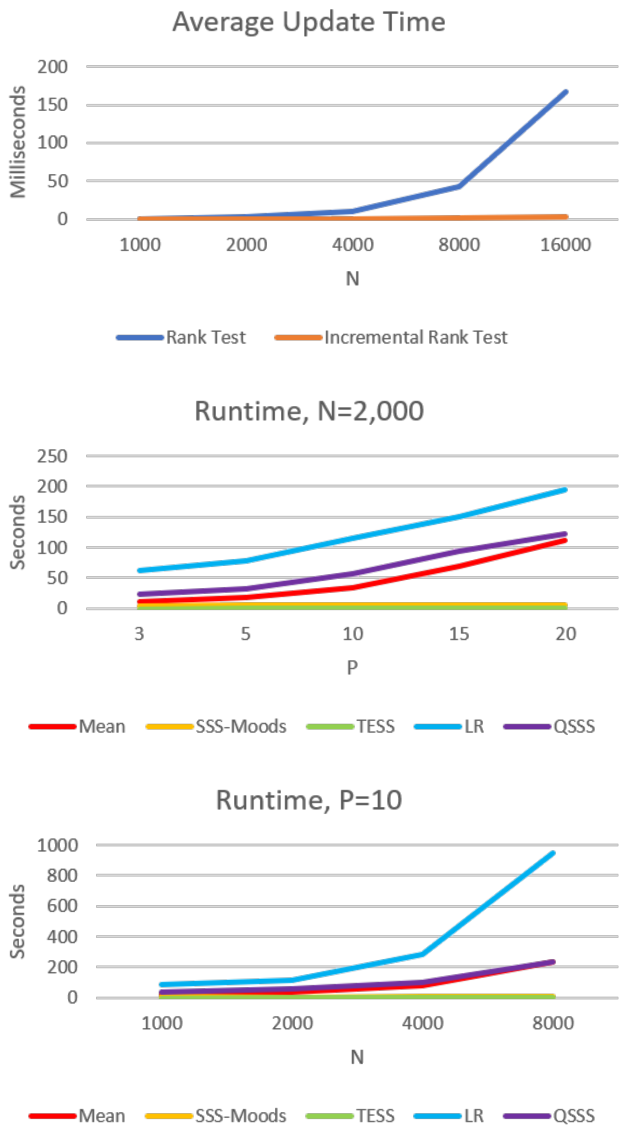


Figure 2.2: (Left) Average time to update the Rank test statistic, for the naive and incremental formulations. (Middle, Right) Average runtime for each algorithm for increasing P and N.

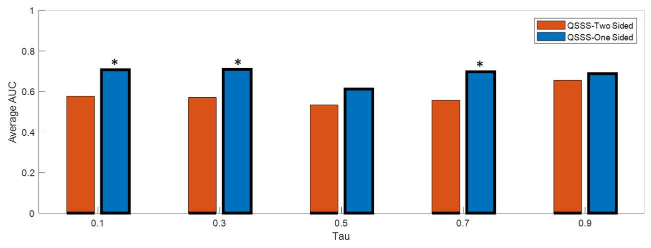


Figure 2.3: Average AUC of one-sided QSSS vs two-sided QSSS. Bold denotes the best performing algorithm, while * indicates algorithms that are significantly better (Wilcoxon signed-rank test, $\alpha = 0.05$).

of the five setups.

2.5.1.5 One-Sided Test Runtime Results

We use our data simulator to evaluate the difference in runtime between the QSSS algorithm using a two sided rank test vs a one sided rank test. Figure 2.4 shows the runtime of each algorithm on datasets of increasing size n and with increasing dimensionality p. Each runtime is averaged over 30 randomly generated datasets. Algorithmically, we have observed that the one sided test should take strictly more time than the two sided test, but is unlikely to be of a different complexity order. Our results indicate that the additional overhead of the one sided test is negligible compared to the total runtime, making it nearly identical to the two-sided test in runtime.

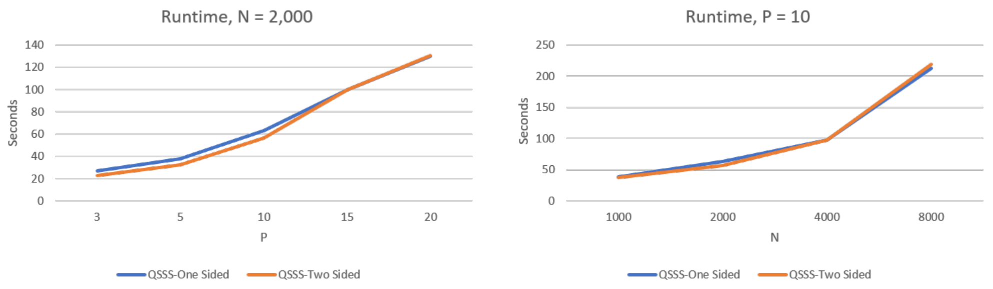
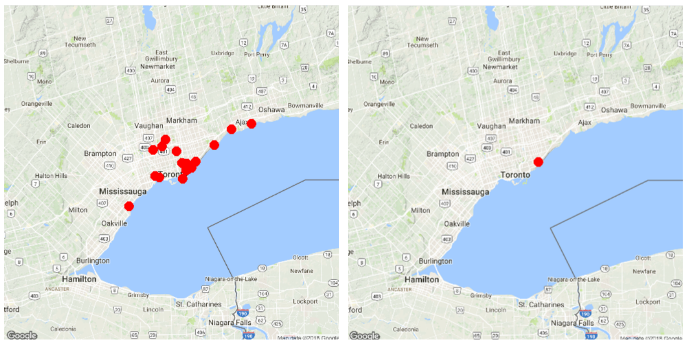


Figure 2.4: Average runtime for QSSS algorithm using a one sided test and two sided test, for increasing values of n, and increasing values of p.

2.5.2 Robustness to Outliers

One of the benefits of quantile based analysis is that it is more robust to data outliers than mean-based methods. We illustrate how this can affect spatial scan analysis using eButterfly data as an example use case.



(a) QSSS 50th Percentile (b) Mean

Figure 2.5: Most significant regions found from eButterfly data, with data points in the region shown as red dots. The figures are zoomed in on the Toronto region for visibility.

Citizen science biodiversity monitoring programs, such as eButterfly [Prudic et al., 2017], play an important role in ecology by providing data for species distribution models and also conservation programs. Participants in these programs submit checklists which record observations of certain types of organisms, such as butterflies in the case of eButterfly, identified by species.

We construct a dataset out of the abundance counts of monarch butterflies (i.e. the number of butterfly individuals observed) in Ontario in 2016. Quantile regression on count data can be addressed using the smoothing method in Machado and Silva [2002], which turns the counts into continuous values by adding uniform noise. This transformation allows us to perform inference with the Rank test as we would with continuous data.

In our analysis, we include the time spent observing for each checklist as the covariate, since there should be a strong correlation between this value and the number of monarchs observed.

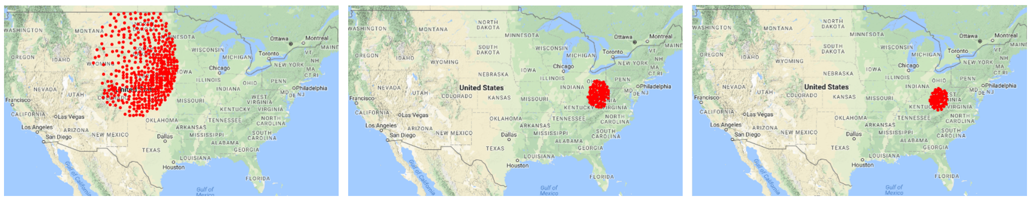
Figure 2.5 shows the most significant regions found by the mean regression spatial scan and QSSS at the 50th percentile[3]. We ran TESS on the dataset at the 50th percentile as well, but it failed to find any significant region. Inspection of the data, and verification with domain experts at eButterfly reveal an important benefit to using QSSS. The region found by the mean spatial scan only detects checklists from a single observer, whose extremely high counts are an outlier in the dataset. In contrast, our QSSS was less affected by the outliers, and found an area of high monarch counts due to migration routes around the great lakes.

If we were limited to only mean-based spatial scans, we would have to filter out the outlier data to find the desired trends in the dataset. Being able to adjust the percentile of QSSS allows us to reduce the influence of outliers as desired, without explicit removal of outliers from the data.

2.5.3 Quantile Based Region Detection

We now demonstrate the usefulness of detecting unusual spatial regions based on different quantiles.

2.5.3.1 Education and Unemployment Data



(a) 10th Percentile (b) 90th Percentile (c) Mean

Figure 2.6: Most significant region found by the QSSS algorithm for the 10th and 90th percentiles of the Education and Unemployment dataset. Most significant region by the mean spatial scan is included for comparison. Regions are illustrated by the centroids of the counties they contain.

We combine the county-level education and unemployment datasets from the USDA Economic Research Service web page [Parker, 2017]. We use the county-level unemploy-

[3]All maps generated using ggmap in R [Kahle and Wickham, 2013]

ment rates from 2016 as the response variable, and combine the education percentages from 2012-2016 with median household income (as percentage of state total) values from 2016 as the covariates. The education percentages are the proportion of adults in each county with less than a high school diploma, just a high school diploma, one to three years of college, and four years of college or more. We only use the counties from the continental US.

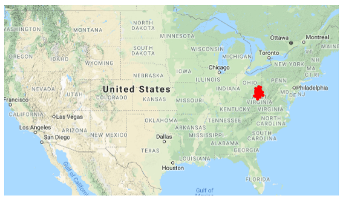
We ran TESS and our QSSS algorithm on the 10th and 90th percentile of the data, along with a mean-based approach using least squares regression. Figure 2.6 shows the most significant region found by QSSS and the mean scan. TESS was unable to find any significant regions at the 10th or 90th percentiles. Both the mean and 90th percentile QSSS search found the Appalachian region that intersects Kentucky, West Virginia and Virginia, which is well-known to have high unemployment rates with the collapse of the coal industry [Caruthers, 2016]. In the 10th percentile QSSS region, South Dakota, North Dakota, Nebraska, and Colorado are rated 2,3,4, and 6 in unemployment in the continental US as a whole. This middle region of the country enjoys lower unemployment rates due to the local oil industry and relatively low fallout from the Great Recession [DePillis, 2018]. The most significant region discovered at the 10th percentile has a 2 point lower unemployment rate on average, which is abnormally low even when compared to other low unemployment areas.

The unemployment data results highlight the fact that the QSSS, unlike the mean scan, can identify multiple trends in a dataset by changing the modeled quantile.

2.5.3.2 One Sided Unemployment Tests

For our one sided experiments, we modify the unemployment data to include more unique and diverse covariates. We keep the percent of county populations without high school diplomas and the percent with 4 or more years of college from the education covariates, as well as the median household income. We also add the labor force size and urban/rural continuum code for each county. The continuum code is a discrete numerical value ranging from 1 (very urban) to 9 (very rural). As before, we use these covariates to predict unemployment rate at a county level.

Figure 2.7 shows the most significant region from two different one sided tests performed on the 90th percentile. The first test specifies that all the model parameters should be increasing in the region, necessitating that the 90th percentile unemployment rate is abnormally high for that area. The second test specifies all model parameters as decreasing in the region. This will find an area with significantly low unemployment at the 90th percentile.


(a) Increasing

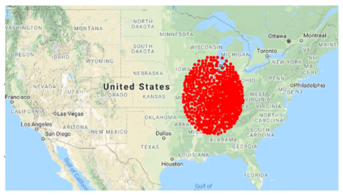
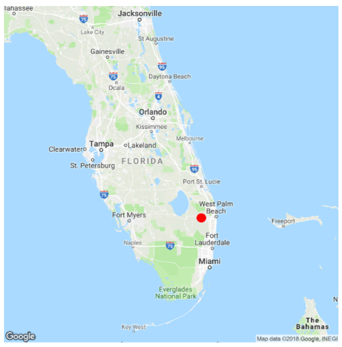

(b) Decreasing

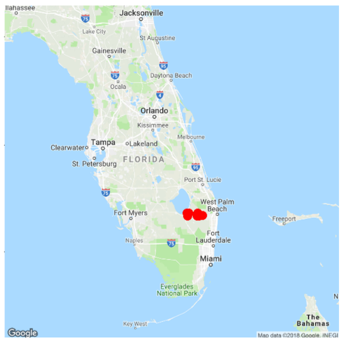
Figure 2.7: Significant regions for 3 different one sided tests performed on the unemployment dataset.

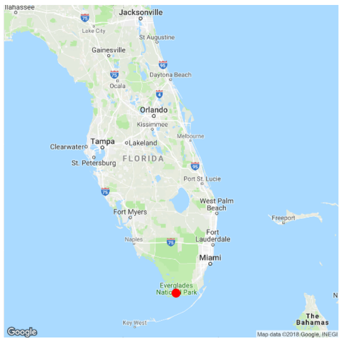
The result of the first one sided test is unsurprising, as it found a subset of the Appalachian region from the two sided test. We had surmised from our previous results that this area was significant due to unusually high unemployment, and the one sided test corroborates this conclusion. In the second test we only considering decreasing parameters. The region shown was the only significant one found, with significant decreases in the parameters for less than high school diploma and median household income. This region contains an area of the country with many manufacturing and blue-collar jobs. These jobs tend to have lower education requirements and lower pay, making this region unique in that low education and income decrease the unemployment rate at the 90th percentile.

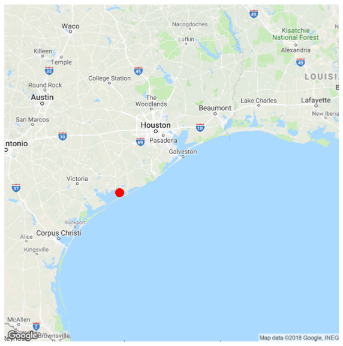
2.5.3.3 eBird

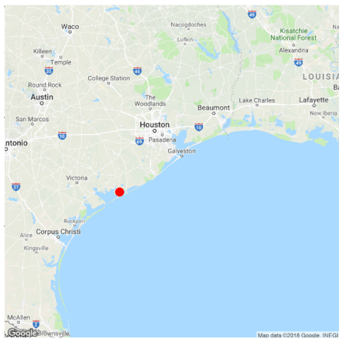
The next case study presents the results of applying QSSS to eBird [Sullivan et al., 2014] data. The eBird project collects bird observation checklists from citizen scientists around the world. We compiled two datasets from eBird data collected in 2017 between March and April. These datasets correspond to two different Bird Conservation Regions (BCRs) within the U.S. We divide the data by BCR because they represent cohesive habitats for different bird species. Our choice of March and April is to mitigate the effects of seasonality on the algorithms.


(a) TESS 90th Percentile


(b) QSSS 90th Percentile


(c) Mean


(d) TESS 90th Percentile


(e) QSSS 90th Percentile

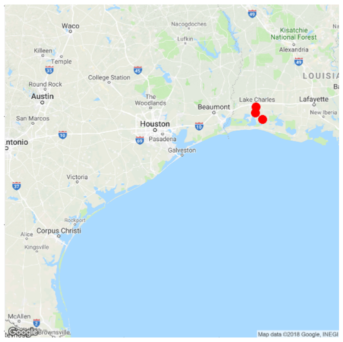

(f) Mean

Figure 2.8: The most significant regions found by TESS and QSSS at the 90th percentile, and by the mean spatial scan on eBird data from BCRs 31 (Florida) and 37 (Gulf coast). Data points within a region are shown as red dots.

Different from our eButterfly study, we used the total number of species observed from each checklist as our response variable, and the time spent observing as the covariate. Past work has shown that the number of species observed per unit time is highly predictive of the skill level of an observer [Kelling et al., 2015]. We use the same count smoothing approach on the eBird data as we did on eButterfly to fit the quantile regression model.

Figure 2.8 shows the most significant regions found for the mean spatial scan compared to TESS and QSSS run at the 90th percentile. We corresponded with domain experts from eBird, who offered an analysis on the regions detected.

For BCR 31, TESS and QSSS found an unusual birding location – one that is less frequented by beginners. The birders who visit this region are highly skilled and are able to continue observing a high number of bird species as they stay there. In contrast, the mean scan found a popular species-rich hotspot in the Everglades frequented by both experts and novices. This region has many large wading birds which are easy to see and identify initially.

In BCR 37, TESS and QSSS again find the same region. This area in Matagorda Bay is a hotspot, because it has an unusually high number of bird species along the shoreline that can be readily observed as compared to the surrounding area. Our domain expert commented that the area found by the mean scan was an area that was not particularly high in species. Upon inspecting the models for the inside versus outside region, we found that the models indicate that observers appear to find fewer species initially inside that area than outside that area.

The mean scan and quantile based algorithms found very different but meaningful regions for the BCRs. We hypothesize that TESS and QSSS are finding unusual areas for more skilled observers (as in BCR 31), specifically areas where they are able to identify more birds over time.

We also ran our one-sided QSSS algorithm on these datasets. When looking for areas with strictly increasing parameters at the 90th percentile, our one-sided test found the same areas in BCRs 31 and 37 as our two sided test. This further verifies our hypothesis that these areas are significant due to a higher number of birds being spotted by expert observers.

2.6 Conclusion

The QSSS discovers unusual spatial regions that differ from the surrounding area. The inner loop of the algorithm relies on comparing quantile regressions fit to data from inside and outside a region under consideration. To perform these comparisons efficiently, we developed an incremental rank test, which is over an order of magnitude faster than a naive implementation. Our results on simulated data and on three real-world datasets show that QSSS enables a new type of analysis for spatial data that is different from mean-based methods and that the QSSS is also robust to outliers.

Chapter 3: Quantile Snapshot Scan

3.1 Introduction

The QSSS presented in the previous chapter only finds unique regions over space. There are many potential applications where a similar kind of quantile region detection could include a time aspect as well. For example, suppose an analyst is comparing counts of a specific species between the current year and the previous year. Each data point represents a geographic location with a response (e.g., the number of individuals observed) and an associated vector of covariates (e.g., features related to the observation process such as the time of day, time spent observing, etc.). A common task in this analysis is to look for the spatial region that is the most different between the two snapshots in time. In addition, the analyst may be interested in regions that differ according to a specific quantile of the response value. For example, the analyst may be interested in areas of high density for the species, such as those in the 75th percentile, and thus focus on how these regions have changed between the two snapshots.

This type of spatial analysis is applicable to many other spatial datasets, such as data from crop yields, property tax assessments and unemployment surveys. To solve problems of this nature, we introduce a novel algorithm, called the Quantile Snapshot Scan (Qsnap for short), that extends the previous QSSS algorithm for use on different snapshots in time[1]. Qsnap finds the spatial region that differs most between two time periods, where the difference is measured relative to a specific quantile of the response variable. More precisely, Qsnap looks for differences relative to the two models predicting the conditional quantile function for the two snapshots. This modification violates the assumptions used in the QSSS fast update algorithm and requires a new method to be developed for the snapshot setting.

We show that Qsnap is more robust at detecting quantile differences for a variety of distributions than competing approaches, and we develop a new efficient incremental update that speeds up a naive implementation of the algorithm by an order of magnitude.

[1]An earlier version of this work was published in AISTATS [Moore and Wong, 2020]

We also apply Qsnap to the tasks of identifying bird migration routes and detecting changes in drought conditions.

3.2 Related Work

Qsnap is based on many of the same building blocks as QSSS, including quantile regression [Koenker and Bassett, 1978], the Spatial Scan Statistic [Kulldorff, 1997], and the rank test for quantile regression [Gutenbrunner et al., 1993]. Details on these related works can be found in section 2.2 in the previous chapter.

The computer vision and remote sensing communities address a seemingly related problem of change detection for images and landsat data taken at different times (see Radke et al. [2005], Zhu [2017] for surveys). This line of research is different from our work, as those techniques are specifically intended for images and regularly gridded landsat values, rather than randomly located spatial data. In addition, these methods do not compare quantiles of the data distributions.

3.3 Methodology

Suppose we are given two spatial datasets obtained at two different times (i.e., snapshots). We denote the two datasets as $\boldsymbol{D}^{(1)} = \{\boldsymbol{Y}^{(1)}, \boldsymbol{X}^{(1)}, \boldsymbol{L}^{(1)}\}$ and $\boldsymbol{D}^{(2)} = \{\boldsymbol{Y}^{(2)}, \boldsymbol{X}^{(2)}, \boldsymbol{L}^{(2)}\}$. For the dataset at snapshot s, we denote the ith data point as $\boldsymbol{D}_i^{(s)} = (y_i^{(s)}, \boldsymbol{x}_i^{(s)}, \boldsymbol{l}_i^{(s)})$ where $y_i^{(s)}$ is the continuous response, $\boldsymbol{x}_i^{(s)} = (x_{i,1}^{(s)}, \ldots, x_{i,p}^{(s)})$ are the p covariates associated with the ith data point and $\boldsymbol{l}_i^{(s)} = (l_{i,1}^{(s)}, \ldots, l_{i,d}^{(s)})$ are the d dimensional coordinates specifying the spatial location of the data point. If $d = 2$, the location tuple $(l_{i,1}^{(s)}, l_{i,2}^{(s)})$ can represent latitude and longitude. Note that the set of locations $\boldsymbol{L}^{(1)}$ and $\boldsymbol{L}^{(2)}$ are not required to be from the exact same locations for the two snapshots, but they should be from the same general region. Our goal is to find a region $\boldsymbol{C}$ that is the most different between the two snapshots (with respect to the τth quantile of the response variable) and to compute a score that characterizes this region's unusualness.

