# STATISTICAL ESTIMATION OF CROP MANAGEMENT ZONES FROM MULTI-YEAR YIELD DATA AND THE OADA API FRAMEWORK

by

**Alex Layton**

**A Dissertation**

*Submitted to the Faculty of Purdue University*

*In Partial Fulfillment of the Requirements for the degree of*

**Doctor of Philosophy**


School of Electrical and Computer Engineering

West Lafayette, Indiana

December 2021

# THE PURDUE UNIVERSITY GRADUATE SCHOOL
# STATEMENT OF COMMITTEE APPROVAL

**Dr. James Krogmeier, Chair**

School of Electrical and Computer Engineering

**Dr. Dennis Buckmaster**

Department of Agricultural and Biological Engineering

**Dr. Mark Bell**

School of Electrical and Computer Engineering

**Dr. David Love**

School of Electrical and Computer Engineering

**Approved by:**

Dr. Dimitrios Peroulis

## ACKNOWLEDGMENTS

The author would like to thank Ault Farms for supplying the yield data used in the development and evaluation of the described algorithm.

This work was partially supported by the Foundation for Food and Agricultural Research (FFAR) under award number 534662.

This work was partially supported by and AWS Cloud Credits for Research.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# LIST OF SYMBOLS

| | |
|---|---|
| $\mathcal{N}$ | Multivariate Gaussian distribution |
| $\mathcal{U}(a, b)$ | Continuous uniform distribution on interval $(a, b)$ |
| $\mathcal{U}[a, b]$ | Discrete uniform distribution on interval $[a, b]$ |
| $E[\ \cdot\ ]$ | Expectation operator |
| $X$ | $M \times N$ random field of management zone labels |
| $Y$ | $M \times N \times P$ random field of yield observations |
| $Y_s$ | $P$-vector of yield observations for coordinate $s$ |
| $\theta$ | Set of estimated model parameters |
| $S$ | Set of all field coordinates |
| $\partial s$ | Set of all field coordinates neighboring coordinate $s$ |
| $\delta(\cdot)$ | Kronecker delta function |
| $\varnothing$ | The empty set |
| $\|\| \cdot \|\|_F$ | Frobenius norm |

# ABBREVIATIONS

| | |
|---|---|
| API | Application Programming Interface 13, 70–74, 76, 77, 79, 80, 84, 86, 88, 91–96, 99, 101, 103, 105, 108 |
| CDF | cumulative distribution function 124, 125 |
| EM | expectation-maximization 36, 38, 64–66, 121 |
| ERU | environmental response unit 63 |
| HMRF | hidden Markov random field 28, 33, 36, 124 |
| HTTP | Hypertext Transfer Protocol 72, 74, 76–78, 93, 96, 97, 99 |
| HTTPS | Hypertext Transfer Protocol Secure 76, 99 |
| i.i.d. | independent and identically distributed 125 |
| ISO | the International Organization for Standardization 72 |
| JSON | JavaScript Object Notation 76, 78, 80, 85, 88, 90, 95, 99 |
| JWT | JSON Web Token 80 |
| MAP | maximum a posteriori 40 |
| ML | maximum likelihood 39, 121, 122 |
| MPM | maximizer of the posterior marginals 40, 64 |
| MRF | Markov random field 30, 33, 38, 124 |
| MZA | Management Zone Analyst 13, 52–54, 56–59 |
| OADA | the Open Ag Data Alliance 11, 14, 71, 76–88, 90–98, 100–106, 108 |
| OData | Open Data Protocol 72 |
| pdf | probability density function 118, 124 |
| pmf | probability mass function 30, 36, 124 |
| PoC | Proof-of-Concept 71, 96, 99 |
| REST | Representational State Transfer 70, 72–74, 76, 77, 93, 94, 105 |
| RFC | Request for Comments 82, 83 |
| SEM | stochastic expectation-maximization 11–13, 36, 40, 42, 49–54, 63, 106 |
| URI | Uniform Resource Identifier 73 |
| URL | Uniform Resource Locator 76, 77, 79, 84, 85, 93, 103 |

UTM Universal Transverse Mercator 13, 25, 52, 53

XML Extensible Markup Language 76

# GLOSSARY

| | |
|---|---|
| change feed | OADA change feed 84 |
| client | A piece of software which interacts with a server using an API. 14, 72, 77, 79, 80, 85, 88, 91–94, 98, 102, *see* server & API |
| link | OADA link 82, 84–86, 88 |
| meta resource | OADA resource meta document 82, 84 |
| resource | OADA resource 77–79, 82, 84, 88, 91, 98, 99 |
| server | A platform which hosts an API and serves API requests from clients. 14, 72, 88, 92–94, 96, 97, 102, 103, 105, *see* client & API |
| SOAP | formerly Simple Object Access Protocol 72 |
| Trellis | The Trellis Framework 102, 103 |
| unversioned link | OADA unversioned link 86 |
| user | A person who interacts with a client. Users do not interact directly with servers or APIs. 70, 79–82, 84, 91, 92, 97, 98, 101–103, *see* client, server & API |
| user account | A representation of a user on server. Users may have multiple user accounts on the same server and accounts across multiple servers. 79–81, *see* user & server |
| versioned link | OADA versioned link 86, 88, 90, 98 |
| WebSocket | WebSocket 76 |

# ABSTRACT

Precision agriculture equipment enables treating different areas of a field differently (i.e., site-specific management). The first part of this work presents an algorithm for inferring the management zones of fields based on multiple years' yield data. It seeks regions that correspond to the same underlying yield distribution. Zones are assumed to be the same each year, but their distributions are allowed to change year-to-year to account for variability. Zones are estimated using stochastic expectation maximization and maximization of the posterior marginals. The underlying assumption is that the yields corresponding to a given zone will behave similarly, and are drawn from the same distribution. This requires only the yield data automatically collected during harvest. This method requires no crop-specific calibration.

The second part of this work presents the Open Ag Data Alliance (OADA) Application Programming Interface (API) framework. It is a generic specification that can be used by third parties' APIs to reduce the complexity of interoperating with multiple entities. This is especially useful in intercloud scenarios, for example, moving data between a farmer, a processor, and a distributor. Several existing standards that were leveraged are identified, the graph-based data representation is illustrated, and key API specifications and features are highlighted. Some of the contributions of OADA include user-centric Representational State Transfer (REST) so users can select API clients, resource meta-data stored externally to the resource, live data graphs via change feeds, intercloud data push, and format indifference. A reference implementation is presented and use cases are demonstrated.

# Part I

# STATISTICAL ESTIMATION OF CROP MANAGEMENT ZONES FROM MULTI-YEAR YIELD DATA

# 1. INTRODUCTION

The rise in precision agriculture has resulted in more and more machines having monitors which automatically record yield data during harvest. Such monitors also allow for easily breaking a field into smaller regions, referred to as "management zones", and applying different inputs to each of these regions (e.g., applying more fertilizer to one region than to others). Realizing these two things, it seems natural to try to use the output recorded at harvest to determine a set of management zones to use when deciding how to apply inputs.

This work presents a probabilistic model relating observed yields to management zones, and a corresponding algorithm for estimating the management zones based on the model. The model is designed to allow for year-to-year variability within a management zone. Such variability can come from sources like differences in weather (e.g., "wet" year or "dry" year), differences in measurement calibration (e.g., using a different combine), or growing different crops or crop varieties (which have different yield characteristics). The algorithm aims to find the likely management zone assignments for a field based on the yields recorded from multiple years' harvests and the management zone model.

Experimental results are shown in which the algorithm was run on real yield data spanning multiple years and multiple fields. The data were recorded automatically by the combines used for harvest. Simulated results are also shown in which the algorithm was run on simulated yields for which the "true" management zones are known. The data were simulated based on the real data collected. The simulated data were also run through a pre-existing algorithm for comparison purposes.

# 2. THE CROP YIELD DATA

## 2.1 Initial Data

The data used were yield data exported from a combine's monitor. The data had been collected automatically during the normal operation of the combine. The data comprised grain flow, speed, grain moisture, date, and GPS location. The collected data came from multiple fields, over multiple years. This paper will focus on the data of fields with multiple years of corn harvest data, as well as one field with multiple years of both corn harvest data and soybean harvest data. These fields used in the paper are shown in Fig. 2.1 and described in Table 2.1.

## 2.2 Data Pre-Processing

### 2.2.1 Unit Conversion

The collected data had wet grain flow versus time, but what was needed was dry yield versus area. The dry yield versus area was computed from the collected data according to (2.1)

$$y = \frac{f}{(5280v)\,(43560w)} \cdot \frac{100 - m}{100 - m_{dry}} \tag{2.1}$$

where

$$\begin{aligned} y &= \text{dry yield in bu/ac} \\ f &= \text{grain flow in bu/h} \\ v &= \text{speed in mi/h} \\ w &= \text{combine header width in ft} \\ m &= \text{grain moisture in \%} \\ m_{dry} &= \begin{cases} 15.5\%, & \text{for corn} \\ 13\%, & \text{for soybeans} \end{cases}. \end{aligned}$$