The Qsnap algorithm searches over candidate regions $\boldsymbol{C}$. As is commonly done in SSS variants, the search involves looking at regions of expanding size (e.g., circles of increasing radius), centered at evenly-spaced gridpoints covering the space $\boldsymbol{L} = \boldsymbol{L}^{(1)} \cup \boldsymbol{L}^{(2)}$. For each candidate region, Qsnap performs a hypothesis test to determine when the τth quantile of $\boldsymbol{D}^{(1)}$ is different from the τth quantile of $\boldsymbol{D}^{(2)}$ in region $\boldsymbol{C}$. For this paper, we will focus on optimizing the speed and power of the hypothesis test.

3.3.1 Snapshot Hypothesis Test

Given a region $\boldsymbol{C}$, we denote the response variables and associated covariates of the datapoints from snapshot s in region $\boldsymbol{C}$ as $\boldsymbol{Y}_{\boldsymbol{C}}^{(s)}$ and $\boldsymbol{X}_{\boldsymbol{C}}^{(s)}$ respectively. We want to compare $Q_{\boldsymbol{Y}_{\boldsymbol{C}}^{(1)}}(\tau|\boldsymbol{X}_{\boldsymbol{C}}^{(1)})$ and $Q_{\boldsymbol{Y}_{\boldsymbol{C}}^{(2)}}(\tau|\boldsymbol{X}_{\boldsymbol{C}}^{(2)})$, the τth quantile regression models of $\boldsymbol{C}$ at snapshots 1 and 2. We define $\boldsymbol{X} = [\boldsymbol{X}_{\boldsymbol{C}}^{(1)}; \boldsymbol{X}_{\boldsymbol{C}}^{(2)}]$ and $\boldsymbol{Y} = [\boldsymbol{Y}_{\boldsymbol{C}}^{(1)}; \boldsymbol{Y}_{\boldsymbol{C}}^{(2)}]$, where the symbol ";" indicates matrix concatenation (as in Matlab). We construct the matrix $\tilde{\boldsymbol{X}}$ such that $\tilde{\boldsymbol{X}}_i = \boldsymbol{X}_i$ if point i is from snapshot 2, and $\boldsymbol{0}$ (i.e., the zero vector) otherwise. We can then set up the quantile regression model $Q_{\boldsymbol{Y}}(\tau|\boldsymbol{X}, \tilde{\boldsymbol{X}}) = \boldsymbol{X}\boldsymbol{\beta}_{\boldsymbol{1}} + \tilde{\boldsymbol{X}}\boldsymbol{\beta}_{\boldsymbol{2}}$. This represents a nested model where $\boldsymbol{\beta}_{\boldsymbol{1}}$ are the parameters for snapshot 1 and $(\boldsymbol{\beta}_{\boldsymbol{1}} + \boldsymbol{\beta}_{\boldsymbol{2}})$ are the parameters for snapshot 2. Testing the null hypothesis $\boldsymbol{\beta}_{\boldsymbol{2}} = \boldsymbol{0}$ vs the alternative $\boldsymbol{\beta}_{\boldsymbol{2}} \neq \boldsymbol{0}$ will determine if the region $\boldsymbol{C}$ in snapshot 2 is significantly different from the region in snapshot 1 at the τth quantile.

We choose to use the rank test for quantile regression, because it only requires fitting a quantile regression model for $\boldsymbol{\beta}_{\boldsymbol{1}}$ under the null hypothesis, and it has the same asymptotic power as a likelihood ratio test. The test statistic T can be formulated as $T = \hat{\boldsymbol{b}}'\boldsymbol{Z}(\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{Z}'\hat{\boldsymbol{b}}/\Psi^2$ where $\boldsymbol{Z} = \tilde{\boldsymbol{X}} - \boldsymbol{H}\tilde{\boldsymbol{X}}$.

Alternatively, we can write this as $T = \hat{\boldsymbol{b}}'\boldsymbol{Q}_{\boldsymbol{Z}}\boldsymbol{Q}_{\boldsymbol{Z}}'\hat{\boldsymbol{b}}/\Psi^2$ if $\boldsymbol{Q}_{\boldsymbol{Z}}$ is an orthonormal basis for $\boldsymbol{Z}$. In either form, re-calculating T from scratch each time $\boldsymbol{C}$ changes can be very time consuming. The incremental update used in QSSS employed the same testing framework, but with the assumptions that $\boldsymbol{X}$, $\boldsymbol{H}$, $\hat{\boldsymbol{b}}$, and $\boldsymbol{\beta}_{\boldsymbol{1}}$ are all constant between iterations. These assumptions are violated in the snapshot scan setting. Specifically, when $\boldsymbol{C}$ grows by one point, both $\tilde{\boldsymbol{X}}$ and $\boldsymbol{X}$ grow by an additional row, which requires $\boldsymbol{H}$, $\hat{\boldsymbol{b}}$, and $\boldsymbol{\beta}_{\boldsymbol{1}}$ to be recalculated. In the following section we will outline a novel speedup for the snapshot scan setting.

3.3.2 An Efficient Incremental Update

We now derive a novel incremental update for Qsnap that drastically reduces the time needed to calculate T when $\boldsymbol{C}$ grows. In each iteration of Qsnap, the size of the region $\boldsymbol{C}$ grows by one point, which increases the size of $\boldsymbol{X}$ and $\tilde{\boldsymbol{X}}$ by one row. Changes to $\boldsymbol{X}$ and $\tilde{\boldsymbol{X}}$ means $\boldsymbol{Q_Z}$, $\boldsymbol{Z}$, $\boldsymbol{H}$, and $\hat{\boldsymbol{b}}$ must be updated to re-calculate T. While T only depends on $\boldsymbol{Q_Z}$ and $\hat{\boldsymbol{b}}$, $\boldsymbol{Q_Z}$ will change in relation to $\boldsymbol{Z}$, since it is a basis for $\boldsymbol{Z}$, and $\boldsymbol{Z}$ depends on $\boldsymbol{H}$. In the following sections, we describe efficient incremental updates for these values. For many values, we save time by updating the QR decomposition of the data matrix, instead of the matrix itself. We use the superscript t to indicate the current iteration, and $t+1$ the iteration after adding a new data point. The proofs of the theorems used to derive these updates are in section 3.3.3.

Updating $\boldsymbol{H}$: We start by addressing how to quickly update $\boldsymbol{H}^t$ as $\boldsymbol{X}^t$ grows in size. First, let $\boldsymbol{X}^t = \boldsymbol{Q}^t_{\boldsymbol{X}}\boldsymbol{R}^t_{\boldsymbol{X}}$ be the QR factorization of $\boldsymbol{X}^t$. Note that $\boldsymbol{H}^t = \boldsymbol{X}^t(\boldsymbol{X}^{t\prime}\boldsymbol{X}^t)^{-1}\boldsymbol{X}^{t\prime}$ is a projection matrix, and can be re-written as $\boldsymbol{H}^t = \boldsymbol{Q}^t_{\boldsymbol{X}}\boldsymbol{Q}^{t\prime}_{\boldsymbol{X}}$ since $\boldsymbol{Q}^t_{\boldsymbol{X}}$ is an orthonormal basis for $\boldsymbol{X}^t$. An initial $\boldsymbol{Q_X}$ comes from the Q matrix of the QR factorization of $\boldsymbol{X}^t$ at $t=1$.

Theorem 3

Let $\boldsymbol{x}^{t+1}$ be the new row added to $\boldsymbol{X}^t$ at iteration $t+1$. If the QR factorization $\boldsymbol{Q}^t_{\boldsymbol{X}}\boldsymbol{R}^t_{\boldsymbol{X}} = \boldsymbol{X}^t$ is known, then

$$\boldsymbol{H}^{t+1} = \begin{bmatrix} 1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{H}^t \end{bmatrix} - \boldsymbol{v}\boldsymbol{v}' \tag{3.1}$$

where $\boldsymbol{v}$ is the last column of $\boldsymbol{Q}^{t+1}_{\boldsymbol{X}}$.

$\boldsymbol{Q}^{t+1}_{\boldsymbol{X}}$ can be efficiently found using the modified Golub and Loan [2012] update shown in Algorithm 2 of the previous chapter. This uses $O(p)$ Givens rotation to update the existing QR factorization.

Updating $\boldsymbol{Z}$: Next, we look at the incremental update for calculating $\boldsymbol{Z}^{t+1}$

Theorem 4

If $\boldsymbol{Z}^t$ and $\boldsymbol{Q}_{\boldsymbol{X}}^{t+1}$ are known, then

$$\boldsymbol{Z}^{t+1} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{Z}^t \end{bmatrix} + \boldsymbol{v}\boldsymbol{g}' \tag{3.2}$$

where $\boldsymbol{g} = [\tilde{\boldsymbol{x}}^{t+1}, \tilde{\boldsymbol{X}}^t]\boldsymbol{v}$ and $\boldsymbol{v}$ is the last column of $\boldsymbol{Q}_{\boldsymbol{X}}^{t+1}$.

The incremental update to $\boldsymbol{Z}^t$ has two simple steps: append a zero row, and add the rank one matrix $\boldsymbol{v}\boldsymbol{g}'$. Explicitly calculating $\boldsymbol{H}^{t+1}$ to find $\boldsymbol{Z}^{t+1}$ is not needed as computing $\boldsymbol{Q}_{\boldsymbol{X}}^{t+1}$ and then reading off its last column to produce $\boldsymbol{v}$ is sufficient.

Updating $\boldsymbol{Q}_{\boldsymbol{Z}}$:

Theorem 5

Assume the QR factorization $\boldsymbol{Z}^t = \boldsymbol{Q}_{\boldsymbol{Z}}^t \boldsymbol{R}_{\boldsymbol{Z}}^t$ is known. Two sets of $O(p)$ Givens rotations $\boldsymbol{G}_{\boldsymbol{R}} = \boldsymbol{G}_{\boldsymbol{R},1} \dots \boldsymbol{G}_{\boldsymbol{R},(p-1)}$ and $\boldsymbol{G}_{\boldsymbol{B}} = \boldsymbol{G}_{\boldsymbol{B},1} \dots \boldsymbol{G}_{\boldsymbol{B},p}$ can be constructed such that the factorization of $\boldsymbol{Z}^{t+1}$ is

$$\boldsymbol{Z}^{t+1} = \boldsymbol{Q}_{\boldsymbol{Z}}^{t+1} \boldsymbol{R}_{\boldsymbol{Z}}^{t+1} \tag{3.3}$$

$$\boldsymbol{Q}_{\boldsymbol{Z}}^{t+1} = \boldsymbol{Q}_{\boldsymbol{Z}}^{t} \boldsymbol{G}_{\boldsymbol{R}}' \boldsymbol{G}_{\boldsymbol{B}}' \tag{3.4}$$

$$\boldsymbol{R}_{\boldsymbol{Z}}^{t+1} = \boldsymbol{G}_{\boldsymbol{B}} \boldsymbol{G}_{\boldsymbol{R}} \left(\begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{R}_{\boldsymbol{Z}}^t \end{bmatrix} + \boldsymbol{c}\boldsymbol{g}' \right) \tag{3.5}$$

where $\boldsymbol{c} = \boldsymbol{Q}_{\boldsymbol{Z}}^{t\prime} \boldsymbol{v}$ and $\boldsymbol{g}$ is as defined in Theorem 4.

$\boldsymbol{G}_{\boldsymbol{R}}$ and $\boldsymbol{G}_{\boldsymbol{B}}$ are both computed by multiplying $O(p)$ Givens rotation matrices. The proof uses the algorithm in section 12.5.1 of Golub and Loan [2012] for rank one updates of QR decompositions.

Updating $\hat{\boldsymbol{b}}$: $\hat{\boldsymbol{b}}$ is based on the dual solution to the quantile regression and can be directly computed if the primal solution $\boldsymbol{\beta}_1$ is known. $\boldsymbol{\beta}_1$ can be calculated more efficiently by warmstarting the optimization algorithm (i.e., starting the optimization at the previous solution point). Since the changes to $\boldsymbol{Y}$ and $\boldsymbol{X}$ are small, we can expect the previous solution to be relatively close to the new solution. We use the simplex method as the optimization algorithm, because pivot operations are fast, and the algorithm is very efficient when started near the optimal solution.

Algorithm 3 Qsnap Incremental Rank Test

```
Inputs: $\boldsymbol{Y}, \boldsymbol{X}, \tilde{\boldsymbol{X}}, \boldsymbol{x}^{t+1}, \tilde{\boldsymbol{x}}^{t+1}, \boldsymbol{Q_X}, \boldsymbol{R_X}, \boldsymbol{Q_Z}, \boldsymbol{R_Z}, \beta_1, \tau$
$[\boldsymbol{Q_X}, \boldsymbol{R_X}] = \text{rowUpdate}(\boldsymbol{Q_X}, \boldsymbol{R_X}, \boldsymbol{x}^{t+1})$
$\boldsymbol{v} = \boldsymbol{Q_X}.lastColumn$
$\boldsymbol{Q_X} = \text{removeLastColum}(\boldsymbol{Q_X})$
$\tilde{\boldsymbol{X}} = \begin{bmatrix} \tilde{\boldsymbol{x}}^{t+1} \\ \tilde{\boldsymbol{X}} \end{bmatrix}$
$\boldsymbol{g} = \tilde{\boldsymbol{X}}'v$
$[\boldsymbol{Q_Z}, \boldsymbol{R_Z}] = \text{rowUpdate}(\boldsymbol{Q_Z}, \boldsymbol{R_Z}, \boldsymbol{0})$
$[\boldsymbol{Q_Z}, \boldsymbol{R_Z}] = \text{rankOneUpdate}(\boldsymbol{Q_Z}, \boldsymbol{R_Z}, \boldsymbol{v}, \boldsymbol{g}')$
$\beta_1 = \text{simplexSolve}(\boldsymbol{X}, \boldsymbol{Y}, \tau, \beta_1)$
$\boldsymbol{a} = \text{dualSolution}(\boldsymbol{X}, \boldsymbol{Y}, \tau, \beta_1)$
$\hat{\boldsymbol{b}} = \boldsymbol{a} - (1 - \tau)$
$T = \hat{\boldsymbol{b}}'\boldsymbol{Q_Z}\boldsymbol{Q}'_{\boldsymbol{Z}}\hat{\boldsymbol{b}}/(\tau(1-\tau))$
Return($T, \tilde{\boldsymbol{X}}, \boldsymbol{Q_X}, \boldsymbol{R_X}, \boldsymbol{Q_Z}, \boldsymbol{R_Z}, \beta_1$)
```

Full Update: Algorithm 3 shows the entire update process when point j is added to $\boldsymbol{C}$. The functions rowUpdate() and rankOneUpdate() correspond to the QR update algorithms from Golub and Loan [2012]. simplexSolve() uses the simplex algorithm to compute a quantile regression solution, warmstarted at a given solution value. dualSolution() returns the dual solution to the quantile regression given the primal solution.

Both rowUpdate() and rankOneUpdate() can be run in $O(np)$ time, as their main bottleneck is the multiplication of $O(p)$ Givens rotations; the rotation matrices are sparse, so each can be done in $O(n)$ time. The quantile regression dual solution can be calculated in $O(np)$ time when the primal solution is known. For the simplex solver, each pivot operation takes $O(np)$ time. Convergence is guaranteed by the convex solution space, but in general we cannot say how many pivots will be required. Provided the data is reasonably well behaved, warmstarting should significantly reduce the number of pivots required, especially when adding a single new data point. In practice, we have observed that the warmstarted simplex algorithm often converges in a sublinear number of pivots.

3.3.3 Theorem Proofs

Theorem 3

Let $\boldsymbol{x}^{t+1}$ be the new row added to $\boldsymbol{X}^t$ at iteration $t+1$. If the QR factorization $\boldsymbol{Q}^t_{\boldsymbol{X}}\boldsymbol{R}^t_{\boldsymbol{X}} = \boldsymbol{X}^t$ is known, then

$$\boldsymbol{H}^{t+1} = \begin{bmatrix} 1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{H}^t \end{bmatrix} - \boldsymbol{v}\boldsymbol{v}' \tag{3.6}$$

where $\boldsymbol{v}$ is the last column of $\boldsymbol{Q}^{t+1}_{\boldsymbol{X}}$.

Proof: We start by finding the new QR factorization of $\boldsymbol{X}^{t+1}$.

$$\boldsymbol{X}^{t+1} = \begin{bmatrix} \boldsymbol{x}^{t+1} \\ \boldsymbol{X}^t \end{bmatrix} \tag{3.7}$$

$\boldsymbol{Q}^t_{\boldsymbol{X}}$ is an orthonormal basis for $\boldsymbol{X}^t$, meaning $\boldsymbol{Q}^{t\prime}_{\boldsymbol{X}}\boldsymbol{Q}^t_{\boldsymbol{X}} = \boldsymbol{I}$, and thus $\boldsymbol{Q}^{t\prime}_{\boldsymbol{X}}\boldsymbol{X}^t = \boldsymbol{R}^t_{\boldsymbol{X}}$. We can expand the size of $\boldsymbol{Q}^t_{\boldsymbol{X}}$ to account for the increased size of $\boldsymbol{X}^{t+1}$ while maintaining orthonormality, giving us

$$\begin{bmatrix} 1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{Q}^{t\prime}_{\boldsymbol{X}} \end{bmatrix} \boldsymbol{X}^{t+1} = \begin{bmatrix} 1 & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{Q}^{t\prime}_{\boldsymbol{X}} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}^{t+1} \\ \boldsymbol{X}^t \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}^{t+1} \\ \boldsymbol{R}^t_{\boldsymbol{X}} \end{bmatrix} = \tilde{\boldsymbol{R}}_{\boldsymbol{X}} \tag{3.8}$$

In equation 3.8, we use a boldface zero ($\boldsymbol{0}$) to indicate the rest of the row or column of the matrix is filled with 0s.

Our goal is then to refactor $\tilde{\boldsymbol{R}}_{\boldsymbol{X}}$ into an upper triangular matrix while preserving the orthonormality of $\boldsymbol{Q}^{t+1}_{\boldsymbol{X}}$. Givens rotations are a common tool used to accomplish this refactoring efficiently in QR factorization algorithms. $\tilde{\boldsymbol{R}}_{\boldsymbol{X}}$ is upper Hessenberg, meaning we can quickly construct the upper triangular matrix $\boldsymbol{G}'_{\boldsymbol{X}}\tilde{\boldsymbol{R}}_{\boldsymbol{X}}$ using only p Givens rotations $\boldsymbol{G}_{\boldsymbol{X},1}\ldots\boldsymbol{G}_{\boldsymbol{X},p} = \boldsymbol{G}_{\boldsymbol{X}}$ (where $\boldsymbol{G}_{\boldsymbol{X},k}$ indicates the kth Givens rotation matrix). Note that $\boldsymbol{G}_{\boldsymbol{X}}\boldsymbol{G}'_{\boldsymbol{X}} = \boldsymbol{I}$. Thus we have

$$\boldsymbol{G}'_{\boldsymbol{X}} \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \boldsymbol{Q}^{t\prime}_{\boldsymbol{X}} \end{bmatrix} \boldsymbol{X}^{t+1} = \boldsymbol{G}'_{\boldsymbol{X}} \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \boldsymbol{Q}^{t\prime}_{\boldsymbol{X}} \end{bmatrix} \begin{bmatrix} \boldsymbol{x}^{t+1} \\ \boldsymbol{X}^{t} \end{bmatrix} \tag{3.9}$$

$$= \boldsymbol{G}'_{\boldsymbol{X}} \begin{bmatrix} \boldsymbol{x}^{t+1} \\ \boldsymbol{R}^{t}_{\boldsymbol{X}} \end{bmatrix} \tag{3.10}$$

$$= \boldsymbol{R}^{t+1}_{\boldsymbol{X}} \tag{3.11}$$

$$\boldsymbol{Q}^{t+1}_{\boldsymbol{X}} = \begin{bmatrix} 1 & \mathbf{0} \\ \mathbf{0} & \boldsymbol{Q}^{t}_{\boldsymbol{X}} \end{bmatrix} \boldsymbol{G}_{\boldsymbol{X}} \tag{3.12}$$

However, in this form $\boldsymbol{Q}^{t+1}_{\boldsymbol{X}}$ is an $(n+1)\times(p+1)$ matrix, which has one more column than we need for an orthonormal basis of $\boldsymbol{X}^{t+1}$. $\boldsymbol{R}^{t+1}_{\boldsymbol{X}}$ has dimensions $(p+1)\times p$ and is almost a triangular matrix except its last row is $\mathbf{0}$. This means that the last column of $\boldsymbol{Q}^{t+1}_{\boldsymbol{X}}$, denoted as $\boldsymbol{v}$, is part of the left null space of $\boldsymbol{X}^{t+1}$ and contributes nothing to the reconstruction of $\boldsymbol{X}^{t+1}$. Thus we can safely remove this column from $\boldsymbol{Q}^{t+1}_{\boldsymbol{X}}$ and the last row of $\boldsymbol{R}^{t+1}_{\boldsymbol{X}}$ without changing the factorization.