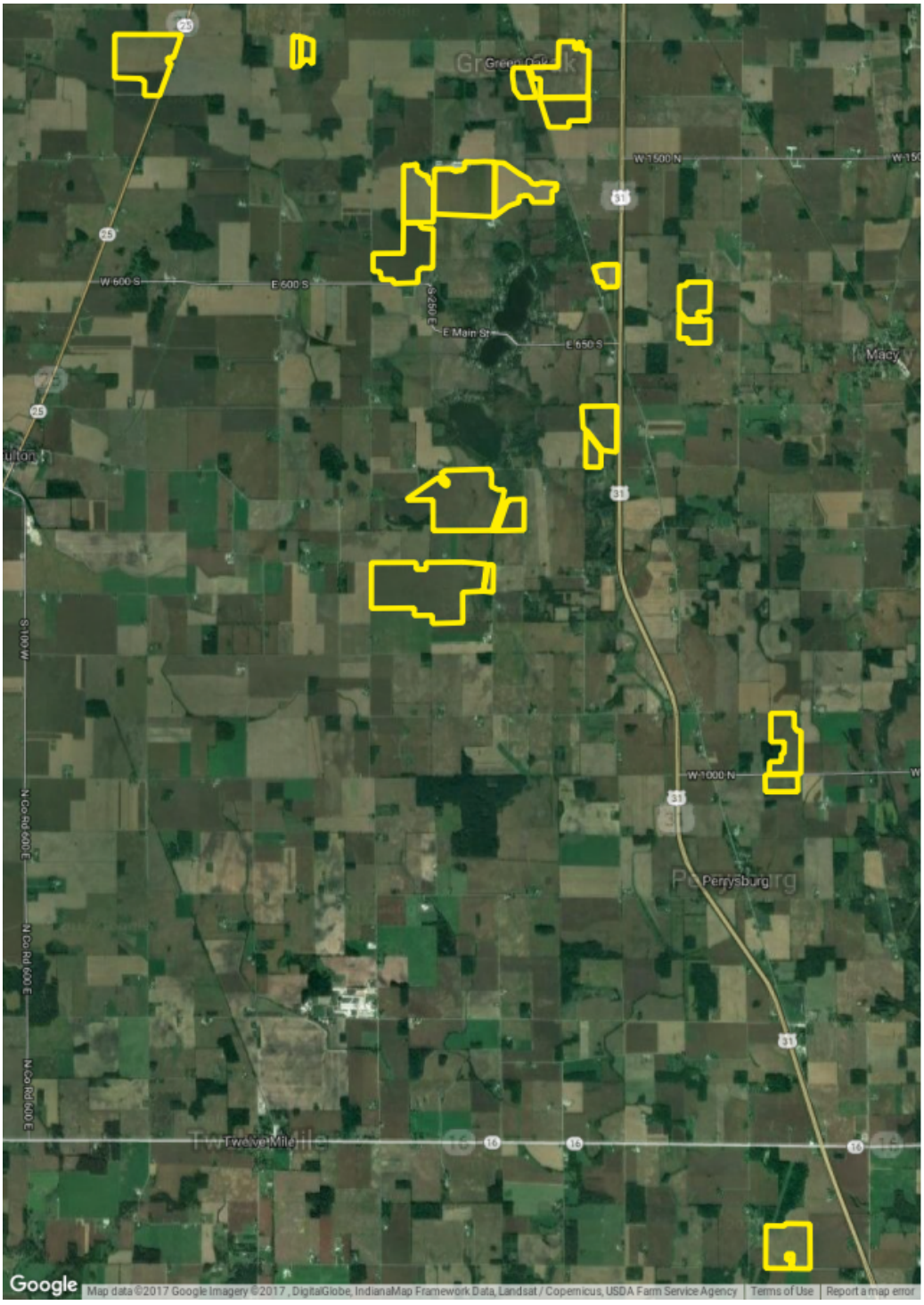



**Figure 2.1.** These are the outlines of the fields corresponding to the data used in this paper. For each of these fields, there were two or more years of corn yield data. The data span two counties in Indiana and correspond to roughly 1400 acres of farmland.

**Table 2.1.** This table lists the fields whose data were used in this work. Shown are the number of years of corn harvest data available and the number of acres covered by all the years' data.

| Field | Years of Data | Area (acres) |
|---|---|---|
| Rusty 100 | 3 | 94 |
| Boots 72 | 3 | 61 |
| Bank 53 | 4 | 53 |
| Church 17 | 4 | 19 |
| Coondog 45 | 4 | 43 |
| Deedsville North 63 | 5 | 61 |
| Deedsville South 24 | 5 | 23 |
| Macy 25 | 3 | 27 |
| Muck 17 | 5 | 17 |
| Drycow 61 | 5 | 65 |
| Eber 124 | 3 | 126 |
| Shackleford East 50 | 5 | 49 |
| Gott East 93 | 4 | 93 |
| Gott West 24 | 4 | 23 |
| Layton 192 | 3 | 197 |
| Home 128 | 5 | 72 |
| Horn 235 | 2 | 239 |
| Lillian South Mucks 21 | 4 | 21 |
| Mont North 100 | 2 | 86 |

The above calculation was performed for every collected data point (where one point was comprised of a wet flow measurement, a GPS position measurement, a speed measurement, and a moisture measurement). This resulted in a yield map that was non-uniformly sampled in space because the collected data were non-uniformly sampled in space. To remedy this, the data were interpolated to a uniform spatial grid.

### 2.2.2 Interpolation

For a given field, the grid used was the same across all the years of data. Before the actual interpolation could be performed for each year's data, this grid has to be determined. To simplify distance calculations, the GPS latitudes and longitudes were converted to Universal Transverse Mercator (UTM) coordinates [1]. Next, a bounding box was found for the set of all data points for the given field. Then the South-West corner of that box was used as the first grid point. Finally, the grid was expanded North and East, with the same spacing in each direction, to cover the bounding box.

Once the target grid is determined, the interpolation is performed for each year of data using a modified Shepard's method [2], as illustrated in Fig. 2.2. This interpolation method uses a weighted average of the points within a neighborhood of the new grid location, as shown in (2.2) and (2.3)

$$y_s = \frac{\sum_{\mathrm{i} \in N_{s,R}} y_\mathrm{i} w_\mathrm{i}(s)}{\sum_{\mathrm{i} \in N_{s,R}} w_\mathrm{i}(s)} \tag{2.2}$$

$$w_\mathrm{i}(s) = \left( \frac{R - \|s - \mathrm{i}\|}{R \, \|s - \mathrm{i}\|} \right)^2 \tag{2.3}$$

where

$$\begin{aligned} y_s &= \text{yield value for coordinate } s \\ \mathrm{i} &= \text{coordinate of uninterpolated data point} \\ s &= \text{coordinate of interpolated data point} \\ R &= \text{radius of interpolation neighborhood} \\ N_{s,R} &= \text{set of uninterpolated coordinates within } R \text{ of } s. \end{aligned}$$