We can represent this removal by multiplication by $\begin{bmatrix} \boldsymbol{I}_p \\ \mathbf{0} \end{bmatrix}$, where $\boldsymbol{I}_p$ is the $p\times p$ identity matrix.

Recall that $\boldsymbol{H} = \boldsymbol{Q}_{\boldsymbol{X}}\boldsymbol{Q}'_{\boldsymbol{X}}$ when $\boldsymbol{Q}_{\boldsymbol{X}}$ is an orthonormal basis for $\boldsymbol{X}$. Let us denote the last column of $\boldsymbol{Q}^{t+1}_{\boldsymbol{X}}$ as $\boldsymbol{v}$. Using the vector removal notation from above, we can rewrite the form of $\boldsymbol{H}^{t+1}$, where $\mathbf{0}_p$ is the $p\times p$ zero matrix:

$$\boldsymbol{H}^{t+1} = \boldsymbol{Q}^{t+1}_{\boldsymbol{X}} \begin{bmatrix} \boldsymbol{I}_p \\ \mathbf{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_p & \mathbf{0} \end{bmatrix} \boldsymbol{Q}^{t+1\prime}_{\boldsymbol{X}} = \tag{3.13}$$

$$\boldsymbol{Q}^{t+1}_{\boldsymbol{X}} \left(\boldsymbol{I}_{p+1} - \begin{bmatrix} \mathbf{0}_p & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \right) \boldsymbol{Q}^{t+1\prime}_{\boldsymbol{X}} = \tag{3.14}$$

$$\boldsymbol{Q}^{t+1}_{\boldsymbol{X}}\boldsymbol{Q}^{t+1\prime}_{\boldsymbol{X}} - \boldsymbol{Q}^{t+1}_{\boldsymbol{X}} \begin{bmatrix} \mathbf{0}_p & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \boldsymbol{Q}^{t+1\prime}_{\boldsymbol{X}} = \tag{3.15}$$

$$\begin{bmatrix}1 & \mathbf{0}\\ \mathbf{0} & \boldsymbol{Q}_{\boldsymbol{X}}^{t}\end{bmatrix}\boldsymbol{G}_{\boldsymbol{X}}\boldsymbol{G}_{\boldsymbol{X}}'\begin{bmatrix}1 & \mathbf{0}\\ \mathbf{0} & \boldsymbol{Q}_{\boldsymbol{X}}^{t\prime}\end{bmatrix} - \boldsymbol{Q}_{\boldsymbol{X}}^{t+1}\begin{bmatrix}\mathbf{0}\\ 1\end{bmatrix}\begin{bmatrix}\mathbf{0} & 1\end{bmatrix}\boldsymbol{Q}_{\boldsymbol{X}}^{t+1\prime} = \tag{3.16}$$

$$\begin{bmatrix}1 & \mathbf{0}\\ \mathbf{0} & \boldsymbol{Q}_{\boldsymbol{X}}^{t}\boldsymbol{Q}_{\boldsymbol{X}}^{t\prime}\end{bmatrix} - \boldsymbol{v}\boldsymbol{v}' = \tag{3.17}$$

$$\begin{bmatrix}1 & \mathbf{0}\\ \mathbf{0} & \boldsymbol{H}^{t}\end{bmatrix} - \boldsymbol{v}\boldsymbol{v}' \tag{3.18}$$

□

Theorem 4

If $\boldsymbol{Z}^t$ and $\boldsymbol{Q}_{\boldsymbol{X}}^{t+1}$ are known, then

$$\boldsymbol{Z}^{t+1} = \begin{bmatrix}\mathbf{0}\\ \boldsymbol{Z}^{t}\end{bmatrix} + \boldsymbol{v}\boldsymbol{g}' \tag{3.19}$$

where $\boldsymbol{g} = [\tilde{\boldsymbol{x}}^{t+1}, \tilde{\boldsymbol{X}}^{t}]\boldsymbol{v}$ and $\boldsymbol{v}$ is the last column of $\boldsymbol{Q}_{\boldsymbol{X}}^{t+1}$.

Proof: From Theorem 3 we know how to express the form of $\boldsymbol{H}^{t+1}$. Let $\tilde{\boldsymbol{x}}^{t+1}$ be the row added to $\tilde{\boldsymbol{X}}^{t}$.

$$\boldsymbol{Z}^{t+1} = \tilde{\boldsymbol{X}}^{t+1} - \boldsymbol{H}^{t+1}\tilde{\boldsymbol{X}}^{t+1} = \tag{3.20}$$

$$\begin{bmatrix}\tilde{\boldsymbol{x}}^{t+1}\\ \tilde{\boldsymbol{X}}^{t}\end{bmatrix} - \left(\begin{bmatrix}1 & \mathbf{0}\\ \mathbf{0} & \boldsymbol{H}^{t}\end{bmatrix} - \boldsymbol{v}\boldsymbol{v}'\right)\begin{bmatrix}\tilde{\boldsymbol{x}}^{t+1}\\ \tilde{\boldsymbol{X}}^{t}\end{bmatrix} = \tag{3.21}$$

$$\begin{bmatrix}\tilde{\boldsymbol{x}}^{t+1}\\ \tilde{\boldsymbol{X}}^{t}\end{bmatrix} - \begin{bmatrix}\tilde{\boldsymbol{x}}^{t+1}\\ \boldsymbol{H}^{t}\tilde{\boldsymbol{X}}^{t}\end{bmatrix} + \boldsymbol{v}\boldsymbol{v}'\begin{bmatrix}\tilde{\boldsymbol{x}}^{t+1}\\ \tilde{\boldsymbol{X}}^{t}\end{bmatrix} = \tag{3.22}$$

$$\begin{bmatrix}\mathbf{0}\\ \boldsymbol{Z}^{t}\end{bmatrix} + \boldsymbol{v}\boldsymbol{g}' \tag{3.23}$$

where $\boldsymbol{g} = [\tilde{\boldsymbol{x}}^{t+1}, \tilde{\boldsymbol{X}}^{t}]\boldsymbol{v}$. □

Theorem 5

Assume the QR factorization $\boldsymbol{Z}^t = \boldsymbol{Q}^t_{\boldsymbol{Z}}\boldsymbol{R}^t_{\boldsymbol{Z}}$ is known. Two sets of $O(p)$ Givens rotations $\boldsymbol{G}_{\boldsymbol{R}} = \boldsymbol{G}_{\boldsymbol{R},1} \dots \boldsymbol{G}_{\boldsymbol{R},(p-1)}$ and $\boldsymbol{G}_{\boldsymbol{B}} = \boldsymbol{G}_{\boldsymbol{B},1} \dots \boldsymbol{G}_{\boldsymbol{B},p}$ can be constructed such that the factorization of $\boldsymbol{Z}^{t+1}$ is

$$\boldsymbol{Z}^{t+1} = \boldsymbol{Q}^{t+1}_{\boldsymbol{Z}}\boldsymbol{R}^{t+1}_{\boldsymbol{Z}} \tag{3.24}$$

$$\boldsymbol{Q}^{t+1}_{\boldsymbol{Z}} = \boldsymbol{Q}^t_{\boldsymbol{Z}}\boldsymbol{G}'_{\boldsymbol{R}}\boldsymbol{G}'_{\boldsymbol{B}} \tag{3.25}$$

$$\boldsymbol{R}^{t+1}_{\boldsymbol{Z}} = \boldsymbol{G}_{\boldsymbol{B}}\boldsymbol{G}_{\boldsymbol{R}}\left(\begin{bmatrix}\boldsymbol{0}\\ \boldsymbol{R}^t_{\boldsymbol{Z}}\end{bmatrix} + \boldsymbol{c}\boldsymbol{g}'\right) \tag{3.26}$$

where $\boldsymbol{c} = \boldsymbol{Q}^{t\prime}_{\boldsymbol{Z}}\boldsymbol{v}$ and $\boldsymbol{g}$ is as defined in Theorem 4.

Proof: Given an existing decomposition $\boldsymbol{Z}^t = \boldsymbol{Q}^t_{\boldsymbol{Z}}\boldsymbol{R}^t_{\boldsymbol{Z}}$,

$$\begin{aligned}\boldsymbol{Z}^{t+1} &= \begin{bmatrix}\boldsymbol{0}\\ \boldsymbol{Z}^t\end{bmatrix} + \boldsymbol{v}\boldsymbol{g}' \\ &= \begin{bmatrix}\boldsymbol{0}\\ \boldsymbol{Q}^t_{\boldsymbol{Z}}\boldsymbol{R}^t_{\boldsymbol{Z}}\end{bmatrix} + \boldsymbol{v}\boldsymbol{g}' \\ &= \boldsymbol{Q}^t_{\boldsymbol{Z}}\left(\begin{bmatrix}\boldsymbol{0}\\ \boldsymbol{R}^t_{\boldsymbol{Z}}\end{bmatrix} + \boldsymbol{c}\boldsymbol{g}'\right)\end{aligned}$$

where $\boldsymbol{c} = \boldsymbol{Q}^{t\prime}_{\boldsymbol{Z}}\boldsymbol{v}$.

We can compute a set of $(p-1)$ Givens rotations $\boldsymbol{G}_{\boldsymbol{R}} = \boldsymbol{G}_{\boldsymbol{R},1} \dots \boldsymbol{G}_{\boldsymbol{R},(p-1)}$ such that $\boldsymbol{G}_{\boldsymbol{R}}\boldsymbol{c} = ||\boldsymbol{c}||\boldsymbol{e_1}$ where $\boldsymbol{e_1}$ is the first unit basis vector. Then $\boldsymbol{G}_{\boldsymbol{R}}\left(\begin{bmatrix}\boldsymbol{0}\\ \boldsymbol{R}^t_{\boldsymbol{Z}}\end{bmatrix} + \boldsymbol{c}\boldsymbol{g}'\right) = \begin{bmatrix}\boldsymbol{0}\\ \tilde{\boldsymbol{R}}\end{bmatrix} + ||\boldsymbol{c}||\boldsymbol{e}_1\boldsymbol{g}' = \boldsymbol{B}$. The matrix $\boldsymbol{B}$ is guaranteed to be upper Hessenberg; as before, we can construct p Givens rotations $\boldsymbol{G}_{\boldsymbol{B}} = \boldsymbol{G}_{\boldsymbol{B},1} \dots \boldsymbol{G}_{\boldsymbol{B},p}$ such that $\boldsymbol{G}_{\boldsymbol{B}}\boldsymbol{B} = \boldsymbol{R}^{t+1}$ is upper triangular. It then follows that $\boldsymbol{Q}^{t+1}_{\boldsymbol{Z}} = \boldsymbol{Q}^t_{\boldsymbol{Z}}\boldsymbol{G}'_{\boldsymbol{R}}\boldsymbol{G}'_{\boldsymbol{B}}$. $\square$

3.3.4 Multiple Hypothesis Test Correction

With multiple hypothesis tests being performed, we cannot use the rank test p-value to determine the significance of the most extreme region. Instead, we apply a Gumbel correction [Abrams et al., 2010], which uses the most significant test values from randomized data permutations to parameterize an extreme value distribution and identify significant values. This approach requires fewer data permutations than the traditional randomization test. If multiple significant regions need to be returned, methods like the false discovery rate [Benjamini and Hochberg, 1995] or Bonferroni correction [Bonferroni, 1936], can be used.

3.4 Results

Evaluating Qsnap on real-world data is challenging due to the lack of datasets with ground-truth identification of which region(s) change (or did not change) from one time period to another. Consequently, we create simulated data where the ground truth is known, and we perform an extensive set of experiments under different simulator settings. We also compare Qsnap against other algorithm on two real-world problems, and we corroborate the results against findings by independent sources[2].

3.4.1 Runtime Analysis

We compare our rank test speedup against two baselines. The first baseline is a naive implementation that calculates the test statistic T from scratch every iteration; the second is a naive implementation that uses warmstarting to re-learn $\boldsymbol{\beta}_1$ quickly each iteration.

Table 3.1 shows the average update time of each algorithm as the size of the dataset increases, with the number of features constant at $p = 3$. Our incremental algorithm is by far the fastest for larger n, demonstrating a linear increase in runtime while the others increase quadratically. The difference between the naive and warmstarted algorithms is relatively small, showing that the primary computational bottleneck is in calculating $\boldsymbol{Z}(\boldsymbol{Z}'\boldsymbol{Z})^{-1}\boldsymbol{Z}'$.

[2]Matlab code for our experiments and algorithms can be found at https://github.com/moortrav/Qsnap

n	1000	2000	4000	6000	8000
Naive	1.8	6.1	20.9	48.6	81.4
Warmstart	1.1	4.4	17.6	43.0	74.9
Incremental	**0.3**	**0.4**	**0.8**	**1.1**	**1.4**

Table 3.1: Average test statistic update time in ms, averaged over 1000 updates for randomly generated data.

3.4.2 Simulation Experiments

We evaluate the accuracy of Qsnap on simulated data where the changed regions are known. We compare Qsnap against two other quantile spatial scan algorithms, with the first being a variant of the Treatment-Effect Spatial Scan (TESS) [McFowland et al., 2018]. We adapt TESS to the snapshot scan setting by using the first and second snapshots as the "control" and "treatment" groups, respectively. In the original paper, p-values for each data point are calculated non-parametrically as the ratio of points with a higher response value. For our multi-variate data, we compute a quantile regression on the control set at the desired quantile, compute the residuals for this regression on the treatment set, then find the non-parametric p-values of those residuals. We found this approach produces far better results than assuming a parametric distribution form. Like Qsnap, TESS can find subsets of the second snapshot that most differ from their expected distribution with respect to a specific quantile. We replace the discrete subset search from the original TESS paper with the same search used by Qsnap, for a more direct comparison of the hypothesis tests.

To our knowledge, TESS is the only existing algorithm that can be directly applied to our task. We create a second baseline using the SSS framework. First, we modify the SSS to search for the most significant region between two snapshots in time. Second, we replace the likelihood ratio test of the SSS with Mood's hypothesis test [Mood, 1950] so that we can compare the τth quantiles of the regions under consideration. For a given region $\boldsymbol{C}$, this test fits the quantile regression $Q_{\boldsymbol{Y}_{\boldsymbol{C}}^{(1)}}(\tau|\boldsymbol{X}_{\boldsymbol{C}}^{(1)})$, then performs a 2×2 chi-squared test on the number of points above and below the regression line in snapshot one and snapshot two. We refer to this algorithm as SSS-Moods.

3.4.2.1 Simulator

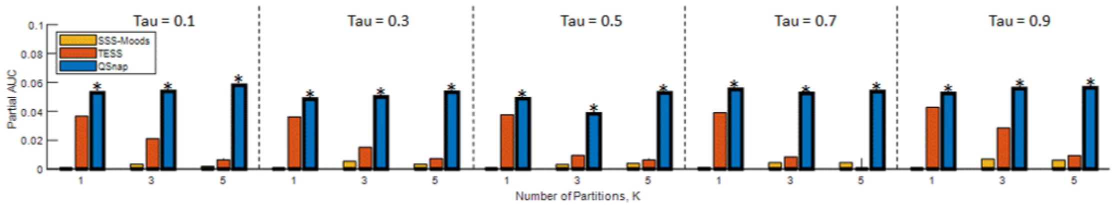
Our simulator generates data points as a tuple $\{y_i, \boldsymbol{x}_i, \boldsymbol{l}_i\}$. The location ($\boldsymbol{l}_i$) and parameter values ($\boldsymbol{x}_i$) are created uniformly at random between a set of maximum and minimum values. Each dataset is partitioned into K spatially contiguous sections. For each section k, the response, y_i, is calculated from a linear relationship $y_i = \boldsymbol{\beta}^k \boldsymbol{x}_i + \epsilon$, where $\boldsymbol{\beta}^k$ is a randomly generated parameter vector that is different for each section, and ϵ is a random noise term. In our experiments, we compare normal, exponential, and uniform distributions for the random noise term ϵ. Our simulator models data that is globally heterogeneous, but homogeneous for specific local spatial areas.

Two data snapshots of n points each are generated using the same partition boundaries and values of $\boldsymbol{\beta}^1, \ldots, \boldsymbol{\beta}^K$. In the second snapshot, a partition j is designated as the target area. In the target area, a subset of the response values are generated from a shifted distribution as $y_i = (\boldsymbol{\beta}^j + \boldsymbol{\delta})\boldsymbol{x}_i + \epsilon$, where $\boldsymbol{\delta}$ is a random vector with $||\boldsymbol{\delta}|| = p$. When generating data, we can identify which quantile q_i of the error distribution each data point falls into. Any point in the target area with a quantile value q_i such that $|\tau - q_i| \leq 0.1$ is generated from this shifted distribution with parameters $\boldsymbol{\beta}^j + \boldsymbol{\delta}$, while the rest are generated with $\boldsymbol{\beta}^j$. This effectively creates a change in 20% of the distribution in the target area, centered around the τth quantile; detecting this change in the simulated datasets is in general a very challenging problem. A successful algorithm will find the area j by performing tests at the τth quantile.

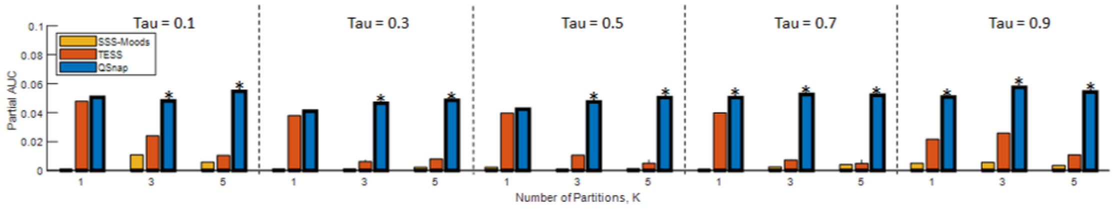
3.4.2.2 Evaluation Metric

We evaluate the algorithms' true positive rate (TPR) versus false positive rate (FPR) curve, calculated on a per-point basis. Only points generated from the shifted distribution are counted as true positives. Typically, a good summary of this curve can be captured with the area under the curve (AUC). In a real-world setting, we would never realistically operate the algorithm under a high false positive rate and the AUC for high FPR values is not meaningful. As a result, we use the partial AUC to measure performance; specifically, we report the AUC from $FPR = [0, 0.2]$ to emphasize lower FPR values. This makes a value of 0.2 a perfect partial AUC score. We compute the partial AUC for each algorithm on each dataset, and report the average value across 30 randomly generated datasets. See Appendix A for calculation details.

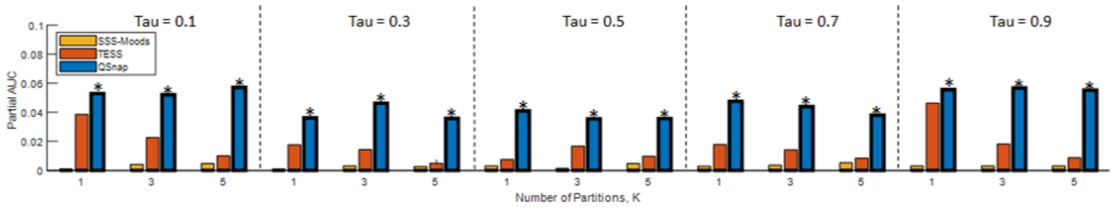
3.4.2.3 Simulation Results



(a) Normal Noise Distribution.



(b) Exponential Noise Distribution.



(c) Uniform Noise Distribution.

Figure 3.1: Partial AUC of TESS, SSS-Moods, and Qsnap on simulated data. The best performing algorithms are bolded, * indicates the best algorithm is statistically significant (Wilcoxon signed-rank test, $\alpha = 0.05$).