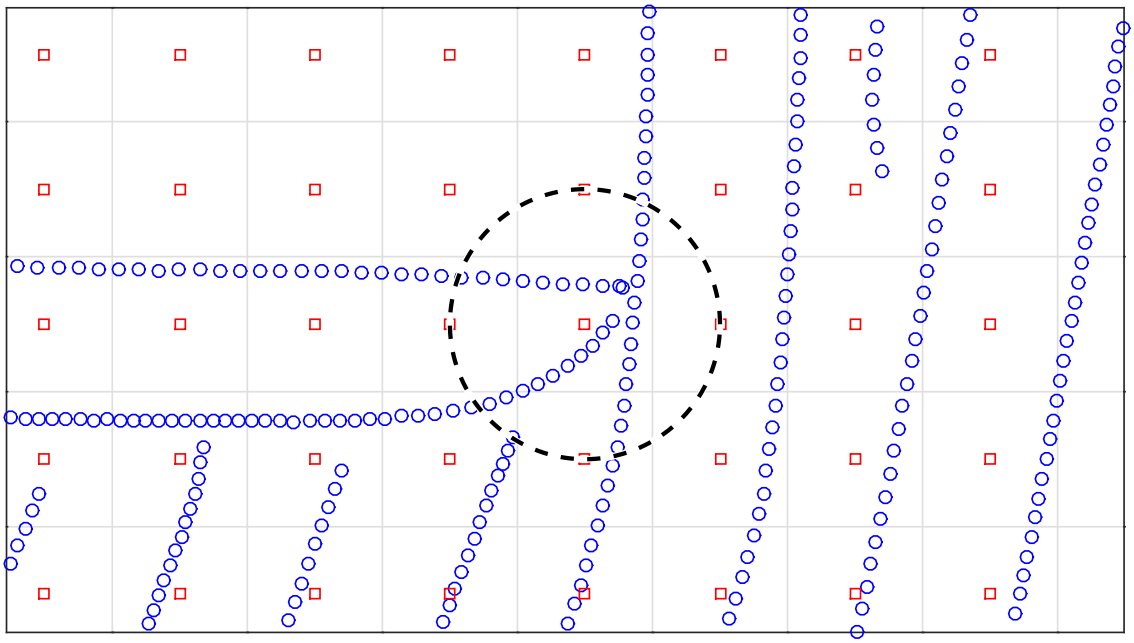


**Figure 2.2.** This is an illustration of the interpolation performed on the yield data. It is a section of the real data. The circular points are locations of the input non-uniform data (for a particular year), and the square points are the locations of the output uniform data (for all years). The grid shows the borders of the interpolation grid regions. For the interpolation location in the center of the figure, the interpolation neighborhood radius is shown with a dotted line.

In the case when there is no data inside the grid square (such as the squares in the upper left of Fig. 2.2), that location is given a yield of not a number (NaN). NaN is a possible value in computation, used to represent an undefined value [3]. When an arithmetic operation is performed with a NaN, the result is a NaN (thus, the undefined grid locations remain undefined in the algorithm output).

This interpolation is a form of Inverse Distance Weighting (IDW). IDW has been shown to perform well for interpolating GPS referenced yield measurements [4]. The modified version is used to prevent interpolation across field boundaries. The data were interpolated using a grid size of 10m and a neighborhood radius, $R$, of 10m. The size 10m was picked because it is on the order of the width of a combine header. After this interpolation is run on each year of data for the field, the outputs for each grid location (i.e., each $s$) are combined into vectors, which is why it was necessary to determine a grid based on all the years before interpolating the years separately.

# 3. CROP MODEL

This work employs a hidden Markov random field (HMRF). The model is comprised of two parts [5], a model for the unobserved (i.e., "hidden") management zones, and a conditional model of the observed crop yields. These models all assume the data are on a uniform spatial grid, and thus represent them with matrices and vectors.

It is worth noting that while there are certainly other data that can be used in the determination of management zones, e.g., soil type and topography [6], they are not included in this model. The model and algorithm were developed to leverage the readily available yield data, and so that is what is used in determining management zones. While the model does not incorporate such input data, it should be able to discern their effect on the output yields given enough years of yield observations.

## 3.1 Management Zone Model

A "management zone" can be many things, depending on who is asked and the context. A common agronomic interpretation of "management zones" is the regions of a field having similar yield potential, or "yield zones", sometimes also called "response zones" or yield "productivity zones" (YPZ) [7]–[9]. This work, and the described model, take this "yield zone" view of what a management zone is. This is not an assumption, but rather a definition of management zones as regions of a field having similar productivity potential.

For the model, a management zone is viewed as a region of the field where the corresponding yields have the same underlying distribution. The management zones are assumed to be constant year-to-year, but the distributions of their corresponding yields are allowed to change over time.

A management zone assignment for the field is denoted $X$, and the assignment for a particular location, $s$, is denoted $X_s$. The value of $X_s$ is represented as an integer between 0 and $K-1$ (where $K$ is the total number of management zones), or NaN if the location $s$ is deemed not in the field (i.e., any year was missing data for that grid location). Therefore, $X$ is represented as a matrix where each element is the management zone assignment for

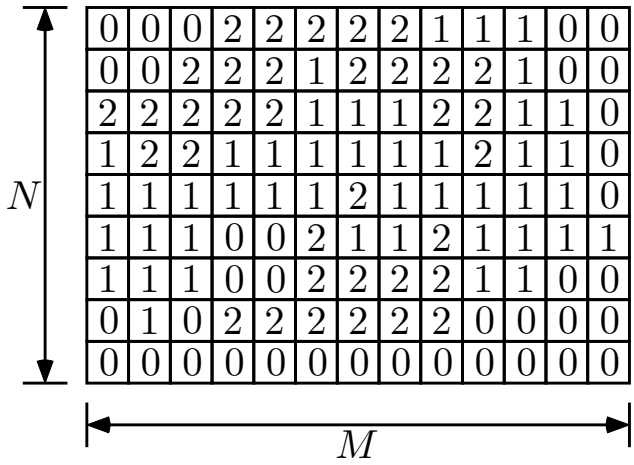



**Figure 3.1.** This is an illustration of an example management zone assignment image (denoted $X$). In this figure $N$ (one spatial dimension) is 9, and $M$ (the other spatial dimension) is 13. Since the element values range from 0 to 2, the corresponding $K$ (number of management zones) is 3.

the corresponding grid location. Figure 3.1 illustrates the representation of the management zones for the case of an $M$ by $N$ spatial grid.

The matrix of management zone assignments is modeled as a Markov random field (MRF). An MRF is a set of random variables (e.g., the set of elements of $X$) such that the conditional probability of one element of the set given all the other elements of the set depends only on the neighboring elements, i.e., it satisfies the Markov property (3.1) [5]

$$P(X_s \mid X_r,\ \forall r \neq s) = P(X_s \mid X_r,\ \forall r \in \partial s) \tag{3.1}$$

where

$$\partial s = \text{set of all coordinates neighboring coordinate } s.$$

Since neighboring locations are more likely to be in the same management zone, a Potts model is used for the management zone MRF [10]–[12]. The specific probability mass function (pmf) used is shown in (3.2)

$$X \sim p(x) = \frac{1}{z} \mathrm{e}^{-\beta \sum_{s,r \in S} b_{|s-r|} \delta(x_r \neq x_s)} \tag{3.2}$$

$$b_{|s-r|} = \begin{cases} \frac{1}{4\left(1+\sqrt{2}\right)}, & \text{for } |s-r| = \sqrt{2} \\ \frac{1}{2\left(2+\sqrt{2}\right)}, & \text{for } |s-r| = 1 \end{cases} \tag{3.3}$$

where

$z =$ partition function

$\beta =$ smoothness factor

$\delta =$ Kronecker delta function

$b =$ neighbor weights

$S =$ set of all coordinates on uniform spatial grid.

This model uses the Hamming distance between management zone assignments, which handles the fact that the values of the assignments have no numerical meaning (i.e., zone 1 is not "greater" or "lesser" than zone 2 in any particular way). The parameter $\beta$, when positive, causes neighboring elements to tend to be similar, with larger values increasing this likelihood of similarity. The value of $\beta$ used was 10, but could be tweaked to produce management zone more or less smooth edges if needed.

### 3.2 Yield Model

The yield model describes the distribution of yields within a given management zone. The set of interpolated yields for the $P$ years of data for the field is denoted $Y$. $Y$ is a 3-dimensional object with dimensions $M \times N \times P$, where the $M$ and $N$ dimensions correspond to space and the $P$ dimension corresponds to time. This means the yield observation for a location, $s$, is a $P$-vector denoted $Y_s$. Figure 3.2 illustrates the representation of the yield data for a case with 3 years of data, with a yield vector for a particular location highlighted. As shown in (3.4),

$$(Y_s \mid X_s = k) \sim \mathcal{N}(\mu_k, R_k) \tag{3.4}$$

where

$$\mu_k \in \mathbb{R}^P$$
$$R_k \in \mathbb{R}^{P \times P},$$

the yield vectors are assumed Gaussian, given their management zone assignment. The mean and covariance of their distribution depend on the value of their management zone assignment. The covariance matrices, $R_k$, are non-diagonal, allowing for correlations between the yields of different years. It is worth noting these are conditional distributions on the yield, they are not the unconditional distribution on the yield.

Estimating the means and variances for each year allows the model to handle year-to-year variability in the crops, or even different crops being planted on different years. This

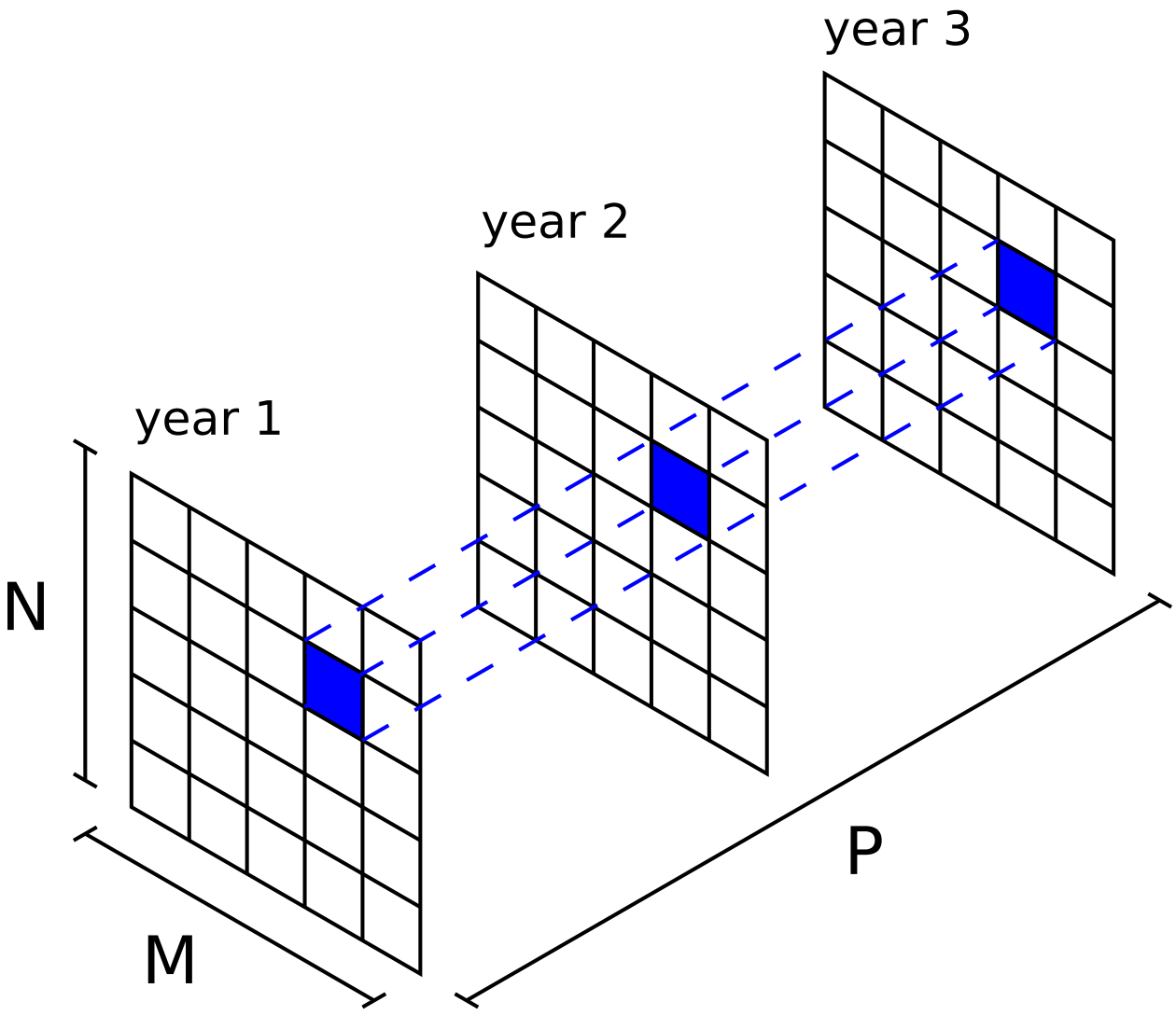



**Figure 3.2.** This is an illustration of an example input yield array (denoted $Y$). In this figure $N$ (one spatial dimension) is 5, $M$ (the other spatial dimension) is 5, and $P$ (the temporal dimension) is 3. The yield vector (denoted $Y_s$) for a particular location (e.g., $s$) is highlighted.

removes the need for yield normalization that is typically done when dealing with multiple years [7], [8], which can result in loss of information [8].