We performed a suite of experiments to compare the performances of Qsnap, TESS, and SSS-Moods. In each experiment, each algorithm is tasked with finding the target area in snapshot two where 20% of the points are generated from the shifted distribution. Each algorithm performs its search on the τth quantile, the center of the distribution shift. Each algorithm uses the same spatial search routine, allowing a direct comparison of their hypothesis tests. We evaluate each algorithm's performance by the most significant area reported by each algorithm on each dataset.

We tested each algorithm on simulated data with normal, exponential, and uniform noise distributions, with distribution changes centered at quantiles $\tau = 0.1, 0.3, 0.5, 0.7, 0.9$ in the target area. We also varied the number of partitions K between 1 and 3. For $K = 1$, the snapshots have the same base distribution at all locations, and the target region is a random circle in $\boldsymbol{L}$.

Figure 3.1 shows the average partial AUCs for each algorithm at different values of τ, K, and different forms of the noise distribution. These experiments were run for $n = 5000$ and $p = 5$. The size of the target area in snapshot two was set at 1000 points, meaning approximately 200 points generated from the shifted distribution. We do not show experiments with changing values of n, p, or target area size, as these parameters affected each algorithm in similar, intuitive ways.

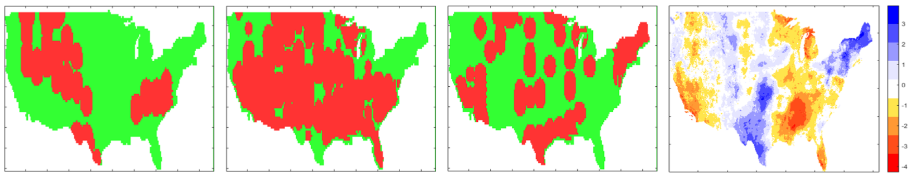
SSS-Moods performs very poorly in these experiments, barely detecting any changes. The low-power Mood's quantile test is ill-suited to the difficulty of this problem. TESS does well for $K = 1$, but poorly on larger K. The speed of TESS is dependent on fitting a global model instead of many local models, making it ill-suited for data with local variation.

Qsnap performs significantly better than the other algorithms in 42/45 of the experiments. Like SSS-Moods, it fits a model to each local region, allowing it to account for spatially varying distributions. It also uses a more powerful test statistic, giving it significantly greater accuracy than the other algorithms.

The runtimes of the algorithms, averaged over 10 datasets ($n = 5000$, $p = 5$, and $\tau = 0.7$) are: 87 seconds (Qsnap), 25 seconds (SSS-Moods) and 0.2 seconds (TESS). SSS-Moods and TESS perform simpler hypothesis tests and are faster than Qsnap. However, the simpler tests cause them to be unable to detect many changed regions and they lack robustness. In the next section, we show that both TESS and SSS-Moods also perform poorly on real-world data.

3.4.3 Drought Detection

We use Qsnap, TESS, and SSS-Moods to detect changes in drought conditions in the continental US using climate data collected from `climateengine.org` [Huntington et al., 2017]. Our model uses precipitation as the response, with humidity, evaporation, mean temperature, max temperature, and soil moisture (5cm level) as covariates. We use 2001, a relatively mild drought year for most of the country, and 2007, which had extensive



(a) TESS (b) SSS-Moods (c) Qsnap (d) Change in SPEI

Figure 3.2: (a)-(c) Significant areas found on climate data. (d) Change in SPEI between 2001 and 2007.

	Avg. $\Delta(C)$	$\Delta(\overline{C})$	Avg. $\Delta_\tau(C)$
TESS	0.76	0.95	0.66
SSS-Moods	0.87	0.95	0.86
Qsnap	**1.20***	**0.81**	**1.17***

Table 3.2: Evaluation of detected regions for climate data using change in SPEI on regions found, regions omitted, and the τth quantile change in regions found. * for statistical significance (paired t-test, $\alpha = 0.05$)

droughts in California and the South, as our two time snapshots to compare.

Each algorithm is tasked with finding areas of significant change between the two years at the 10th percentile. Tuning the algorithms to a low percentile of precipitation makes them better suited to finding changes in drought conditions. Since our dataset contains many areas of significant drought change, each algorithm reports all significant regions instead of the most significant, using the Holm–Bonferroni correction with $\alpha = 0.01$ [Holm, 1979]. Though the observations in this data are in a consistent grid structure, our algorithm does not require such conditions.

To evaluate the regions returned by each algorithm, we use the Standardised Precipitation-Evapotranspiration Index (SPEI). SPEI is a numerical measure of drought severity, calculated based on the difference of precipitation and potential evaporation. These SPEI values are independent from our climate data, and we use the difference in SPEI between 2007 and 2001 as a proxy for the (unknown) ground truth.

In Table 3.2 we report three evaluation metrics. First is $\Delta(C)$, the average change in SPEI for observations in each detected region C, using the absolute difference of each observation. An algorithm that more accurately detects regions of high change will have a higher value. The second metric is the average absolute change in all regions not detected by the algorithm, $\overline{C}$, which will be lower for better performers. We also report the average absolute change in the 10th percentile of each detected region, $\Delta_\tau(C)$, which should increase for better performers. Qsnap performs the best for all three algorithms in all of these categories.

Figure 3.2 shows the significant areas found by each algorithm, along with a heat map of the change in SPEI between 2007 and 2001. Red and orange areas in the heat map experienced increased drought conditions, while blue areas decreased. Qsnap finds many areas of significant change, while including fewer areas of no change. In comparison, TESS fails to identify many of the areas with the most significant change, and SSS-Moods identifies nearly the entire map as changing significantly. Note that we do not claim Qsnap is the best tool for drought detection – only that it outperforms TESS and SSS-Moods.

3.4.4 Discovering Migration Paths in eBird

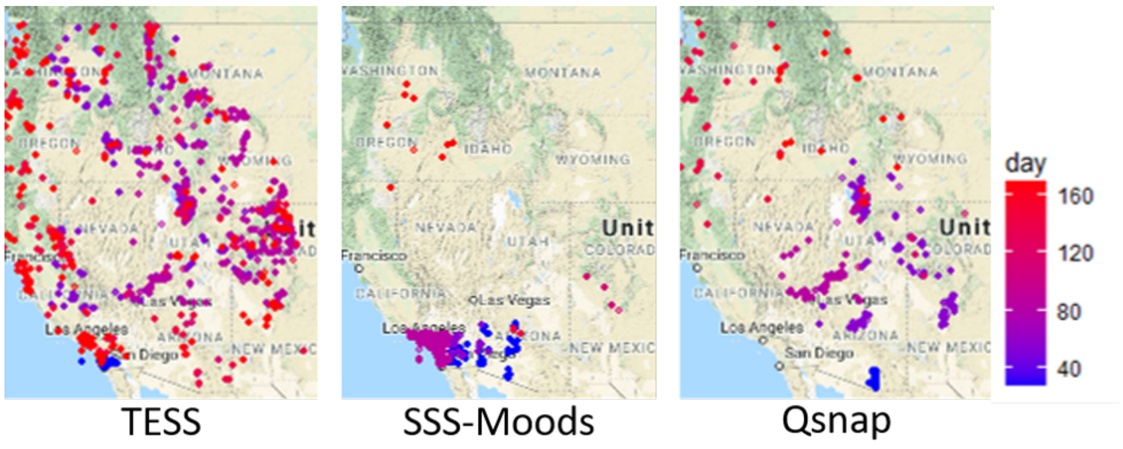


Figure 3.3: Most significant regions ($\tau = 0.9$) found on eBird data. Regions are color coded for time of year.

We applied Qsnap to eBird checklists [Sullivan et al., 2014], which record bird observations identified by species by citizen scientists. We aim to discover migration routes by identifying areas with an unusual influx of birds between time periods. By segmenting the migration period into snapshots, and running Qsnap on those snapshots, the anomalous regions found should discover the migration path. We identify unusual regions by finding changes in the 90th percentile of the number of birds reported on checklists in a region.

We restricted our dataset to the western flyway of the United States, during 2017. We only included checklists from that year, and in Bird Conservation Regions[3] (BCRs) 5, 9-10, 15-16, 32-35. Past work [La Sorte and Fink, 2017] has shown that migrating birds pass through this area during the first half of the year, traveling north from Central America and staying west of the Rocky Mountains. We divided the data from the first half of the year into 6 snapshots spanning 4 weeks each, and ran Qsnap, TESS, and SSS-Moods on pairs of consecutive snapshots. Note that, unlike other studies of migration patterns (e.g. La Sorte and Fink [2017]), we did not filter our dataset to only include migrating species or smooth the observations over space. Our dataset included every checklist for every species in the region and time frame, making this dataset very challenging due to the noise from non-migratory bird species and the imperfect observation process. We used time spent observing as the covariate, and number of individual birds seen as the response. For all algorithms, we used expanding rectangles instead of circles for the spatial scan search because rectangles can better detect longer regions that capture the South to North migration. Each rectangle is allowed to expand to the width of the data area, with its height constrained to a reasonable distance that the birds could travel between time frames.

Figure 3.3 shows the areas[4] reported by each algorithm over the 6 snapshots of data from the first 168 days of 2017. The areas are represented by the individual checklists inside them, and color coded for time of year. The areas found by TESS were large and sporadic, with no clear pattern over time. The areas found by SSS-Moods seem to identify the start and end of the expected migration path, but fail to connect them well. Qsnap produced the best South to North gradient over time, identifying a clear progression that closely matches the migration route shown in La Sorte and Fink [2017]. Being able to identify a shifting spatial trend in the complex and highly noisy eBird dataset without any data filtering illustrates the robustness and usefulness of Qsnap.

[3]See http://nabci-us.org/resources/bird-conservation-regions-map/ for the regions

[4]All maps generated using ggmaps in R [Kahle and Wickham, 2013]

3.5 Conclusion

We presented Qsnap, which finds a region in spatial data that has undergone the most significant change between two data snapshots, with respect to the τth quantile. To reduce the computational cost of a search over multiple regions, we developed an efficient incremental update to the rank test that makes the algorithm scalable to large datasets. Compared to other similar algorithms, Qsnap is less dependent on assumptions about the data, and performs better on a variety of different distributions. Qsnap was also better suited to our two real-world data analysis tasks.

Chapter 4: Multiview Collective Graphical Models

4.1 Introduction

Modeling population movement is an essential component of many research problems in a wide variety of fields such as urban planning, economics and ecology. Building these spatio-temporal models is challenging because data at the individual level is typically not available. There are many reasons for this lack of data such as privacy concerns, insufficient sensor coverage and logistical challenges in collecting this type of data for a large population. In several cases, however, aggregate data are available, such as the total number of cars observed at a traffic intersection at a given time.

Collective Graphical Models (CGMs) [Sheldon and Dietterich, 2011] are probabilistic graphical models that connect individual behavior with aggregate data observed at specific locations and times. In CGMs, the behavior of an individual can be described with a latent *individual model* (e.g., a Markov Chain [Sheldon et al., 2007]) and a collection of these latent individual models produces the observed aggregate data. There are two key challenges with CGMs. First, exact inference is computationally intractable, and several approaches for approximate inference have been developed [Liu et al., 2014, Nguyen et al., 2016, Sheldon et al., 2013, Sun et al., 2015]. Second, the CGM is under-specified, as there are many possible individual behaviors that can produce the aggregate data. In our work, the primary goal is to address the second issue by incorporating information from multiple correlated datasets of aggregated data; we also keep the first issue in mind, as we need to incorporate this additional information without incurring a large computational cost.

Incorporating information from multiple datasets can be achieved through multiview learning [Li et al., 2019, Sun, 2013, Xu et al., 2013], where *views* correspond to different kinds of data. For instance, one view for a webpage classifier could be a set of features derived from the text in the page, while another view could be a set of features derived from the hyperlinks pointing to the webpage [Blum and Mitchell, 1998]. Traditionally, each view on its own is assumed to be sufficient to train a learning algorithm that performs well. The general strategy for multiview learning is to leverage the relationship between different views of the data and, in doing so, learn a model that generalizes better than a model that is learned from a single view.

Our work is the first to extend CGMs to the multiview setting with aggregate data coming from different views. For example, we can infer population movement across the US using the number of jobs in each state (view 1) and the population in each state (view 2), both collected over multiple years. We introduce a novel hierarchical Bayesian model for incorporating multiview information into CGMs through Dirichlet priors on the transition population (the number of individuals that move between each location at each time), and we highlight critical details for making approximate inference on this model efficient. When compared against learning multiple CGMs independently, our method is more accurate at modeling latent population movement while being computationally efficient, especially for large spatio-temporal grids[1].

4.2 Related Work and Background

4.2.1 Collective Graphical Models

CGMs [Sheldon et al., 2007], shown as plate notation in Figure 4.2(a), are a class of latent variable graphical models that represent the distribution of the movements of a group of individuals, consistent with observed aggregate statistics at specific locations and times. The nodes of the graph are the set of locations $\boldsymbol{L}$, and the edges are defined by the neighborhood function $n(i \in \boldsymbol{L})$ which returns the set of locations $j \in \boldsymbol{L}$ that are reachable from i in a single time step. Each edge $\{i, j\}$ will have an associated probability $\theta_{(i,j)}$, with $\sum_j \theta_{(i,j)} = 1$. These transition probabilities may change with respect to time, in which case t will be added as a subscript.

Let x_t be an individual's location at time t. If we have a sequence of movements $\boldsymbol{x} = \{x_1, ..., x_T\} \subset \boldsymbol{L}^T$ by an individual through this graph, then the likelihood of that sequence is $p(\boldsymbol{x}) = \prod_{t=1}^{T-1} \theta_{(x_t, x_{t+1}, t)}$.

Now consider a population of N individuals moving through the graph simultaneously, with movement sequences $\boldsymbol{x}^{(1)}, ..., \boldsymbol{x}^{(N)}$. Instead of tracking the separate movements of each individual, CGMs let us describe the joint distribution of movements

[1]This work is currently under review for NeurIPS 2021

in terms of aggregate statistics. Let $\boldsymbol{Z} = \{Z_{(i,t)}\}_{t=1,i\in\boldsymbol{L}}^{T}$ be the set of populations s.t. $Z_{(i,t)} = \sum_{p=1}^{N} I(x_t^{(p)} = i)$. In addition, let $\boldsymbol{M} = \{M_{(i,j,t)}\}_{t=1,i\in\boldsymbol{L},j\in n(i)}^{T}$ be the set of transition populations. We define $M_{(i,j,t)} = \sum_{p=1}^{N} I(x_t^{(p)} = i)I(x_{t+1}^{(p)} = j)$ i.e. the transition population moving from location i to j at time t. The conditional distribution of $\boldsymbol{M}$ is

$$p(\boldsymbol{M}|\boldsymbol{Z},\boldsymbol{\theta}) = \prod_{i\in\boldsymbol{L}}\prod_{t=1}^{T} \frac{Z_{(i,t)}!}{\prod_{j\in n(i)} M_{(i,j,t)}!} \prod_{j\in n(i)} \theta_{(i,j,t)}^{M_{(i,j,t)}} \tag{4.1}$$

Figure 4.1 shows an example relationship between a single location population and the transitions populations that flow from it. Z_5 is the observed population at a given time t for location 5. Each M value then denotes the amount of that population that flows to each neighbor location as the time transitions to $t+1$. Our constraints say that $Z_5 = \sum_{i=1}^{9} M_{5,i}$.

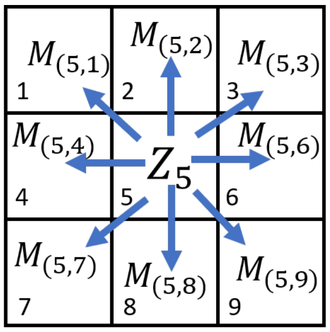


Figure 4.1: Example population flow for grid locations at a single time step. Locations are labeled 1-9. Z_5 is the population at location 5. The values of $\boldsymbol{M}$ are the transition populations that move from location 5 to each of it's neighbors.

$\boldsymbol{Z}$ and $\boldsymbol{M}$ are integer valued, which makes general MAP inference intractable [Sheldon et al., 2013]. In practice this condition is relaxed, allowing $\boldsymbol{Z}$ and $\boldsymbol{M}$ to be real-valued and non-negative. As a model governing the flow of a population through space and time, there are also flow constraints that need to be incorporated when learning $\boldsymbol{M}$, namely $\sum_j M_{(i,j,t)} = Z_{(i,t)}$ and $\sum_i M_{(i,j,t)} = Z_{(j,t+1)}$. These ensure that the total flow going into and coming out of a location match the total population at that time. A variety of approximate inference techniques for CGMs have been developed, including a tractable convex relaxation [Sheldon et al., 2013], Gaussian approximations [Liu et al., 2014], message passing [Sun et al., 2015] and a difference-of-convex functions program [Nguyen et al., 2016].

CGMs are a natural fit to population tracking, where individual transitions are often unknown, but group dynamics can be inferred from aggregate observations at different times and locations. While many models may be suited to modeling the movements of a single individual, such as Hidden Markov Models and Kalman filters [Welch et al., 1995], they are unable to predict group dynamics when individual observations are missing.

CGMs have been extended in a variety of ways including differential privacy [Bernstein et al., 2017], collective graphical mixture models [Iwata et al., 2017] and incorporating neural networks into CGMs to model the transition probabilities [Iwata and Shimizu, 2019]. However, none of these extensions so far have included multi-view learning.

4.2.2 Multiview Learning

Multiview learning [Sun, 2013, Xu et al., 2013] has been applied to many areas of machine learning, including semi-supervised learning [Blum and Mitchell, 1998], clustering [Yang and Wang, 2018] and representation learning [Li et al., 2019]. The literature is extensive, and we will focus on multiview learning techniques that are closely related to our approach of capturing the relationship between views by aligning lower dimensional representations of each view. Li et al. [2019] call this category of approaches multi-view representation alignment and it includes similarity-based [Frome et al., 2013, Karpathy and Fei-Fei, 2017], distance-based [Li et al., 2003, Feng et al., 2014] and correlation-based [Hotelling, 1936, Bach and Jordan, 2005] measures of alignment.

Correlation-based alignment relies on Canonical Correlation Analysis (CCA) [Hotelling, 1936]. Given two data matrices $\boldsymbol{X}^1$ and $\boldsymbol{X}^2$, which consist of observations of random vectors $\boldsymbol{x}^1$ and $\boldsymbol{x}^2$, CCA computes the projections $\boldsymbol{U}^1$ and $\boldsymbol{U}^2$ such that $corr(\boldsymbol{X}^1\boldsymbol{U}^1, \boldsymbol{X}^2\boldsymbol{U}^2) = \boldsymbol{P}$ where $\boldsymbol{P}$ is a diagonal matrix with $\boldsymbol{p}$ on the diagonal and each value p_i maximized in sequence. The diagonal entries of $\boldsymbol{P}$ are the canonical correlations, which gives an indication of how much information is shared between the views along each projected axis.

Many extensions to CCA have been proposed including Sparse CCA [Witten et al., 2009], Kernel CCA [Lai and Fyfe, 2000, Akaho, 2001] and Deep CCA [Andrew et al., 2013]. We focus on probabilistic CCA (PCCA) [Bach and Jordan, 2005] because we need a probabilistic approach to CCA in order to integrate it with CGMs. Bach and Jordan [2005] showed that CCA is equivalent to solving a latent variable model, with $\boldsymbol{x}^1$ and $\boldsymbol{x}^2$ assumed to be normally distributed. In this probabilistic CCA (PCCA) interpretation, $\boldsymbol{x}^1$ and $\boldsymbol{x}^2$ are seeded by the values of a latent variable $\boldsymbol{s} \sim N(\boldsymbol{0}, \boldsymbol{I})$, using the following relationships:

$$\begin{aligned} \boldsymbol{x}^1 | \boldsymbol{s} &\sim N(\boldsymbol{s}\boldsymbol{W}^1 + \boldsymbol{\mu}^1, \boldsymbol{\Phi}^1) \qquad (4.2) \\ \boldsymbol{x}^2 | \boldsymbol{s} &\sim N(\boldsymbol{s}\boldsymbol{W}^2 + \boldsymbol{\mu}^2, \boldsymbol{\Phi}^2) \end{aligned}$$

$$\begin{aligned} \boldsymbol{\mu}^v &= \text{mean}(\boldsymbol{x}^v) \qquad (4.3) \\ \boldsymbol{W}^v &= \sqrt{\boldsymbol{P}}\boldsymbol{U}^v\boldsymbol{\Sigma}^{vv} \\ \boldsymbol{\Phi}^v &= \boldsymbol{\Sigma}^{vv} - \boldsymbol{W}^{v\prime}\boldsymbol{W}^v \end{aligned}$$

In this model, $\boldsymbol{s}$ represents the information that is shared between the views, which is projected by $\boldsymbol{W}^v$ and offset by $\boldsymbol{\mu}^v$, while $\boldsymbol{\Phi}^v$ is the amount of variability remaining in view v when this shared information is accounted for. $\boldsymbol{W}^v$ is calculated using the canonical covariates $\boldsymbol{U}^v$ and canonical correlations $\boldsymbol{P}$ from CCA.