The yield vectors are assumed conditionally independent of one another, given their respective management zone assignments. However, because the management zones are modeled with an MRF, the model does not make the yield vectors unconditionally independent. This means the model still expects nearby yields to be similar (i.e., there is spatial dependence of the yields). This form of conditional independence is a required property of an HMRF [5].

### 3.3 Temporal Variability of Model

For the purposes of this model, management zones are defined to be regions whose yields all correspond to a particular probability distribution in a given year. It is important to note that while the management zone *assignments* are fixed across all years, the management zone *behavior* is allowed to vary year-to-year in the described model.

One might want to think of these yield zones in terms of good zones, bad zones, etc. However, a given zone is not necessarily always good or always bad [13]. A given zone may have good yields in a wet year, but have bad yields in a dry year (or vice versa). The idea is that while the year-to-year behavior of these zones may not be fixed, the assignment of these zones is modeled to be the same from year to year. That is, each year all of the field that is a part of a given zones will have similar productivity in that year (be that good yields, bad yields, etc.).

An illustration of these fixed assignments with varying yield behavior is shown in Fig. 3.3. This example covers three years' yields. The first year was a wet year, and the histograms for that year's yield are shown in blue. In this year, the high ground and side slope both had good mean yields, but the low ground did not because it became flooded. The second year was an average year, and the histograms for that year's yield are shown in green. In this year, all the zones performed similarly average means because there was no flooding or drought. The third year was a dry year, and the histograms for that year's yield are shown

in red. In this year the low ground had the best mean and the high ground had the worst because the higher ground dried out more.

In reality, there is more than just topography that can determine the difference between zones and there is more than just wet vs dry that causes year-to-year variability of yields, but this simple example shows how the zone assignments are consistent even when the yields vary each year.

The fact that the zones assignments are modeled to be constant year-to-year is intentional. When a field is planted, much of what may influence the productivity of zones is unknown (e.g., weather or pestilence). Despite this, the field management still needs to be decided, ideally in such a way as to minimize risk from the unknown factors. Knowing which regions of the field will have similar productivity will simplify these decisions. Also, while the model does not predict yields, past behavior of a zone could be used in determining management. Once one knows where a zone is, they could, for example, look at how that zone behaved in previous dry years and plan accordingly if the coming year were expected to be a dry one.

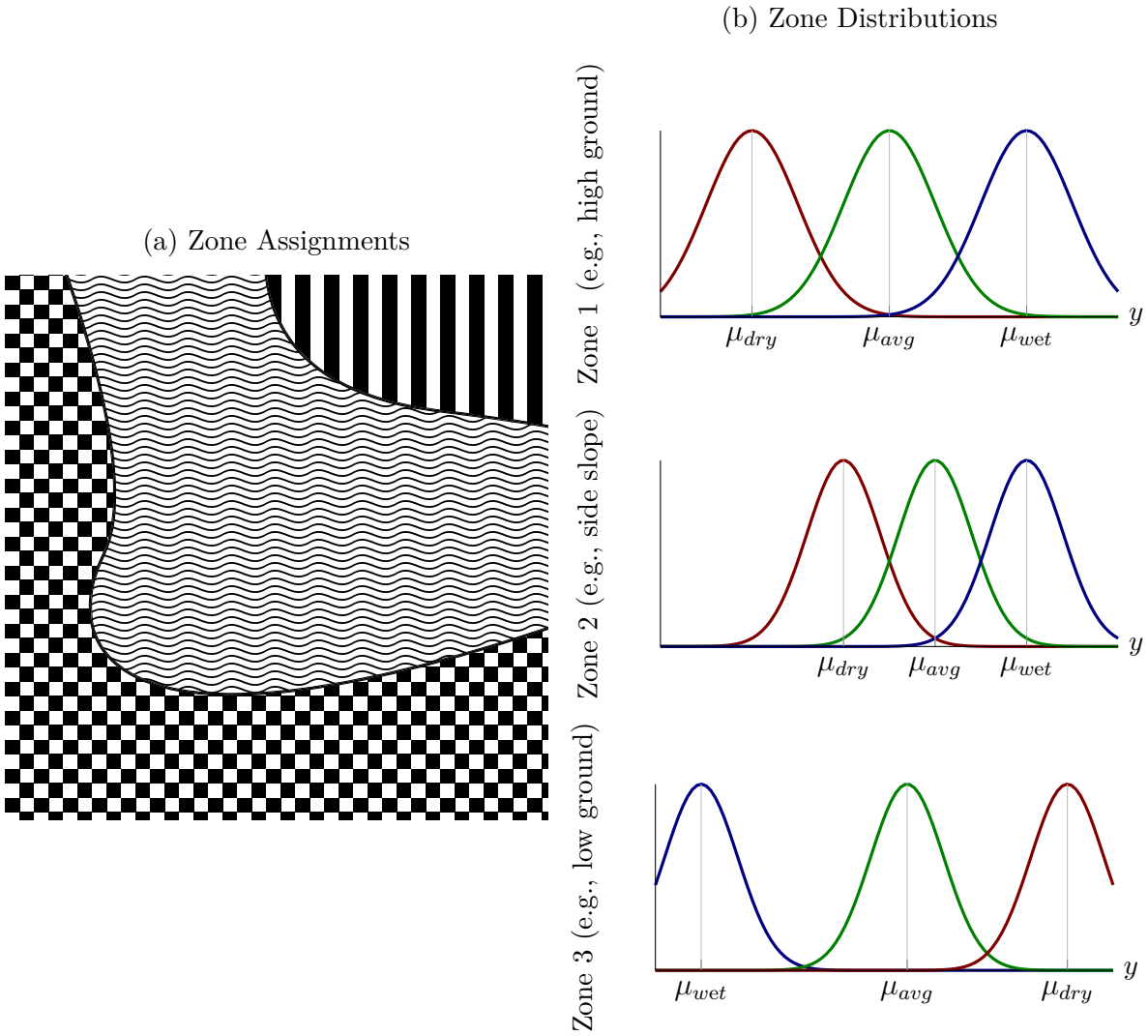



**Figure 3.3.** This diagram illustrates a simplified example of the model of management zones which have fixed assignments year-to-year, yet still yield differently each year. A simple example field is shown in Fig. 3.3a which has three zones with a slope. The vertical lines are the high ground zone, the waves are the side slope zone, and the checkerboard is the low ground zone. These zones are fixed year-to-year. However, their yield distributions shown in Fig. 3.3b are different each year due to weather.

# 4. MANAGEMENT ZONE ESTIMATION ALGORITHM

The inputs to the algorithm are yield maps on a uniform spatial grid, and the number of management zones to find. The general idea of the algorithm is to find the parameter values for the model which maximize the probability of the observed yields. Once parameter estimates are obtained, the model can be used to find the most likely management zone assignments, given the observed yields.

The algorithm achieves this likelihood maximization in three stages. The first stage is making a rough guess at the model parameters using fuzzy c-means [14]. The second stage is iteratively improving the parameter estimates using a stochastic version of an expectation-maximization algorithm [5], [15] to maximize the probability of the observed yields given the estimated parameters. Lastly, once the model parameters are estimated, the most likely management zones are computed according to the model and the parameter estimates. The overall flow of the algorithm is shown in Fig. 4.1, and the different parts are detailed in the following subsections.

## 4.1 Stochastic Expectation-Maximization

Stochastic expectation-maximization (SEM) is an iterative method for calculating the most likely parameter estimates for a model with hidden variables, such as the management zones variable $X$ in the HMRF described in this paper. SEM differs from classical expectation-maximization (EM) in that it calculates sample means rather than true expectations. SEM is used instead of EM because, for the model and input sizes used, explicit calculations involving the pmf of $X$ are intractable.

There are three steps, described in the following sections. One iteration of the algorithm involves running the three steps in order. Multiple iterations are run until the parameter estimates have converged sufficiently. For this paper, "sufficient convergence" was assumed to occur within 100 iterations of SEM. The order and looping of these steps can be seen within the SEM block shown in Fig. 4.1. The specific EM step equations used in this algorithm are derived from the fact that the joint model of $X$ and $Y$ is an exponential family [15],