4.3 Methodology

We consider the scenario where the data consists of observations about two distinct populations moving through the same locations at the same time. Each of these population movements can be considered as a different view of some underlying latent process that influences individuals from both groups. Thus, both populations possess some shared information, and we can improve estimates on both views by accounting for what information is shared and what is unique to each population.

We first describe the algorithm we use as a default CGM optimizer. We then explain the pitfalls of a naive approach to multiview CGMs in Section 4.3.2. Finally, we present our unique multiview CGM method in Section 4.3.3, with potential modifications in 4.3.4.

4.3.1 CGM Optimization

In our CGM model, shown in plate notation in Figure 4.2a, we are tasked with estimating both $\boldsymbol{M}$ and $\boldsymbol{\theta}$ given the observations $\boldsymbol{Z}$. To estimate $\boldsymbol{M}$ we use the Non-Linear Belief Propagation (NLBP) algorithm of Sun et al. [2015] over the DC approach of Nguyen et al. [2016], as the latter assumes a noise model over observations, and we are not in that setting.

Using Stirling's approximation, we can express the log-likelihood of Equation 4.1 as

$$L(\boldsymbol{M}|\boldsymbol{Z},\boldsymbol{\theta}) = \sum_{i\in\boldsymbol{L}}\sum_{t=1}^{T} Z_{(i,t)}\log(Z_{(i,t)}) - \sum_{j\in n(i)} M_{(i,j,t)}\log(M_{(i,j,t)}) + \sum_{j\in n(i)} M_{(i,j,t)}\log(\theta_{(i,j,t)}) \tag{4.4}$$

The goal then is to maximize this likelihood (or minimized the negative log-likelihood) with respect to $\boldsymbol{M}$. Sun et al. [2015] made the observation that the negative log-likelihood can be decomposed as $-L(\boldsymbol{M}|\boldsymbol{Z},\boldsymbol{\theta}) = E_{CGM}(\boldsymbol{M}) - H_B(\boldsymbol{M},\boldsymbol{Z})$, where

$$E_{CGM}(\boldsymbol{M}) = -\sum_{i\in\boldsymbol{L}}\sum_{t=1}^{T}\sum_{j\in n(i)} M_{(i,j,t)}\log(\theta_{(i,j,t)}) \tag{4.5}$$

$$H_B(\boldsymbol{M},\boldsymbol{Z}) = \sum_{i\in\boldsymbol{L}}\sum_{t=1}^{T} Z_{(i,t)}\log(Z_{(i,t)}) - \sum_{j\in n(i)} M_{(i,j,t)}\log(M_{(i,j,t)}) \tag{4.6}$$

$H_B(\boldsymbol{M},\boldsymbol{Z})$ is the Bethe entropy of the graph [Yedidia et al., 2005], while $E_{CGM}(\boldsymbol{M})$ is referred to as the CGM energy function. In this current form, the CGM energy function matches the energy function of the Bethe free energy objective for a standard graphical model [Yedidia et al., 2000]. This means that the values of $\boldsymbol{M}$ can be found using belief propagation, re-normalized to match the population values. However, this would not incorporate the flow constraints of the model.

Sun et al. [2015] developed a generalized belief propagation algorithm for non-linear energy functions (NLBP). The flow constraints of the model are incorporated as a function $C(\boldsymbol{Z}|\boldsymbol{M})$, which quantifies how close the predicted incoming and outgoing flow values of $\boldsymbol{M}$ are to the observed values of $\boldsymbol{Z}$. The form of $C()$ is an implementation choice, based on how the user wants to model the flow constraints. The only require-ment is that $C()$ is differentiable with respect to $\boldsymbol{M}$. The CGM energy function is then modified to be

$$E_{CGM}(\boldsymbol{M}) = -\sum_{i \in \boldsymbol{L}} \sum_{t=1}^{T} \sum_{j \in n(i)} M_{(i,j,t)} \log(\theta_{(i,j,t)}) - C(\boldsymbol{Z}|\boldsymbol{M}) \tag{4.7}$$

The NLBP algorithm equates to performing standard belief propagation with the modified potentials $\hat{\theta}_{(i,j,t)} = \exp\left(-\frac{\partial E_{CGM}(\boldsymbol{M})}{\partial M_{(i,j,t)}}\right)$. The two step process is used along with a dampening factor to update $\boldsymbol{M}$ as illustrated in Algorithm 4. This process is repeated until convergence.

Algorithm 4 Non-Linear Belief Propagation $\boldsymbol{M}$ update

```
  Input: $\boldsymbol{Z}$, $\boldsymbol{\theta}$, initial $\boldsymbol{M}$, $\beta > 0$
  $\hat{\theta}_{(i,j,t)} = \exp\left(-\frac{\partial E_{CGM}(\boldsymbol{M})}{\partial M_{(i,j,t)}}\right) \forall i, j, t$
  $\boldsymbol{M}^{NEW} = BP(\hat{\boldsymbol{\theta}}, \boldsymbol{Z})$
  $\boldsymbol{M} = (1 - \beta)\boldsymbol{M} + \beta \boldsymbol{M}^{NEW}$
```

NLBP normalizes the beliefs by the population values, which handles the outgoing flow constraints by setting $\sum_j M_{(i,j,t)} = Z_{(i,t)}$. We enforce incoming flow constraints by choosing the soft constraint function

$$C(\boldsymbol{Z}|\boldsymbol{M}) = -\frac{\lambda}{2} \sum_{i \in \boldsymbol{L}} \sum_{t=1}^{T} \left(Z_{(i,t+1)} - \sum_{j \in n(i)} M_{(j,i,t)} \right)^2 \tag{4.8}$$

Estimating $\boldsymbol{\theta}$ along with $\boldsymbol{M}$ makes the model under-specified, making it difficult to find a meaningful joint solution. In practice this is combated by adding some structure to $\boldsymbol{\theta}$ to inform its solution, such as making $\boldsymbol{\theta}$ a function of a set of transitions features, or sharing values of $\boldsymbol{\theta}$ across multiple times and/or locations.

We adopt a similar Bayesian method that was used in Iwata et al. [2017], where a Dirichlet prior is estimated over subdivisions of $\boldsymbol{\theta}$. How these partitions are defined can be tailored to each application. For example, the priors could be shared over all locations, but differ at each time step if time is expected to be the main source of change in the data.

Let $\boldsymbol{g}$ be a partition of the CGM graph over $\boldsymbol{L}$ and T. We estimate the parameters $\boldsymbol{\alpha_g}$ of a Dirichlet distribution, used as a prior for all $\boldsymbol{\theta_g}$. Utilizing the relationship between the Dirichlet and multinomial distributions, we can estimate $\boldsymbol{\theta}$ using the Dirichlet prior and current estimates for $\boldsymbol{M}$.

$$\forall (i,t) \in \boldsymbol{g} : \boldsymbol{\theta}_{(i,:,t)} = \frac{\boldsymbol{\alpha}_{\boldsymbol{g}} + \boldsymbol{M}_{(i,:,t)}}{\sum_j \alpha_{\boldsymbol{g},j} + M_{(i,j,t)}} \tag{4.9}$$

Here $(i, :, t)$ is a vector indexing of the values $(i, j, t)\forall j$. To update $\boldsymbol{\alpha}_{\boldsymbol{g}}$, we can used the fixed-point equation derived in [Minka, 2000],

$$\alpha^{NEW}_{\boldsymbol{g},j} = \alpha_{\boldsymbol{g},j} \frac{\sum_{(i,t)\in \boldsymbol{g}} \Psi(M_{(i,j,t)} + \alpha_{\boldsymbol{g},j}) - \Psi(\alpha_{\boldsymbol{g},j})}{\sum_{(i,t)\in \boldsymbol{g}} \Psi(\sum_k M_{(i,k,t)} + \alpha_{\boldsymbol{g},k}) - \Psi(\sum_k \alpha_{\boldsymbol{g},k})} \tag{4.10}$$

Both of these updates depend on the current estimate of $\boldsymbol{M}$. We perform them concurrently with the estimation of $\boldsymbol{M}$, as illustrated in Algorithm 5.

Algorithm 5 CGM optimization

```
Input: $\boldsymbol{Z}$, initial $\boldsymbol{M}$, initial $\boldsymbol{\theta}$, partitions $\boldsymbol{g} \in \boldsymbol{G}$
while !converged do
    $\boldsymbol{M} = \text{NLBP}(\boldsymbol{Z}, \boldsymbol{\theta}, \boldsymbol{M})$
    $\boldsymbol{\theta}_{(i,:,t)} = \text{normalize}(\boldsymbol{\alpha}_g + \boldsymbol{M}_{(i,:,t)})\ \forall (i,t) \in \boldsymbol{g}\ \forall \boldsymbol{g}$
    while !converged do
        $\alpha_{\boldsymbol{g},j} = \alpha_{\boldsymbol{g},j} \frac{\sum_{(i,t)\in \boldsymbol{g}} \Psi(M_{(i,j,t)} + \alpha_{\boldsymbol{g},j}) - \Psi(\alpha_{\boldsymbol{g},j})}{\sum_{(i,t)\in \boldsymbol{g}} \Psi(\sum_k M_{(i,k,t)} + \alpha_{\boldsymbol{g},k}) - \Psi(\sum_k \alpha_{\boldsymbol{g},k})}\ \forall j, \boldsymbol{g}$
    end while
end while
```

4.3.2 A Naive Approach to Combining CGMs With CCA

In the NLBP algorithm, $\boldsymbol{M}$ is updated with a message passing scheme that incorporates the gradient of the CGM likelihood. A straightforward approach in adapting CGMs to a multiview setting is to incorporate a CCA term into the model connecting two views. The probabilistic CCA of Bach and Jordan [2005] works for this, as it re-parameterizes the CCA objective as a latent variable model.

This naive approach adds the PCCA likelihood terms to the CGM model, and includes their gradient in the NLBP update. Section B of the Appendix details this approach, along with the complex and lengthy gradient of the PCCA term. The latent variables of PCCA can either be linked to the view-specific values of $\boldsymbol{M}$ or $\boldsymbol{\theta}$. Unfortunately, due to the nature of CGMs there are three key issues with this naive approach, which we will briefly explain.

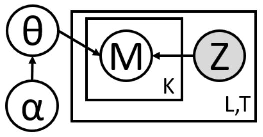
Issue 1: CGMs are under-specified, with $\boldsymbol{M}$ and often $\boldsymbol{\theta}$ being much larger than $\boldsymbol{Z}$. Because there are combinatorially many solutions for $\boldsymbol{M}$ and $\boldsymbol{\theta}$ that would explain $\boldsymbol{Z}$, there is inevitably a realization of both views that is highly correlated. Including PCCA in the likelihood objective will bias estimates of $\boldsymbol{M}$ or $\boldsymbol{\theta}$ to this maximally correlated solution. Since it is rarely the case that the true correlation is this high, this method ultimately harms the accuracy of $\boldsymbol{M}$.

Issue 2: Calculating the gradient of the CCA contribution to the likelihood for each iteration of NLBP is very time consuming. This is a significant bottleneck for any gradient-based solver.

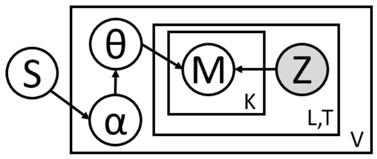
Issue 3: Since $\boldsymbol{\theta}$ takes values on the simplex, it has built-in negative correlations that adversely impact CCA. More specifically, when one value increases, all other values decrease because all the components of $\boldsymbol{\theta}$ have to sum to 1.

Issues 1 and 2 are demonstrated experimentally is section 4.4.1.1. Addressing these issues is non-trivial and requires care during parameter estimation.

4.3.3 A Multiview Approach Using a Hierarchical Bayesian Model



(a) Single View CGM.



(b) Multiview CGM.

Figure 4.2: Plate diagrams for single view and multiview CGM. K represents the neighborhood size of the graph, V the number of data views, L the number of locations and T the number of time steps.

We present a novel hierarchical Bayesian model for incorporating multiview information into a CGM that avoids the three issues above. Figure 4.2 shows the high-level difference between our multiview CGM and the single view CGM. Our method focuses on performing a joint estimation of $\boldsymbol{\theta}^1$ and $\boldsymbol{\theta}^2$ that estimates the shared information between the two views, and converts that information into the Dirichlet priors on $\boldsymbol{\theta}^1$ and $\boldsymbol{\theta}^2$.

Incorporating the shared information into the prior avoids Issue 1, as the prior does not force the algorithm to optimize for high correlation (which does not ensure high accuracy), but to account for correlations that arise during optimization. Issue 2 is avoided by decoupling the optimizations of PCCA and the CGMs for each view. Issue 3 is avoided by transforming $\boldsymbol{\theta}$ from the simplex to the real number plane.

For Issue 3, we use the additive log-ratio (alr) transform [Aitchison, 1986] to transform $\boldsymbol{\theta}^1$ and $\boldsymbol{\theta}^2$. We choose alr for a theoretical property used in Section 4.3.3.2, and because the centered log-ratio transform causes the covariance to be singular. The alr transform removes the kth component of $\log \boldsymbol{\theta}$ and divides the other components by its value. The choice of the index k is arbitrary, so long as it is consistent for all i and t. The transformed values, denoted as $\boldsymbol{\phi}$ are computed as $\phi^v_{(i,j,t)} = \log\left(\frac{\theta^v_{(i,j,t)}}{\theta^v_{(i,k,t)}}\right)$, $\forall j \neq k$.

$\boldsymbol{\phi}$ has one less dimension that $\boldsymbol{\theta}$, does not have constant sum, and does not have the spurious correlations intrinsic on the simplex. $\boldsymbol{\phi}$ can also be accurately fit by a normal distribution, which makes it ideal for use in PCCA.

Both PCCA and CCA operate on a set of n "instances" from different views, where the i-th instances from each view are assumed to be correlated. Each view is required to have the same number of instances, but the length of these instances (i.e., the number of variables in each view) can vary. In our formulation, we define a location/time pair as an instance for PCCA, which requires both views to have the same number of locations and time steps.

We estimate the latent variables $\boldsymbol{s}$ of PCCA, which encode the shared information between the two views. The conditional distributions $p(\boldsymbol{\phi}^v|\boldsymbol{s})$ are then use to compute the Dirichlet priors $\boldsymbol{\alpha}$, which allows the shared information to influence the estimation of $\boldsymbol{\theta}$. The details of these two steps are shown in the following sections.

4.3.3.1 Estimating $\boldsymbol{S}$

Using $\boldsymbol{\phi}^1$ and $\boldsymbol{\phi}^2$ as $\boldsymbol{x}^1$ and $\boldsymbol{x}^2$, we compute the PCCA parameters $\boldsymbol{W}$, $\boldsymbol{\Phi}$, and $\boldsymbol{\mu}$ as described in Equation 4.3. Let $\boldsymbol{S}$ be a matrix of samples of $\boldsymbol{s}$. From Equation 4.2, we can write the joint probability

$$P(\boldsymbol{S}_{(i,:,t)}, \boldsymbol{\phi}^1_{(i,:,t)}, \boldsymbol{\phi}^2_{(i,:,t)}) = P(\boldsymbol{S}_{(i,:,t)})P(\boldsymbol{\phi}^1_{(i,:,t)}|\boldsymbol{S}_{(i,:,t)})P(\boldsymbol{\phi}^2_{(i,:,t)}|\boldsymbol{S}_{(i,:,t)}) \tag{4.11}$$

Using it as a shorthand for $(i, :, t)$, this gives a joint log-likelihood of

$$L(\boldsymbol{S}_{it}, \boldsymbol{\phi}^1_{it}, \boldsymbol{\phi}^2_{it}) = -\frac{1}{2}\boldsymbol{S}_{it}\boldsymbol{S}'_{it} - \sum_{v=1}^{2}\frac{1}{2}(\boldsymbol{\phi}^v_{it} - \boldsymbol{S}_{it}\boldsymbol{W}^v - \boldsymbol{\mu}^v)(\boldsymbol{\Phi}^v)^{-1}(\boldsymbol{\phi}^v_{it} - \boldsymbol{S}_{it}\boldsymbol{W}^v - \boldsymbol{\mu}^v)' + \boldsymbol{C} \tag{4.12}$$

where $\boldsymbol{C}$ is a normalization constant. We can find the MLE of $\boldsymbol{S}_{it}$ by setting the gradient of the joint log-likelihood equal to 0.

$$\frac{\partial L(\boldsymbol{S}_{it}, \boldsymbol{\phi}^1_{it}, \boldsymbol{\phi}^2_{it})}{\partial \boldsymbol{S}_{it}} = -\boldsymbol{S}_{it} + \sum_{v=1}^{2}(\boldsymbol{\phi}^v_{it} - \boldsymbol{S}_{it}\boldsymbol{W}^v - \boldsymbol{\mu}^v)(\boldsymbol{\Phi}^v)^{-1}\boldsymbol{W}^{v\prime} = 0 \tag{4.13}$$

$$\boldsymbol{S}_{it} = \frac{\sum_{v=1}^{2}(\boldsymbol{\phi}^v_{it} - \boldsymbol{\mu}^v)(\boldsymbol{\Phi}^v)^{-1}\boldsymbol{W}^{v\prime}}{\boldsymbol{I} + \sum_{v=1}^{2}\boldsymbol{W}^v(\boldsymbol{\Phi}^v)^{-1}\boldsymbol{W}^{v\prime}} \tag{4.14}$$

Since the denominator is constant with respect to i, t, its inverse only needs to be computed once.

4.3.3.2 Estimating $\boldsymbol{\alpha}$

The conditional distributions $\boldsymbol{\phi}^v|\boldsymbol{S}$ represent the distributions of $\boldsymbol{\phi}^v$ when the shared information $\boldsymbol{S}$ is accounted for. We convert these distributions back to the simplex as Dirichlet distributions with parameters $\boldsymbol{\alpha}$, where they can be used as priors for $\boldsymbol{\theta}^v$.

The parameters $\boldsymbol{\alpha}$ of a Dirichlet can be alternatively defined by a mean $\tilde{\boldsymbol{\alpha}}$ and a precision $\rho = \sum_k \alpha_k$. The mean can easily be found by computing the inverse alr transform of $E(\boldsymbol{\phi}^v_{it}|\boldsymbol{S}_{it}) = \boldsymbol{S}_{it}\boldsymbol{W}^v + \boldsymbol{\mu}^v$. This gives a different mean for each choice of i

and t.

To estimate the precision, we can use a result from [Aitchison, 1986]. If $\boldsymbol{x} \sim Dir(\boldsymbol{\alpha})$, and $\boldsymbol{y} = \text{alr}(\boldsymbol{x})$, then $Cov(y_i, y_j) = \psi'(\alpha_k)$ and $Var(y_i) = \psi'(\alpha_i) + \psi'(\alpha_k)$, where $\psi'()$ is the trigamma function, and k is the variable used in the denominator of the alr transform. Note that the covariance is the same for every choice of i and j. We calculate the average of the off-diagonal values of $\boldsymbol{\Phi}^v$, denoted as c^v, and estimate $\alpha_k^v = [\psi']^{-1}(c^v)$, where $[\psi']^{-1}()$ is a numerically approximated inverse trigamma function. With α_k^v known, we can then compute $\alpha_j^v = [\psi']^{-1}(\Phi_{jj}^v - c^v)$, summing the values of α together to arrive at the precision ρ^v, which is shared among all choices of i and t. This gives us the prior parameters $\boldsymbol{\alpha}_{it}^v = \rho^v[\text{alr}]^{-1}(\boldsymbol{S}_{it}\boldsymbol{W}^v + \boldsymbol{\mu}^v)$.

4.3.3.3 Multiview CGM Algorithm

With the Dirichlet parameters $\boldsymbol{\alpha}_{it}^v$ known $\forall v, i, t$, we can use them as priors and update the values of $\boldsymbol{\theta}$ in a manner similar to section 4.3.1, the only difference being that each $\boldsymbol{\theta}_{it}^v$ has its own prior distribution defined by $\boldsymbol{\alpha}_{it}^v$.

The proceedure for our multiview CGM optimization is shown in Algorithm 6, with the details of MV_Update() in Algorithm 7. Note that, while we use non-linear belief propagation to solve for $\boldsymbol{M}$ in our single and multiview CGMs, any other iterative solver could be used in tandem with our multiview update.

Algorithm 6 Multiview CGM optimization

```
Input: $\boldsymbol{Z}^1$, $\boldsymbol{Z}^2$, initial $\boldsymbol{M}^1$, initial $\boldsymbol{M}^2$, initial $\boldsymbol{\theta}^1$, initial $\boldsymbol{\theta}^2$
while !converged do
    $\boldsymbol{M}^v = \text{NLBP}(\boldsymbol{Z}^v, \boldsymbol{\theta}^v, \boldsymbol{M}^v)\ \forall v$                //Compute $\boldsymbol{M}$ values given $\boldsymbol{\theta}$
    $[\boldsymbol{\theta}^1, \boldsymbol{\theta}^2] = \text{MV\_Update}(\boldsymbol{M}^1, \boldsymbol{M}^2)$        //Compute PCCA priors and estimate $\boldsymbol{\theta}$
end while
```

4.3.4 Parameter Sharing and Additional Views

Section 4.3.3 outlines the case when each location and time in the CGM is given a unique prior. We can implement parameter sharing by sharing these priors over groups of locations/times. In general, if we let $\boldsymbol{g}$ be a subdivision of the CGM over time and space, we can perform the previous derivation by first calculating $\boldsymbol{\theta}_{\boldsymbol{g}}^v = \frac{1}{|\boldsymbol{g}|}\sum_{i,t\in\boldsymbol{g}} \boldsymbol{\theta}_{it}^v$. $\boldsymbol{\phi}_{\boldsymbol{g}}^v$

Algorithm 7 MV_Update

```
Input: $\boldsymbol{M}^1$, $\boldsymbol{M}^2$
$\boldsymbol{\phi}^v_{(it)} = \text{alr}(\text{normalize}(\boldsymbol{M}^v_{it}))\ \forall v, i, t$
$[\boldsymbol{U}^1, \boldsymbol{U}^2, \boldsymbol{P}] = \text{CCA}(\boldsymbol{\phi}^1, \boldsymbol{\phi}^2)$
$\boldsymbol{\Sigma}^v = \text{Cov}(\boldsymbol{\phi}^v)\ \forall v$          //Start: Compute Probabilistic CCA parameters
$\boldsymbol{W}^v = \sqrt{\boldsymbol{P}}\boldsymbol{U}^v\boldsymbol{\Sigma}^v\ \forall v$
$\boldsymbol{\Phi}^v = \boldsymbol{\Sigma}^v - \boldsymbol{W}^{v\prime}\boldsymbol{W}^v\ \forall v$
$\boldsymbol{\mu}^v = \text{mean}(\boldsymbol{\phi}^v)\ \forall v$          // End: Compute Probabilistic CCA parameters
$\boldsymbol{S}_{it} = \frac{\sum_{v=1}^{2}(\boldsymbol{\phi}^v_{it}-\boldsymbol{\mu}^v)(\boldsymbol{\Phi}^v)^{-1}\boldsymbol{W}^{v\prime}}{\boldsymbol{I}+\sum_{v=1}^{2}\boldsymbol{W}^v(\boldsymbol{\Phi}^v)^{-1}\boldsymbol{W}^{v\prime}}\ \forall i, t$
$\tilde{\boldsymbol{\alpha}}^v_{it} = [\text{alr}]^{-1}(\boldsymbol{S}_{it}\boldsymbol{W}^v + \boldsymbol{\mu}^v)\ \forall v, i, t$          // Start: Compute prior parameters
$c^v = \text{mean}(\text{OffDiagonal}(\boldsymbol{\Phi}^v))\ \forall v$
$\rho^v = [\psi']^{-1}(c^v) + \sum_j [\psi']^{-1}(\Phi^v_{jj} - c^v)\ \forall v$
$\boldsymbol{\alpha}^v_{it} = \rho^v\tilde{\boldsymbol{\alpha}}^v_{it}\ \forall v, i, t$          // End: Compute prior parameters
$\boldsymbol{\theta}^v_{(it)} = \text{normalize}(\boldsymbol{\alpha}^v_{it} + \boldsymbol{M}^v_{it})\ \forall v, i, t$
```

is calculated from $\boldsymbol{\theta}^v_{\boldsymbol{g}}$, and the rest of the algorithm proceeds as before to give $\boldsymbol{\alpha}^v_{\boldsymbol{g}}$. The values of $\boldsymbol{\alpha}^v_{\boldsymbol{g}}$ can then be the shared priors over the set $\boldsymbol{g}$.

Section 4.3.3 assumes the number of views of the data to be 2, but it is trivial to extend this to the case $v > 2$ by utilizing generalized CCA methods, of which there are many options [Kettenring, 1971]. Our algorithm utilizes a shared set of canonical correlations $\boldsymbol{P}$, and canonical covariates $\boldsymbol{U}^v$ for each view. Calculating these using existing GCCA algorithms is all that is required to implement our multiview CGM for more than two views.

It is also important to note that while we define a PCCA "instance" as a location/time pair, instances could alternatively be defined over locations or times separately. This would allow either the number of time steps or locations to vary between views, which would permit the comparison of a wider variety of data views. The trade-off to this approach is that it reduces the total number of instances and increases the length of each instance (to absorb the additional dimension not used), which can reduce the effectiveness of PCCA.

4.4 Results and Discussion

Since there are no other known methods for this task, we compare our multiview CGM against the baseline of using a separate CGM for each view (denoted as the 2 CGM approach). All CGMs use the belief propagation optimization method described in Section 4.3.1. Algorithms are given population observations $\boldsymbol{Z}$ and tasked with finding the MAP estimates of the latent variables $\boldsymbol{M}$ while estimating parameters $\boldsymbol{\theta}$. Both approaches are evaluated based on their Normalized Absolute Error (NAE) for $\boldsymbol{M}$ on a suite of simulated and real world data sets. NAE, which has been used in past work to compare CGMs (e.g. [Iwata et al., 2017]), is computed as $\frac{\sum_{i \in \boldsymbol{L}} \sum_{j \in n(i)} \sum_{t=1}^{T} |M_{(i,j,t)} - M^*_{(i,j,t)}|}{\sum_{i \in \boldsymbol{L}} \sum_{j \in n(i)} \sum_{t=1}^{T} M^*_{(i,j,t)}}$ for each view, where $\boldsymbol{M}^*$ are the true transition population values and $\boldsymbol{M}$ are the predicted values.

Unless stated otherwise, the locations in each dataset are arranged as an $m \times m$ grid and the neighborhood of a location consists of the 8 adjacent grids, plus itself, with special cases for edge cells and corner cells. Our default CGM shares prior values over locations at each time step, while the multiview algorithm computes a unique prior at each time and location.

We first present the simulated data results, along with a description of our simulator. Second, we present the real world population flow data results. Finally, we show the runtimes for all performed experiments together in Section 4.4.3.[2] Code and data to reproduce all experiments are available at https://github.com/MVCGM/Code-and-Data.

4.4.1 Accuracy on Simulated Data

Our simulation algorithm generates a set of transition probabilities $\boldsymbol{\theta}$ for each view, along with an initial population distribution at time $t = 1$. $\boldsymbol{M}^v_{it}$ is drawn from a multinomial distribution with $\boldsymbol{p} = \boldsymbol{\theta}^v_{it}$ and $n = Z^v_{(i,t)}$. The populations of the next time step are then computed as $Z^v_{(i,t+1)} = \sum_j M^v_{(j,i,t)}$. This two step process is iterated until the last time step.

We use three different methods for generating $\boldsymbol{\theta}$. In the Attract-Same method, two locations are randomly chosen as centers of attraction, with the locations changing every m time steps. The value of $\theta_{(i,j,t)}$ is calculated using an RBF kernel on the distance of

[2]All experiments were run on a vitual machine with a 3.00GHz Intel Xeon CPU and 8.0 GB of RAM.

Table 4.1: Simulated data

	2 CGMs		Multiview CGM	
Generation	NAE View 1	NAE View 2	NAE View 1	NAE View 2
5x5, Attract-Same	0.355	0.351	**0.203***	**0.204***
5x5, Attract-Diff	0.351	0.369	**0.144***	**0.152***
5x5, Attract/Repel	0.377	0.342	**0.182***	**0.278***
10x10, Attract-Same	0.361	0.366	**0.330**	**0.332***
10x10, Attract-Diff	0.452	0.456	**0.202***	**0.202***
10x10, Attract/Repel	0.375	0.385	**0.159***	**0.298***

j from the closest center of attraction at time t. These values are normalized to sum to 1, and $\boldsymbol{\theta}^1 = \boldsymbol{\theta}^2$.

In the Attract-Diff method, a single center of attraction is generated for each view independently. In the Attract/Repel method, view 1 is attracted to the two centers of attraction, while view 2 is repelled from them, using the negative distance in the RBF kernel.

Table 4.1 shows the simulator results for the different generation methods when locations span a 5×5 and 10×10 grid with 100 time steps. Results are averaged over 10 random generations, $*$ values indicate statistical significance (paired t-test, $p \leq 0.05$). In each case, for each data view, our multiview algorithm achieves better NAE values, with 11 of the 12 views being significantly better. The smallest gains in NAE came when the two views had the same underlying transition probabilities. This is expected, as in the extreme case where both views are exactly the same, the multiview process adds no new information and should have similar performance to the baseline.

4.4.1.1 Comparison to Naive Approach

To illustrate the shortcomings of the naive multiview CGM approach discussed in section 4.3.2, we compared an implementation of it (Naive MVCGM) against the 2 CGM baseline and our hierarchical multiview algorithm (Multiview CGM). The three algorithms were compared on our simulated data, generated on a 5×5 grid with 50 time steps. Results are averaged over 10 randomly generated datasets.

Table 4.2 shows the NAE results for each algorithm on the simulated data. Table

Table 4.2: NAE Values

	2 CGMs		Naive MVCGM		Multiview CGM	
Generation	View 1	View 2	View 1	View 2	View 1	View 2
Attract-Same	0.362	0.358	0.855	0.870	**0.216***	**0.213***
Attract-Diff	0.165	0.219	0.862	0.848	**0.138**	**0.155***
Attract/Repel	0.361	0.334	0.867	0.807	**0.185***	**0.272***

4.3 also shows the runtimes of the algorithms, the correlation of the ground truth $\boldsymbol{M}$ values, and the correlation of the predicted $\boldsymbol{M}$ values for each algorithm. The naive approach is by far the worst of the three algorithms in terms of NAE and runtime, the latter due to the bottleneck of computing the PCCA gradient. While all three algorithms over estimate the correlation between the views, the naive method has the highest predicted correlation for each simulator except Attract-Same (which has the highest true correlation). In contrast, our hierarchical multiview algorithm predicts a lower correlation than the 2 CGM baseline in every case. This illustrates the point of Issue 1 from section 4.3.2, where maximizing for view correlation does not equate to high accuracy.

Table 4.3: Runtime and Correlation

		2 CGMs		Naive MVCGM		Multiview CGM	
Generation	True Corr	Corr	Runtime	Corr	Runtime	Corr	Runtime
Attract-Same	0.727	0.970	**3.45**	**0.932**	725.31	0.966	7.67
Attract-Diff	0.179	0.425	8.64	0.531	505.34	**0.367**	**8.53**
Attract/Repel	0.194	0.692	**4.77**	0.727	539.16	**0.599**	8.51

4.4.2 Accuracy on Real World Data

For our first set of real world data tests, we use anonymized people tracking data made publicly available by Nightley, Inc.[3] This data was collected from geo-tagged tweets

[3]SNS-based People Flow Data, Nightley, Inc., Shibasaki & Sekimoto Laboratory, the University of Tokyo, Micro Geo Data Forum, People Flow project, and Center for Spatial Information Science at the University of Tokyo. https://nightley.jp/archives/1954/

from thousands of participants in Tokyo, Osaka, and Nagoya, and creates movement trajectories for each individual. The original data contains no identifying information about individuals, other than gender. We further anonymize the data by aggregating the individual movements into the population variables $\boldsymbol{Z}$ and $\boldsymbol{M}$. We split the populations by men and women as our two views.

The data is processed as a 10×10 grid over locations, with time steps every 5 mins over a 24-hour period (288 total time steps). Each city has 6 days of data, which are learned in batch with the same $\boldsymbol{\theta}$ used for each day. Each day has a different population sample, thus a distinct $\boldsymbol{M}$ must be learned for each one. For this data, the priors of our multiview CGM are shared over locations for each time step.

Table 4.4 shows the results of learning the two views separately versus our multiview CGM. We also include results for fitting the data as a single population, combining aggregate values of men and women, with a single CGM. The results show that the aggregate movements of men and women in the areas are different enough that treating them as separate populations improves NAE for each view. These values are further improved by our multiview approach in 5 of the 6 cases. Statistical significance cannot be calculated on real world results, since there is only one dataset for each city.

Table 4.4: Japan population flow

	Single CGM	2 CGMs		Multiview CGM	
City	NAE	NAE Women	NAE Men	NAE Women	NAE Men
Tokyo	0.156	0.074	0.131	**0.056**	**0.071**
Osaka	0.052	0.037	0.050	**0.033**	**0.035**
Nagoya	0.064	0.041	**0.040**	**0.026**	0.058

Our final experiment uses data made publicly available by the U.S. Census Bureau.[4] We use aggregated values (over states) from their state-to-state job flow and migration data as the two views, with time steps every year for years 2005 to 2019. Note that ground truth at the state-to-state level is available for evaluation. Locations are taken as the 46 states without missing data during this period. This data set is unique from the others in that the two views are not over the same units (number of jobs vs number of people), and that the locations are fully connected in the span of one time step.

[4]https://www.census.gov/data.html

The fully connected locations makes the data too un-constrained to learn on its own. We add a universal transition prior to all algorithms, based on an RBF kernel over distances between state capitals. This biases the models to favor geographically close transitions, and enables the transitions to be learned with a reasonable amount of accuracy.

Table 4.5 shows the NAE results on this data. Even with the views being over different entities, our multiview approach offers a considerable improvement for both views.

Table 4.5: US job and population flow

2 CGMs		Multiview CGM	
NAE Jobs	NAE Pop	NAE Jobs	NAE Pop
0.404	0.454	**0.315**	**0.348**

4.4.3 Runtime Results

Table 4.6 shows the runtime results from all performed experiments. We include the dimensions of $\boldsymbol{M}$ for each dataset, listed as (number of locations)$\times$(neighborhood)$\times$(time steps). The table is sorted by the overall size of $\boldsymbol{M}$. Our multiview CGM adds increased runtime per iteration over the baseline. However, we notice that this overhead is mitigated by our algorithm converging in fewer iterations. The runtime results indicate that the additional overhead is a significant factor when the problem size is small. As the size of $\boldsymbol{M}$ increases, the faster convergence rate becomes a greater factor in reducing runtime, with multiview CGM being significantly faster on our largest datasets.

4.5 Conclusion

In this chapter we presented a novel hierarchical Bayesian approach to performing multiview inference in Collective Graphical Models. This is the first ever attempt at multiview learning in CGMs, as the under-specified nature of the models prevents a traditional multiview approach from significantly impacting optimization. Our method overcomes this issue by estimating the shared information between views as a transition prior, which

Table 4.6: Runtime results in seconds, sorted by the size of the spatial grid.

Data Set	$\boldsymbol{M}$ Size	$\lvert\boldsymbol{M}\rvert$	2 CGMs	Multiview CGM	Runtime Ratio
5 x 5, Attract-Same	25x9x100	22,500	**7.56**	15.98	2.11
5 x 5, Attract-Diff	25x9x100	22,500	**9.81**	17.31	1.76
5 x 5, Attract/Repel	25x9x100	22,500	**9.59**	17.80	1.86
US Job and People Flow	46x46x15	31,740	**31.79**	46.24	1.45
10 x 10, Attract-Same	100x9x100	90,000	**44.57**	53.98	1.21
10 x 10, Attract-Diff	100x9x100	90,000	**30.73**	78.55	2.56
10 x 10, Attract/Repel	100x9x100	90,000	**73.85**	146.82	1.96
Tokyo	600x9x288	1,555,200	3757.0	**1378.9**	0.37
Osaka	600x9x288	1,555,200	2196.5	**1581.7**	0.72
Nagoya	600x9x288	1,555,200	2242.3	**1412.7**	0.63

allows the model to account for correlations that arise between views rather than arbitrarily increase them during optimization. Our method is evaluated on simulated and real world population movement data, where it consistently and significantly outperforms the baseline method, reducing the error rate in both views.

Chapter 5: General Conclusions

In this thesis, I detailed work done in two main research avenues for spatial and spatiotemporal data: a quantile based spatial scan statistic, and a multiview approach for collective graphical models.

The quantile based scan statistic research produced two new scan algorithms, QSSS and QSnap, for comparing regions in space or across different snapshots in time using quantile regression. Comparing the regions based on their fitted regression quantiles allows the scan statistic to be more robust to data outliers, and helps the user identify changes to specific parts of the data distribution. Both of these points were illustrated in multiple experiments on real world and synthetic data. While the benefits of quantile based statistics usually come with a prohibitively larger cost in runtime, incremental optimization techniques were developed for both QSSS and QSnap that reduce their runtime by an order of magnitude. These incremental optimizations allow the algorithms to run in time comparable to traditional mean-based scans, giving them the best combination of speed, robustness, and statistical power. An adjustment for performing one-sided tests was also introduced, which was optimized to incur a negligible penalty on runtime. This one sided test is applicable for both QSSS and QSnap.

My research into collective graphical models produced the first ever multiview algorithm for CGMs. CGMs are usually under specified in practice, with more variables to determine than data observations. I showed that a traditional multiview approach, incorporating a shared likelihood term between the two model views, is poorly defined due to this lack of observations, strongly influencing the model to a solution that harms its accuracy. Instead, I developed a hierarchical Bayesian multiview model, where the correlations between views are incorporated into parameter priors, softening their influence on optimization. This Bayesian approach demonstrated greatly improved accuracy compared to learning each view separately, on both real world and simulated data. The collection of results also indicate that, while the multiview algorithm has a higher runtime for small data sets, it becomes more comparable to the baseline as the problem size grows, even becoming faster to run on the largest experiments.

While they focused on different algorithmic methods and applications, both research tracks revolve around the shared goals of improving spatiotemporal data analysis in the face of data uncertainty. For the scan statistic work, this was accomplished by utilizing the robustness of quantile test statistics. These quantile scan statistics not only reduce the influence of outliers, they also greatly reduce the influence of any data noise not apparent in the quantile of interest. This, combined with the one-sided test variant, opens up a wide level of customization for performing scan statistic region detection in a dataset. By exploiting domain knowledge to pinpoint quantiles of interest and likely variable interactions, the power of these tests and their ability to overcome data noise is greatly increased. Given how much domain knowledge can be leveraged in the parameters of these tests, I believe their full potential has yet to be fully explored.

My multiview CGM research helps to address the issue of missing data that is inherent in most CGM applications. Previously, parameter sharing has been the primary method for dealing with this missing data, trading generality in order to reduce the number of learned parameters. Multiview learning is a natural approach to this as well, as the additional information from multiple data views, if incorporated correctly, can help account for a lack of observed information. The Bayesian multiview method presented is able to accurately learn multiple CGMs at the same time, and has potential to be useful in the single CGM setting as well. Since the method works by transforming transition correlations into distribution priors, it can also be applied if one of the CGM views is fully observed, or if a view consists of non-population based observations. For example, if the primary view of interest was a CGM for migrating bird populations, the other views could be measures of change in different habitat variables (greenness, land cover, etc) between locations and times. Whether incorporating this information as a multiview prior is more informative than using it as a covariate in the transition distribution is a matter of future research.

Bibliography

A. Abrams, K. Kleinman, and M. Kulldorff. Gumbel based p-value approximations for spatial scan statistics. *International Journal of Health Geographics*, 9(1):61, 2010.

J. Aitchison. *The Statistical Analysis of Compositional Data*, volume Monographs on Statistics and Applied Probability. Chapman & Hall Ltd., London, 1986. (Reprinted in 2003 with additional material by The Blackburn Press).

S. Akaho. A kernel method for canonical correlation analysis. In *International Meeting on Psychometric Society (IMPS2001)*, 2001.

I. M. Almanjahie, Z. C. Elmezouar, B. A. Bachir, and Z. Kaid. Spatial local linear estimation of the l 1-conditional quantiles for functional regressors. *Communications in Statistics-Theory and Methods*, pages 1–20, 2019.

G. Andrew, R. Arora, J. Bilmes, and K. Livescu. Deep canonical correlation analysis. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1247–1255. PMLR, 17–19 Jun 2013.

Francis R. Bach and Michael I. Jordan. A probabilistic interpretation of canonical correlation analysis. Technical Report 688, Department of Statistics, University of California, Berkeley, 2005.

Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society, Series B*, 57(1):289–300, 1995.

G. Bernstein, R. McKenna, T. Sun, D. Sheldon, M. Hay, and G. Miklau. Differentially private learning of undirected graphical models using collective graphical models. In *Proceedings of the 34th International Conference on Machine Learning*, pages 478–487, 2017.

N. Best, S. Richardson, and A. Thomson. A comparison of Bayesian spatial models for disease mapping. *Statistical Methods in Medical Research*, 14(1):35–59, 2005.

A. Blum and T. Mitchell. Combining labeled and unlabeled data with cotraining. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory.*, page 92–100, 1998.

C. E. Bonferroni. Teoria statistica delle classi e calcolo delle probabilità. *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commerciali di Firenze*, 8:3–62, 1936.

Andria Caruthers. Mapping poverty in the appalachian region. `https://www.communitycommons.org/2016/08/mapping-poverty-in-the-appalachian-region/`, 2016.

Lydia DePillis. What america can learn from cities with super-low unemployment. `http://money.cnn.com/2018/01/12/news/economy/cities-unemployment/index.html`, 2018.

L. Duczmal and R. Assuncao. A simulated annealing strategy for the detection of arbitrarily shaped spatial clusters. *Comput Stat Data Anal*, 45:269–286, 2004.

Fangxiang Feng, Xiaojie Wang, and Ruifan Li. Cross-modal retrieval with correspondence autoencoder. In *Proceedings of the 22nd ACM International Conference on Multimedia*, page 7–16, New York, NY, USA, 2014. Association for Computing Machinery. doi: 10.1145/2647868.2654902.

RL Fox and MP Kapoor. Rates of change of eigenvalues and eigenvectors. *AIAA journal*, 6(12):2426–2429, 1968.

Andrea Frome, Greg S. Corrado, Jonathon Shlens, Samy Bengio, Jeffrey Dean, Marc'Aurelio Ranzato, and Tomas Mikolov. Devise: A deep visual-semantic embedding model. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, page 2121–2129, Red Hook, NY, USA, 2013. Curran Associates Inc.

Gene Golub and Charles Van Loan. *Matrix computations*, volume 3. JHU Press, 2012.

C. Gutenbrunner and J. Jurecková. Regression rank scores and regression quantiles. *The Annals of Statistics*, pages 305–330, 1992.

C. Gutenbrunner, J. K. R. S. Jurečková, R. Koenker, and S. Portnoy. Tests of linear hypotheses based on regression rank scores. *Journaltitle of Nonparametric Statistics*, 2(4):307–331, 1993.

S. Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.

H. Hotelling. Relations between two sets of variates. *Biometrika*, 28:312–377, 1936.

J. Huntington, K. Hegewisch, B. Daudert, C. Morton, J. Abatzoglou, D. McEvoy, and T. Erickson. Climate engine: Cloud computing of climate and remote sensing data for advanced natural resource monitoring and process understanding. *Bulletin of the American Meteorological Society*, 2017. `http://journals.ametsoc.org/doi/abs/10.1175/BAMS-D-15-00324.1`.

T. Iwata, H. Shimizu, F. Naya, and N. Ueda. Estimating people flow from spatiotemporal population data via collective graphical mixture models. *ACM Trans. Spatial Algorithms Syst.*, 3(1):1–18, 2017. doi: 10.1145/3080555.

Tomoharu Iwata and Hitoshi Shimizu. Neural collective graphical models for estimating spatio-temporal population flow from aggregated data. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):3935–3942, 2019. doi: 10.1609/aaai.v33i01.33013935.

David Kahle and Hadley Wickham. ggmap: Spatial visualization with ggplot2. *The R Journal*, 5(1):144–161, 2013. URL `http://journal.r-project.org/archive/2013-1/kahle-wickham.pdf`.

Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 39(4):664–676, April 2017. doi: 10.1109/TPAMI.2016.2598339.

S. Kelling, A. Johnston, W. M. Hochachka, M. Iliff, D. Fink, J. Gerbracht, C. Lagoze, F. A. La Sorte, T. Moore, A. Wiggins, W-K. Wong, C. Wood, and J. Yu. Can observation skills of citizen scientists be estimated using species accumulation curves? *PLoS ONE*, 10(10):e0139600, 2015. doi: 10.1371/journal.pone.0139600.

J.R. Kettenring. Canonical analysis of several sets of variables. In *Biometrika*, volume 58, page 433–451, 1971.

C. King and J. J. Song. Bayesian spatial quantile regression for areal count data, with application on substitute care placements in texas. *Journal of Applied Statistics*, 46 (4):580–597, 2019.

R. Koenker and J. A. Machado. Goodness of fit and related inference processes for quantile regression. *Journal of the American Statistical Association*, 94(448):1296–1310, 1999.

Roger Koenker and Gilbert Bassett. Regression quantiles. *Econometrica*, 46(1):33–50, 1978.

Martin Kulldorff. A spatial scan statistic. *Communications in Statistics: Theory and Methods*, 26(6):1481–1496, 1997.

F. A. La Sorte and D. Fink. Migration distance, ecological barriers and en-route variation in the migratory behaviour of terrestrial bird populations. *Global Ecology and Biogeography*, 26:216–227, 2017.

P. L. Lai and C. Fyfe. Kernel and nonlinear canonical correlation analysis. *International Journal of Neural Systems*, 10, 2000.

L. Lan, V. Malbasa, and S. Vucetic. Spatial scan for disease mapping on a mobile population. In *Proceedings of the 28th AAAI Conference*, pages 431–437, 2014.

D. Li, N. Dimitrova, M. Li, and I. K. Sethi. Multimedia content processing through cross-modal association. In *Proceedings of the Eleventh ACM International Conference on Multimedia*, page 604–611, New York, NY, USA, 2003. Association for Computing Machinery. doi: 10.1145/957013.957143.

Y. Li, M. Yang, and Z. Zhang. A survey of multi-view representation learning. *IEEE Transactions on Knowledge and Data Engineering*, 31(10):1863–1883, 2019. doi: 10.1109/TKDE.2018.2872063.

L-P. Liu, D. Sheldon, and T. G. Dietterich. Gaussian approximation of collective graphical models. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32, pages 1602–1610, 2014.

Jose Machado and JMC Santos Silva. Quantiles for counts. *Journal of the American Statistical Association*, 100(472):1226–1237, 2002.

D. P. Macmillan. *Quantile Regression for Spatial Data*. Springer-Verlag, Berlin, 2013.

E. McFowland, III, S. Somanchi, and D. B. Neill. Efficient Discovery of Heterogeneous Treatment Effects in Randomized Experiments via Anomalous Pattern Detection. *ArXiv e-prints*, March 2018.

Thomas Minka. Estimating a dirichlet distribution, 2000.

A. Mood. *Introduction to the Theory of Statistics*. McGraw Hill Book Co., 1950.

T. Moore and W-K. Wong. Discovering hotspots and coldspots of species richness in ebird data. In *Proceedings of the AAAI-15 Workshop on Computational Sustainability*, 2015.

T. Moore and W-K. Wong. An efficient quantile spatial scan statistic for finding unusual regions in continuous spatial data with covariates. In *Uncertainty in Artificial Intelligence*, pages 756–765, Corvallis, OR, 2018. AUAI Press.

Travis Moore and Weng-Keen Wong. The quantile snapshot scan: Comparing quantiles of spatial data from two snapshots in time. In *International Conference on Artificial Intelligence and Statistics*, pages 2677–2686. PMLR, 2020.

D. B. Neill. Fast subset scan for spatial pattern detection. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74(2):337–360, 2012.

D.B. Neill and A. W. Moore. Rapid detection of significant spatial clusters. In *Proceedings of the 10th ACM SIGKDD Conference*, pages 256–265, 2004.

T. Nguyen, A. Kumar, H. C. Lau, and D. Sheldon. Approximate inference using dc programming for collective graphical models. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 51 of *Proceedings of Machine Learning Research*, pages 685–693. PMLR, 2016.

Timothy Parker. Download data. United States Department of Agriculture, 2017. `https://www.ers.usda.gov/data-products/county-level-data-sets/county-level-data-sets-download-data/`.

KL Prudic, KP McFarland, JC Oliver, RA Hutchinson, EC Long, JT Kerr, and M Larrivée. ebutterfly: leveraging massive online citizen science for butterfly conservation, 2017. `http://www.mdpi.com/2075-4450/8/2/53/htm`.

R. J. Radke, S. Andra, O. Al-Kofahi, and B. Roysam. Image change detection algorithms: a systematic survey. *IEEE Transactions on Image Processing*, 14:294–307, 2005.

C. Rao. Large sample tests of statistical hypotheses concerning several parameters with applications to problems of estimation. *Mathematical Proceedings of the Cambridge Philosophical Society*, 44:50–57, 1948.

Brian J. Reich, Montserrat Fuentes, and David B. Dunson. Bayesian spatial quantile regression. *Journal of the American Statistical Association*, 106(493):6–20, 2011.

Peter J. Rousseeuw and Annick M. Leroy. *Robust Regression and Outlier Detection.* John Wiley and Sons, 1987.

D. Sheldon, M. A. S. Elmohamed, and D. Kozen. Collective inference on markov models for modeling bird migration. In *Advances in Neural Information Processing Systems.*, page 1321–1328, 2007.

D. Sheldon, T. Sun, A. Kumar, and T. G. Dietterich. Approximate inference in collective graphical models. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1004–1012, 2013.

D. R. Sheldon and T. G. Dietterich. Collective graphical models. In *Advances in Neural Information Processing Systems*, page 1161–1169, 2011.

J. Sherman and W. J. Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, 21 (1):124–127, 1950.

M. J. Silvapulle and P. Silvapulle. A score test against one-sided alternatives. *Journal of the American Statistical Association*, 90(429):342–349, 1995.

B. L. Sullivan, J. L Aycrigg, J. H. Barry, R. E. Bonney, N. Bruns, C. B. Cooper, T. Damoulas, A. A. Dhondt, T. Dietterich, A. Farnsworth, D. Fink, J. W. Fitzpatrick, T. Fredericks, J. Gerbracht, C. Gomes, W. M. Hochachka, M. J. Iliff, C. Lagoze, F. La Sorte, M. Merrifield, W. Morris, T. B. Phillips, M. Reynolds, A. D. Rodewald, K. V. Rosenberg, N. M. Trautmann, A. Wiggins, D. W. Winkler, W.-K. Wong, C. L. Wood, J. Yu, and S. Kelling. The ebird enterprise: An integrated approach to development and application of citizen science. *Biological Conservation*, 169:31–40, 2014.

S. Sun. A survey of multi-view machine learning. *Neural Computing and Applications*, 23(7-8):2031–2038, 2013.

T. Sun, D. Sheldon, and A. Kumar. Message passing for collective graphical models. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 853–861, 2015.

Abraham Wald. Tests of statistical hypotheses concerning several parameters when the number of observations is large. *Transactions of the American Mathematical society*, 54(3):426–482, 1943.

Greg Welch, Gary Bishop, et al. An introduction to the kalman filter. 1995.

Samuel S Wilks. Certain generalizations in the analysis of variance. *Biometrika*, pages 471–494, 1932.

D. Witten, R. Tibshirani, and T. Hastie. A penalized matrix decomposition, with applications to sparse principal components and canonical correlation analysis. *Biostatistics*, 10(3):515–534, 2009.

Chang Xu, Dacheng Tao, and Chao Xu. A survey on multi-view learning. *CoRR*, abs/1304.5634, 2013. URL `http://arxiv.org/abs/1304.5634`.

Y. Yang and H. Wang. Multi-view clustering: A survey. *Big Data Mining and Statistics*, 1(2):83–107, 2018.

Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. Generalized belief propagation. In *Advances in Neural Information Processing Systems (NIPS)*, volume 13, page 689–695, 2000.

Jonathan S. Yedidia, William T. Freeman, and Yair Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312, 2005.

Zhe Zhu. Change detection using landsat time series: A review of frequencies, preprocessing, algorithms, and applications. *ISPRS Journal of Photogrammetry and Remote Sensing*, 130:370–384, 2017.

APPENDICES

Appendix A: Partial AUC Calculation

For our simulation experiments we evaluate each algorithm by computing the partial AUC of the TPR vs FPR graph, over the FPR range [0,0.2]. For each of the 30 datasets in each experiment setup, each algorithm reports the best region discovered, which we denote as $\boldsymbol{C}^*$. Any points in $\boldsymbol{C}^*$ that are generated from the shifted distribution are true positives (TP), while all other points in $\boldsymbol{C}^*$ are false positives (FP). Points outside $\boldsymbol{C}^*$ generated by the shifted distribution are false negatives (FN), while the other data points outside C^* are true negatives (TN).

For each dataset, we compute the TPR and FPR of the best region from each algorithm as $tpr = \frac{TP}{TP+FN}$ and $fpr = \frac{FP}{FP+TN}$. In our setup we must either accept the entire region or none of it. This means that TPR $= 0$ when FPR $< fpr$, and TPR $= tpr$ when FPR $\geq fpr$. This produces a step graph for each algorithm on each dataset. We calculate the partial AUC as the area under each graph in the FPR range [0,0.2]. Note that if $fpr > 0.2$, then the partial AUC is 0. We report the average partial AUC over all 30 datasets for each algorithm.

In our synthetic experiments, the highest possible partial AUC score is 0.2, if TPR = 1. This value is extremely unlikely, since our true positive points are not perfectly grouped together. Any region that overlaps all true positive points will almost certainly overlap negative points as well.

Appendix B: PCCA Likelihood Gradient

An intuitive method for implementing a multiview CGM framework that accounts for view correlation is to incorporate the probabilistic CCA model into the CGM likelihood. The latent variables of PCCA would be shared by the CGM parameters $\boldsymbol{\theta}^1$ and $\boldsymbol{\theta}^2$, as these are the only CGM values not skewed by the population size. This utilizes the CCA likelihood as an additional term in the objective to shape the maximizing values of $\boldsymbol{M}^1$ and $\boldsymbol{M}^2$. This multiview CGM can still be solved using the NLBP algorithm, as NLBP utilizes the gradient of the log-likelihood with respect to $\boldsymbol{M}$. Thus, this multiview implementation can be realized by calculating the gradient of the PCCA additions to the model. Specifically, $P(\boldsymbol{\theta}^1, \boldsymbol{\theta}^2|\boldsymbol{S})P(\boldsymbol{S})$.

We start with a reparameterization. Our CCA model works on the additive log transform of the transition probabilities $\boldsymbol{\theta}$. This moves the values into the real space, which CCA is more suitably used for, by dividing by an arbitrary index D in the vector. Let $\boldsymbol{\phi} = \log\left(\frac{\boldsymbol{\theta}_{-D}}{\boldsymbol{\theta}_D}\right)$. $P(\boldsymbol{\theta}|\boldsymbol{S})$ follows a logistic normal distribution, which has the form

$$P(\boldsymbol{\theta}^v|\boldsymbol{S}) = \frac{1}{(2\pi^k|\boldsymbol{\Phi}^v|)^{1/2}} \frac{1}{\prod_{j=1}^k \theta_j^v} \exp\left(-\frac{1}{2}(\boldsymbol{\phi^v} - \boldsymbol{\mu}^v)'\boldsymbol{\Phi}^v(\boldsymbol{\phi^v} - \boldsymbol{\mu}^v)\right) \tag{B.1}$$

In order to fully describe the model in terms of the transtition estimates, we will rewrite $\boldsymbol{\theta}$ in terms of $\boldsymbol{M}$, $\theta_{(i,j,t)} = \frac{M_{(i,j,t)}}{\sum_f M_{(i,f,t)}}$. This gives the equality $\boldsymbol{\phi}_{it} = \log\left(\frac{\boldsymbol{M}_{it/D}}{M_{(i,D,t)}}\right)$, which we will use for shorthand.

Using the logistic normal distribution for $P(\boldsymbol{\theta}|\boldsymbol{S})$, a standard normal for $\boldsymbol{S}$, and the parameters associated with probabilistic CCA, our full log likelihood can be written as

$$L(\boldsymbol{S}) + L(\boldsymbol{\theta}^1|\boldsymbol{S}) + L(\boldsymbol{\theta}^2|\boldsymbol{S}) = \tag{B.2}$$

$$\sum_t \sum_i -\frac{1}{2}\log(2\pi^k) - \frac{1}{2}\boldsymbol{S'}_{it}\boldsymbol{S}_{it} + \sum_{v=1,2}\sum_t \sum_i -\frac{1}{2}\log(2\pi^k|\boldsymbol{\Phi}^v|) - \sum_j \log\left(\frac{M^v_{(i,j,t)}}{\sum_f M^v_{(i,f,t)}}\right) \tag{B.3}$$

$$-\frac{1}{2}\left(\log\left(\frac{\boldsymbol{M}^v_{it/D}}{M^v_{(i,D,t)}}\right) - \boldsymbol{S}_{it}\boldsymbol{W}^v - \boldsymbol{\mu}^v\right)(\boldsymbol{\Phi^v})^{-1}\left(\log\left(\frac{\boldsymbol{M}^v_{it/D}}{M_{(i,D,t)}}\right) - \boldsymbol{S}_{it}\boldsymbol{W}^v - \boldsymbol{\mu}^v\right)' \tag{B.4}$$

$$\tag{B.5}$$

Now, we can start to take the derivative of the likelihood with respect to $M^v_{(i,j,t)}$. We will use the shorthand $ijt = (i,j,t)$ for the index of the derivative argument, and $\boldsymbol{m}^1_{it} = \left(\log\left(\frac{\boldsymbol{M}^1_{it/D}}{M^1_{(i,D,t)}}\right) - \boldsymbol{S}_{it}\boldsymbol{W}^1 - \boldsymbol{\mu}^1\right)$. Note that the CCA parameters $\boldsymbol{S}$, $\boldsymbol{W}$, $\boldsymbol{\Phi}$, and $\boldsymbol{\mu}$ all depend on $\boldsymbol{\phi}$, and thus depend on $\boldsymbol{M}$.

$$\frac{\partial(L(\boldsymbol{S}) + L(\boldsymbol{\theta}^1|\boldsymbol{S}) + L(\boldsymbol{\theta}^2|\boldsymbol{S}))}{\partial M^1_{ijt}} = \tag{B.6}$$

$$-\sum_f \boldsymbol{S}'_{(i,f,t)}\frac{\partial \boldsymbol{S}_{(i,f,t)}}{\partial M^1_{ijt}} - \frac{n}{2}tr\left((\boldsymbol{\Phi}^1)^{-1}\frac{\partial \boldsymbol{\Phi}^1}{M^1_{ijt}}\right) - \frac{n}{2}tr\left((\boldsymbol{\Phi}^2)^{-1}\frac{\partial \boldsymbol{\Phi}^2}{M^1_{ijt}}\right) - \frac{1}{M^1_{ijt}} + \frac{k}{\sum_f M^1_{(i,f,t)}} \tag{B.7}$$

$$-\boldsymbol{m}^1_{it}(\boldsymbol{\Phi^1})^{-1}\left(\boldsymbol{I}^1_{ijt} - \frac{\partial S_{it}}{\partial M^1_{ijt}}\boldsymbol{W}^1 - \boldsymbol{S}_{it}\frac{\partial \boldsymbol{W}^1}{\partial M^1_{ijt}} - \frac{\partial \boldsymbol{\mu}^1}{\partial M^1_{ijt}}\right)' + \frac{1}{2}\boldsymbol{m}^1_{it}(\boldsymbol{\Phi^1})^{-1}\frac{\partial \boldsymbol{\Phi}^1}{M^1_{ijt}}(\boldsymbol{\Phi^1})^{-1}\boldsymbol{m}^{1\prime}_{it} \tag{B.8}$$

$$-\boldsymbol{m}^2_{it}(\boldsymbol{\Phi^2})^{-1}\left(-\frac{\partial \boldsymbol{S}_{it}}{\partial M^1_{ijt}}\boldsymbol{W}^2 - \boldsymbol{S}_{it}\frac{\partial \boldsymbol{W}^2}{\partial M^1_{ijk}}\right)' + \frac{1}{2}\boldsymbol{m}^2_{it}(\boldsymbol{\Phi^2})^{-1}\frac{\partial \boldsymbol{\Phi}^2}{M^1_{ijt}}(\boldsymbol{\Phi^2})^{-1}\boldsymbol{m}^{2\prime}_{it} \tag{B.9}$$

The vector $\boldsymbol{I}^v_{ijt}$ represents the derivative $\frac{\partial}{\partial M^v_{ijt}}\log\left(\frac{\boldsymbol{M}^v_{it/D}}{M^v_{(i,D,t)}}\right)$. If $j \neq D$, $\boldsymbol{I}^v_{ijt}$ is a vector of zeros with the jth element equal to $\frac{1}{M^v_{ijt}}$. If $j = D$ then every element of $\boldsymbol{I}^v_{ijt}$ is equal to $\frac{-1}{M^v_{ijt}}$.

What remains to fill in are the partial derivatives of the CCA parameters with respect to M^v_{ijt}. Those are derived in the following sections.

B.1 CCA Parameter Derivatives

Bach and Jordan derived the closed form solution for the CCA parameters that maximize the joint likelihood. We will use those forms to find their derivatives with respect to M^v_{ijt}.

$\boldsymbol{W}$**:** This variable depends on the covariance matrix of $\boldsymbol{\phi}^v$, denoted as $\boldsymbol{\Sigma}^v$, the canonical vector matrix $\boldsymbol{U}^v$, and the diagonal matrix of correlation coefficients $\boldsymbol{P}$.

$$\boldsymbol{W}^v = \sqrt{\boldsymbol{P}}\boldsymbol{U}^{v\prime}\boldsymbol{\Sigma}^v \tag{B.10}$$

$$\frac{\partial \boldsymbol{W}^v}{\partial M^v_{ijt}} = \sqrt{\boldsymbol{P}}\boldsymbol{U}^{v\prime}\frac{\partial \boldsymbol{\Sigma}^v}{\partial M^v_{ijt}} + \sqrt{\boldsymbol{P}}\frac{\partial \boldsymbol{U}^v}{\partial M^v_{ijt}}'\boldsymbol{\Sigma}^v + \frac{\partial \sqrt{\boldsymbol{P}}}{\partial M^v_{ijt}}\boldsymbol{U}^{v\prime}\boldsymbol{\Sigma}^v \tag{B.11}$$

$$= \sqrt{\boldsymbol{P}}\boldsymbol{U}^{v\prime}\frac{\partial \boldsymbol{\Sigma}^v}{\partial M^v_{ijt}} + \sqrt{\boldsymbol{P}}\frac{\partial \boldsymbol{U}^v}{\partial M^v_{ijt}}'\boldsymbol{\Sigma}^v + \frac{1}{2}diag\left((\boldsymbol{p})^{-1/2}\frac{\partial \boldsymbol{p}}{\partial M^v_{ijt}}\right)\boldsymbol{U}^{v\prime}\boldsymbol{\Sigma}^v \tag{B.12}$$

$$\frac{\partial \boldsymbol{W}^f}{\partial M^v_{ijt}} = \sqrt{\boldsymbol{P}}\frac{\partial \boldsymbol{U}^f}{\partial M^v_{ijt}}'\boldsymbol{\Sigma}^f + \frac{1}{2}diag\left((\boldsymbol{p})^{-1/2}\frac{\partial \boldsymbol{p}}{\partial M^v_{ijt}}\right)\boldsymbol{U}^{f\prime}\boldsymbol{\Sigma}^f \tag{B.13}$$

$$\frac{\partial \boldsymbol{\Sigma}^v}{\partial M^v_{ijt}} = \frac{\partial}{\partial M^v_{ijt}}\frac{1}{n^v}(\boldsymbol{\phi}^v - \boldsymbol{\mu}^v)'(\boldsymbol{\phi}^v - \boldsymbol{\mu}^v) \tag{B.14}$$

$$= \frac{1}{n^v}\frac{\partial(\boldsymbol{\phi}^v_{it} - \boldsymbol{\mu}^v)'}{\partial M^v_{ijt}}(\boldsymbol{\phi}^v - \boldsymbol{\mu}^v) + \frac{1}{n^v}(\boldsymbol{\phi}^v - \boldsymbol{\mu}^v)'\frac{\partial(\boldsymbol{\phi}^v_{it} - \boldsymbol{\mu}^v)}{\partial M^v_{ijt}} \tag{B.15}$$

$$\frac{\partial \boldsymbol{\phi}^v_{it}}{\partial M^v_{ijt}} = \frac{\partial}{\partial M^v_{ijt}}\log\left(\frac{\boldsymbol{M}^v_{it/D}}{M^v_{(i,D,t)}}\right) = \boldsymbol{I}^v_{ijt} \tag{B.16}$$

$$\tag{B.17}$$

$\boldsymbol{U}^1$ and $\boldsymbol{P}$ are found using the eigenvalue problem solutions of the matrix $(\Sigma^{11})^{-1}\Sigma^{12}(\Sigma^{22})^{-1}\Sigma^{21}$, with a symmetric matrix for the second viewpoint. $\boldsymbol{U}^v$ is a matrix of all the eigenvectors, while $\boldsymbol{P}$ is a diagonal matrix of the square root of the corresponding eigenvalues. Finding the derivative of these eigenvectors and eigenvalues is detailed in later subsections of this appendix.

$\boldsymbol{\Phi}$:

$$\boldsymbol{\Phi}^v = \boldsymbol{\Sigma}^v - \boldsymbol{W}^{v\prime}\boldsymbol{W}^v \tag{B.18}$$

$$\frac{\partial \boldsymbol{\Phi}^v}{\partial M^v_{ijt}} = \frac{\partial \boldsymbol{\Sigma}^v}{\partial M^v_{ijt}} - \frac{\partial \boldsymbol{W}^v}{\partial M^v_{ijt}}^{\prime}\boldsymbol{W}^v - \boldsymbol{W}^{v\prime}\frac{\partial \boldsymbol{W}^v}{\partial M^v_{ijt}} \tag{B.19}$$

$$\frac{\partial \boldsymbol{\Phi}^f}{\partial M^v_{ijt}} = -\frac{\partial \boldsymbol{W}^f}{\partial M^v_{ijt}}^{\prime}\boldsymbol{W}^f - \boldsymbol{W}^{f\prime}\frac{\partial \boldsymbol{W}^f}{\partial M^v_{ijt}} \tag{B.20}$$

We can substitute in our derivatives of $\boldsymbol{W}^v$ and $\boldsymbol{\Sigma}^v$ from the previous section.

$\boldsymbol{\mu}$:

$$\boldsymbol{\mu}^v = \frac{1}{n^v}\sum_{it}\boldsymbol{\phi}^v_{it} \tag{B.21}$$

$$\frac{\partial \boldsymbol{\mu}^v}{\partial M^v_{ijt}} = \frac{1}{n^v}\boldsymbol{I}^v_{ijt} \tag{B.22}$$

$\boldsymbol{S}$: We start with the closed form solution we derived for $\boldsymbol{S}$ at a given time and location. Recall that $\boldsymbol{S} \sim N(\mathbf{0}, \boldsymbol{I})$.

$$\boldsymbol{S}_{it} = \frac{\sum_{v=1}^{2}(\boldsymbol{\phi}_{it}^{v} - \boldsymbol{\mu}^{v})(\boldsymbol{\Phi}^{v})^{-1}\boldsymbol{W}^{v\prime}}{\boldsymbol{I} + \sum_{v=1}^{2}\boldsymbol{W}^{v}(\boldsymbol{\Phi}^{v})^{-1}\boldsymbol{W}^{v\prime}} = \boldsymbol{N}\boldsymbol{D}^{-1} \tag{B.23}$$

$$\frac{\partial \boldsymbol{S}_{it}}{\partial M_{ijt}^{1}} = -\boldsymbol{N}\boldsymbol{D}^{-1}\frac{\partial \boldsymbol{D}}{\partial M_{ijt}^{1}}\boldsymbol{D}^{-1} + \frac{\partial \boldsymbol{N}}{\partial M_{ijt}^{1}}\boldsymbol{D}^{-1} \tag{B.24}$$

$$\frac{\partial \boldsymbol{D}}{\partial M_{ijt}^{1}} = \sum_{v}\boldsymbol{W}^{v}(\boldsymbol{\Phi}^{v})^{-1}\left(\frac{\partial \boldsymbol{W}^{v}}{\partial M_{ijt}^{1}}\right)' + \left(\frac{\partial \boldsymbol{W}^{v}}{\partial M_{ijt}^{1}}\right)(\boldsymbol{\Phi}^{v})^{-1}\boldsymbol{W}^{v\prime} \tag{B.25}$$

$$- \boldsymbol{W}^{v}(\boldsymbol{\Phi}^{v})^{-1}\frac{\partial \boldsymbol{\Phi}^{v}}{\partial M_{ijt}^{1}}(\boldsymbol{\Phi}^{v})^{-1}\boldsymbol{W}^{v\prime} \tag{B.26}$$

$$\frac{\partial \boldsymbol{N}}{\partial M_{ijt}^{1}} = (\boldsymbol{\phi}_{it}^{1} - \boldsymbol{\mu}^{1})(\boldsymbol{\Phi}^{1})^{-1}\left(\frac{\partial \boldsymbol{W}^{1}}{\partial M_{ijt}^{1}}\right)' + \frac{\partial(\boldsymbol{\phi}_{it}^{1} - \boldsymbol{\mu}^{1})}{\partial M_{ijt}^{1}}(\boldsymbol{\Phi}^{1})^{-1}\boldsymbol{W}^{1\prime} \tag{B.27}$$

$$- (\boldsymbol{\phi}_{it}^{1} - \boldsymbol{\mu}^{1})(\boldsymbol{\Phi}^{1})^{-1}\frac{\partial \boldsymbol{\Phi}^{1}}{\partial M_{ijt}^{1}}(\boldsymbol{\Phi}^{1})^{-1}\boldsymbol{W}^{1\prime} \tag{B.28}$$

$$+ (\boldsymbol{\phi}_{it}^{2} - \boldsymbol{\mu}^{2})(\boldsymbol{\Phi}^{2})^{-1}\left(\frac{\partial \boldsymbol{W}^{2}}{\partial M_{ijt}^{1}}\right)' - (\boldsymbol{\phi}_{it}^{2} - \boldsymbol{\mu}^{2})(\boldsymbol{\Phi}^{2})^{-1}\frac{\partial \boldsymbol{\Phi}^{2}}{\partial M_{ijt}^{1}}(\boldsymbol{\Phi}^{2})^{-1}\boldsymbol{W}^{2\prime} \tag{B.29}$$

B.2 Eigenvector and Eigenvalue Derivative

These derivatives are derived using the process presented in Fox and Kapoor [1968]. Let $\boldsymbol{A}$ be a square matrix with eigenvalues $\lambda_1, ..., \lambda_k$ and eigenvectors $\boldsymbol{v}_1, ..., \boldsymbol{v}_k$, where each eigenvector has unit norm and the eigenvalues are distinct. From the definition of eigenvalues, we have

$$\boldsymbol{A}\boldsymbol{v}_i = \lambda_i \boldsymbol{v}_i \tag{B.30}$$

$$\boldsymbol{A}\boldsymbol{V} = \boldsymbol{V}\boldsymbol{\Lambda} \tag{B.31}$$

Here $\boldsymbol{V}$ is the matrix of eigenvectors and $\boldsymbol{\Lambda}$ is the diagonal matrix of eigenvalues. It can easily be shown that $\boldsymbol{V}^{-1}$ is the set of left eigenvectors, as $\boldsymbol{V}^{-1}\boldsymbol{A} = \boldsymbol{\Lambda}\boldsymbol{V}^{-1}$. Assume that $\boldsymbol{A}$, and by extension λ and $\boldsymbol{v}$, can be parameterized by the scalar t. If we take the partial with respect to t on both sides of equation (B.30) we get

$$\frac{\partial \boldsymbol{A}}{\partial t}\boldsymbol{v_i} + \boldsymbol{A}\frac{\partial \boldsymbol{v}_i}{\partial t} = \frac{\partial \lambda_i}{\partial t}\boldsymbol{v}_i + \lambda_i \frac{\partial \boldsymbol{v}_i}{\partial t} \tag{B.32}$$

We can pre-multiply both sides by $\boldsymbol{v}_i^{-1}$, which is the ith row of $\boldsymbol{V}^{-1}$, and then simplify. Recall that $\boldsymbol{v}_i^{-1}\boldsymbol{A} = \lambda_i \boldsymbol{v}_i^{-1}$, since $\boldsymbol{V}^{-1}$ are the left eigenvectors of $\boldsymbol{A}$.

$$\boldsymbol{v}_i^{-1}\frac{\partial \boldsymbol{A}}{\partial t}\boldsymbol{v_i} + \boldsymbol{v}_i^{-1}\boldsymbol{A}\frac{\partial \boldsymbol{v}_i}{\partial t} = \boldsymbol{v}_i^{-1}\frac{\partial \lambda_i}{\partial t}\boldsymbol{v}_i + \boldsymbol{v}_i^{-1}\lambda_i \frac{\partial \boldsymbol{v}_i}{\partial t} \tag{B.33}$$

$$\boldsymbol{v}_i^{-1}\frac{\partial \boldsymbol{A}}{\partial t}\boldsymbol{v_i} + \lambda_i\boldsymbol{v}_i^{-1}\frac{\partial \boldsymbol{v}_i}{\partial t} = \frac{\partial \lambda_i}{\partial t} + \lambda_i\boldsymbol{v}_i^{-1}\frac{\partial \boldsymbol{v}_i}{\partial t} \tag{B.34}$$

$$\boldsymbol{v}_i^{-1}\frac{\partial \boldsymbol{A}}{\partial t}\boldsymbol{v_i} = \frac{\partial \lambda_i}{\partial t} \tag{B.35}$$

This gives us the derivative of the eigenvalues with respect to $\boldsymbol{A}$ and t.

For the eigenvector derivative we first rearrange equation (B.30), take the derivative with respect to t, and substitute in the the eigenvalue derivative from above.

$$(\boldsymbol{A} - \lambda_i I)\boldsymbol{v}_i = 0 \tag{B.36}$$

$$(\boldsymbol{A} - \lambda_i I)\frac{\partial \boldsymbol{v}_i}{\partial t} + \left(\frac{\partial \boldsymbol{A}}{\partial t} - \frac{\partial \lambda_i}{\partial t}I\right)\boldsymbol{v}_i = 0 \tag{B.37}$$

$$(\boldsymbol{A} - \lambda_i I)\frac{\partial \boldsymbol{v}_i}{\partial t} = -\left(\frac{\partial \boldsymbol{A}}{\partial t} - \boldsymbol{v}_i^{-1}\frac{\partial \boldsymbol{A}}{\partial t}\boldsymbol{v_i}I\right)\boldsymbol{v}_i = \boldsymbol{f}_i \tag{B.38}$$

This is a k x k system of equations, but it is not well defined because the matrix $\boldsymbol{A} - \lambda_i I$ has rank $k-1$. We will have to use this system along with the norm constraint on $\boldsymbol{v}_i$ to fully solve for $\frac{\partial \boldsymbol{v}_i}{\partial t}$.

Since $\boldsymbol{V}$ forms a basis, we can rewrite $\frac{\partial \boldsymbol{v}_i}{\partial t}$ as a linear combination of the eigenvectors.

$$\frac{\partial \boldsymbol{v}_i}{\partial t} = \sum_{i=1}^{k} c_i \boldsymbol{v}_i = \boldsymbol{V}\boldsymbol{c} \tag{B.39}$$

This shifts our task into solving for the vector $\boldsymbol{c}$ to find $\frac{\partial \boldsymbol{v}_i}{\partial t}$. Substituting this value into (B.38) and premultiplying by $\boldsymbol{V}^{-1}$ we get

$$\boldsymbol{V}^{-1}(\boldsymbol{A}-\lambda_i I)\boldsymbol{V}\boldsymbol{c} = \boldsymbol{V}^{-1}\boldsymbol{f}_i \tag{B.40}$$

$$(\boldsymbol{\Lambda}-\lambda_i I)\boldsymbol{c} = \boldsymbol{V}^{-1}\boldsymbol{f}_i \tag{B.41}$$

$$c_j = \frac{\boldsymbol{v}_j^{-1}\boldsymbol{f}_i}{\lambda_j - \lambda_i} \tag{B.42}$$

This gives us the values of $\boldsymbol{c}$ except for c_i. A constraint of the CCA formulation is the unit variance of the projections. More concretely, we must have $\boldsymbol{V}$ such that $\boldsymbol{V}'\boldsymbol{\Sigma}\boldsymbol{V} = \boldsymbol{I}$, where $\boldsymbol{\Sigma}$ is the covariance of the corresponding viewpoint. Utilizing this constraint, we can derive an expression for c_i. We start with the constraint for $\boldsymbol{v}_i$, and then take the derivative with respect to t.

$$\boldsymbol{v}_i'\boldsymbol{\Sigma}\boldsymbol{v}_i = 1 \tag{B.43}$$

$$2\boldsymbol{v}_i'\boldsymbol{\Sigma}\frac{\partial \boldsymbol{v}_i}{\partial t} + \boldsymbol{v}_i'\frac{\partial \boldsymbol{\Sigma}}{\partial t}\boldsymbol{v}_i = 0 \tag{B.44}$$

$$2\boldsymbol{v}_i'\boldsymbol{\Sigma}\boldsymbol{V}\boldsymbol{c} = -\boldsymbol{v}_i'\frac{\partial \boldsymbol{\Sigma}}{\partial t}\boldsymbol{v}_i \tag{B.45}$$

$$\sum_{j\neq i}\left(2\boldsymbol{v}_i'\boldsymbol{\Sigma}\boldsymbol{v}_j c_j\right) + 2\boldsymbol{v}_i'\boldsymbol{\Sigma}\boldsymbol{v}_i c_i = -\boldsymbol{v}_i'\frac{\partial \boldsymbol{\Sigma}}{\partial t}\boldsymbol{v}_i \tag{B.46}$$

$$2c_i = -\boldsymbol{v}_i'\frac{\partial \boldsymbol{\Sigma}}{\partial t}\boldsymbol{v}_i \tag{B.47}$$

$$c_i = -\frac{1}{2}\boldsymbol{v}_i'\frac{\partial \boldsymbol{\Sigma}}{\partial t}\boldsymbol{v}_i \tag{B.48}$$

Note that we make use of the fact that $\boldsymbol{V}'\boldsymbol{\Sigma}\boldsymbol{V} = \boldsymbol{I}$, which means $\boldsymbol{v}_i'\boldsymbol{\Sigma}\boldsymbol{v}_i = 1$ and $\boldsymbol{v}_i'\boldsymbol{\Sigma}\boldsymbol{v}_j = 0$. With every value of $\boldsymbol{c}$ defined, we now have an expression for $\frac{\partial \boldsymbol{v}_i}{\partial t}$.

In the previous sections we required the derivative of the canonical vectors $\boldsymbol{U}^i$ and canonical correlations $\boldsymbol{p}$ for CCA. For this problem, let $t = M_{ijt}^1$ and $\boldsymbol{A}^1 = (\Sigma^{11})^{-1}\Sigma^{12}(\Sigma^{22})^{-1}\Sigma^{21}$. Then the canonical vectors are the eigenvectors of $\boldsymbol{A}^i$, and the canonical correlations are the square root of the eigenvalues. This means that $\partial p_k = \partial(\sqrt{\lambda_k}) = \frac{\partial \lambda_k}{2\sqrt{\lambda_k}}$. Both the eigenvector and eigenvalue derivatives depend on the derivative of $\boldsymbol{A}$, which we must also derive.

$$\frac{\partial \boldsymbol{A}^1}{\partial M^1_{ijt}} = -(\Sigma^{11})^{-1}\frac{\partial \boldsymbol{\Sigma}^{11}}{\partial M^1_{ijt}}(\Sigma^{11})^{-1}\Sigma^{12}(\Sigma^{22})^{-1}\Sigma^{21} \tag{B.49}$$

$$+ (\Sigma^{11})^{-1}\frac{\partial(\boldsymbol{\phi}^1 - \boldsymbol{\mu}^1)'}{\partial M^1_{ijt}}\frac{(\boldsymbol{\phi}^2 - \boldsymbol{\mu}^2)}{n}(\Sigma^{22})^{-1}\Sigma^{21} \tag{B.50}$$

$$+ (\Sigma^{11})^{-1}\Sigma^{12}(\Sigma^{22})^{-1}\frac{(\boldsymbol{\phi}^2 - \boldsymbol{\mu}^2)'}{n}\frac{\partial(\boldsymbol{\phi}^1 - \boldsymbol{\mu}^1)}{\partial M^1_{ijt}} \tag{B.51}$$

$$\frac{\partial \boldsymbol{A}^2}{\partial M^1_{ijt}} = (\Sigma^{22})^{-1}\frac{(\boldsymbol{\phi}^2 - \boldsymbol{\mu}^2)'}{n}\frac{\partial(\boldsymbol{\phi}^1 - \boldsymbol{\mu}^1)}{\partial M^1_{ijt}}(\Sigma^{11})^{-1}\Sigma^{12} \tag{B.52}$$

$$- (\Sigma^{22})^{-1}\Sigma^{21}(\Sigma^{11})^{-1}\frac{\partial \boldsymbol{\Sigma}^{11}}{\partial M^1_{ijt}}(\Sigma^{11})^{-1}\Sigma^{12} \tag{B.53}$$

$$+ (\Sigma^{22})^{-1}\Sigma^{21}(\Sigma^{11})^{-1}\frac{\partial(\boldsymbol{\phi}^1 - \boldsymbol{\mu}^1)'}{\partial M^1_{ijt}}\frac{(\boldsymbol{\phi}^2 - \boldsymbol{\mu}^2)}{n} \tag{B.54}$$

This is the final piece to construct the full gradient from equation B.9.