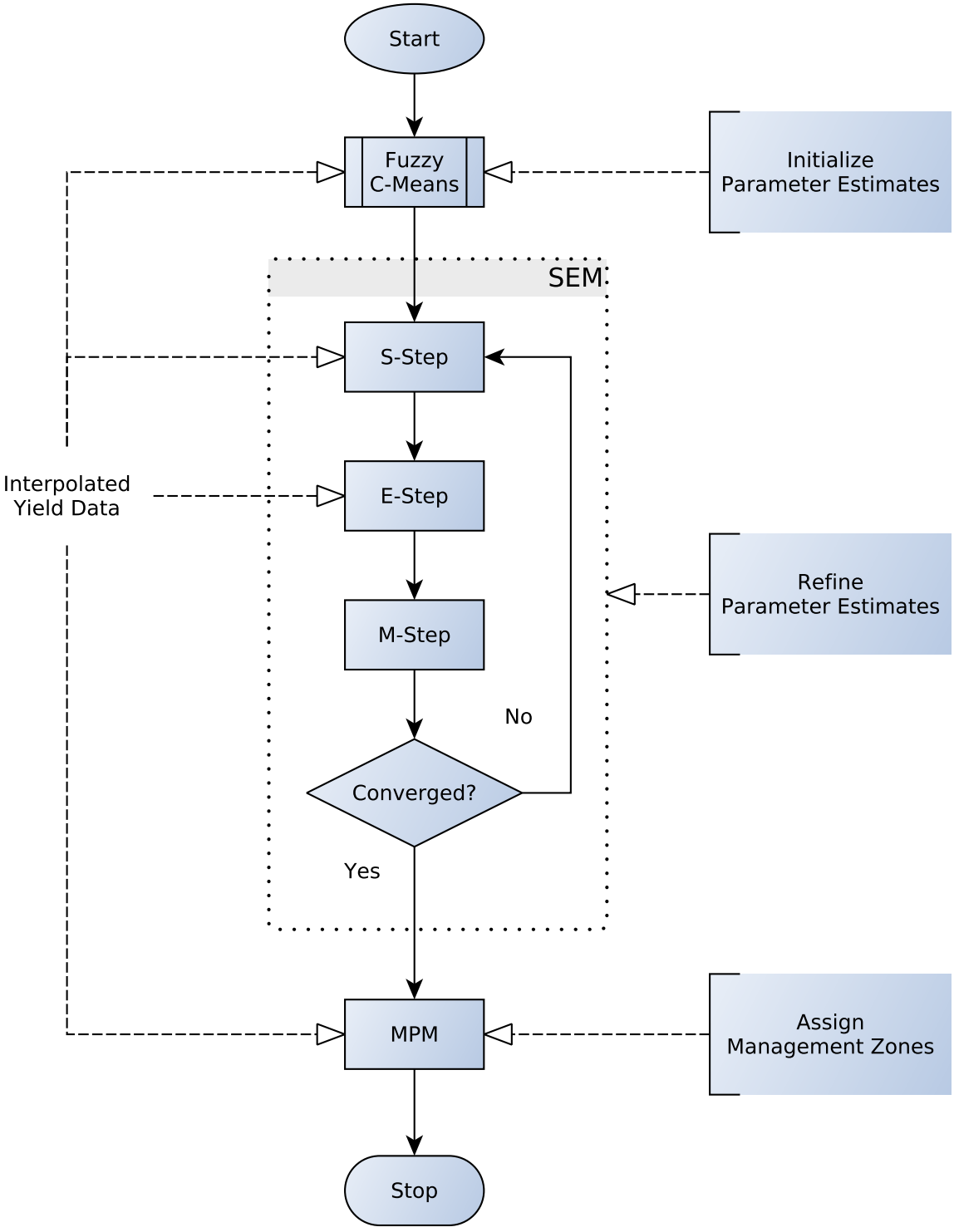



**Figure 4.1.** High level illustration of the steps of the algorithm. The figure also indicates which of the steps utilize the input yield data. The following sections give more detail on the steps.

[16]. A proof that the model is an exponential family can be found in Appendix A, and the derivation of the EM steps can be found in Appendix B.

### 4.1.1 Sampling (S-Step)

A Gibbs sampler [15], [17] is used to generate sample management zone assignment matrices, according to the conditional distribution of the management zone MRF given the current parameter estimates and the observed yields. This conditional distribution is stated mathematically in (4.1),

$$X^{(l)} \sim P(X^{(l)}) = P(X = X^{(l)} \mid Y, \theta) \qquad \text{for } l = 1, \ldots, L \tag{4.1}$$

where

$$\begin{aligned} X_s^{(l)} &= \text{value of } X_s \text{ in } l\text{th sample of MRF} \\ \theta &= [\mu_0, R_0, \ldots, \mu_{K-1}, R_{K-1}] \\ L &= \text{total number of generated MRF samples.} \end{aligned}$$

The sampler is used to generate $L$ separate samples, where each sample is an $N$ by $M$ matrix. For this work, $L = 10000$ was used. These generated samples are needed for computing sample means which will converge to the true expectations [15]. The detail of the Gibbs sampler used can be found in Appendix C.

### 4.1.2 Expectation of Conditional Statistics (E-Step)

Once samples of the MRF are generated, those samples can be used to calculate the sample means of the sufficient statistics for the $K$ Gaussian distributions of the yields in the $K$ management zones. There are three statistics calculated for each of the $K$ distributions,

shown as functions of $k$ in (4.2), (4.3), and (4.4) (i.e., a total of $3K$ statistics are calculated). These statistics are needed for computing new parameter estimates.

$$\bar{N}_k = \frac{1}{L} \sum_{l=1}^{L} \sum_{s \in S} \delta(X_s^{(l)} = k) \tag{4.2}$$

$$\bar{b}_k = \frac{1}{L} \sum_{l=1}^{L} \sum_{s \in S} Y_s \delta(X_s^{(l)} = k) \tag{4.3}$$

$$\bar{S}_k = \frac{1}{L} \sum_{l=1}^{L} \sum_{s \in S} Y_s Y_s^{\intercal} \delta(X_s^{(l)} = k) \tag{4.4}$$

$$\text{for } k = 0, \ldots, K-1$$

### 4.1.3 Maximum Likelihood Parameter Estimate Update (M-Step)

The statistics from the previous step are used to calculate maximum likelihood (ML) parameter estimates. Since, as mentioned earlier, the model describes an exponential family of distributions, the parameter update can be calculated by plugging the expected statistics from the E-step into the ML estimate equations of the distribution parameters in place of the actual statistics. The update equations for the estimated means and covariance matrices are (4.5) and (4.6), respectively.

$$\hat{\mu}_k = \frac{1}{\bar{N}_k} \bar{b}_k \tag{4.5}$$

$$\hat{R}_k = \frac{1}{\bar{N}_k} \bar{S}_k - \frac{1}{\bar{N}_k^2} \bar{b}_k \bar{b}_k^{\intercal} \tag{4.6}$$

$$\text{for } k = 0, \ldots, K-1$$

## 4.2 Parameter Initialization

Expectation-maximization algorithms require an initial guess of the model parameters, which are then iteratively improved upon. The algorithm uses fuzzy c-means [14] to generate its initial model parameters. Fuzzy c-means was chosen because it has been used before to find management zones [7].

### 4.3 Management Zone Assignment

The result of SEM iterations is an estimate of the parameters of the distribution of each management zone. However, we want to know to which management zone each location belongs. In order to estimate a likely set of management zone assignments, the maximizer of the posterior marginals (MPM) estimator is used. The MPM estimator is described in (4.7), though in the algorithm it is evaluated stochastically rather than explicitly.

$$\hat{X}_s = \arg\max_{x_s} p\left(x_s|Y\right) \tag{4.7}$$

To evaluate the MPM estimator of $X$, the same Gibbs sampler is used as in the S-Step of SEM, that is another S-step is run with the final parameter estimates. However, instead of using the generated samples for an E-Step, they are used to evaluate (4.8). This is done because the explicit evaluation of (4.7) is intractable for the input sizes involved.

$$\hat{X}_s \approx \arg\max_{x_s} \frac{1}{L} \sum_{l=1}^{L} \delta(X_s^{(l)} = x_s) \tag{4.8}$$

It is worth noting that the MPM estimator of $X$, that is the set of MPM estimators for each $X_s$, is different from the maximum a posteriori (MAP) estimate of $X$, though the two are defined similarly. The MPM estimator was chosen because it minimizes the number of misclassified elements in $X$ [17].

# 5. MANAGEMENT ZONE ESTIMATION RESULTS

## 5.1 Real Yield Data

### 5.1.1 Corn Yield Results

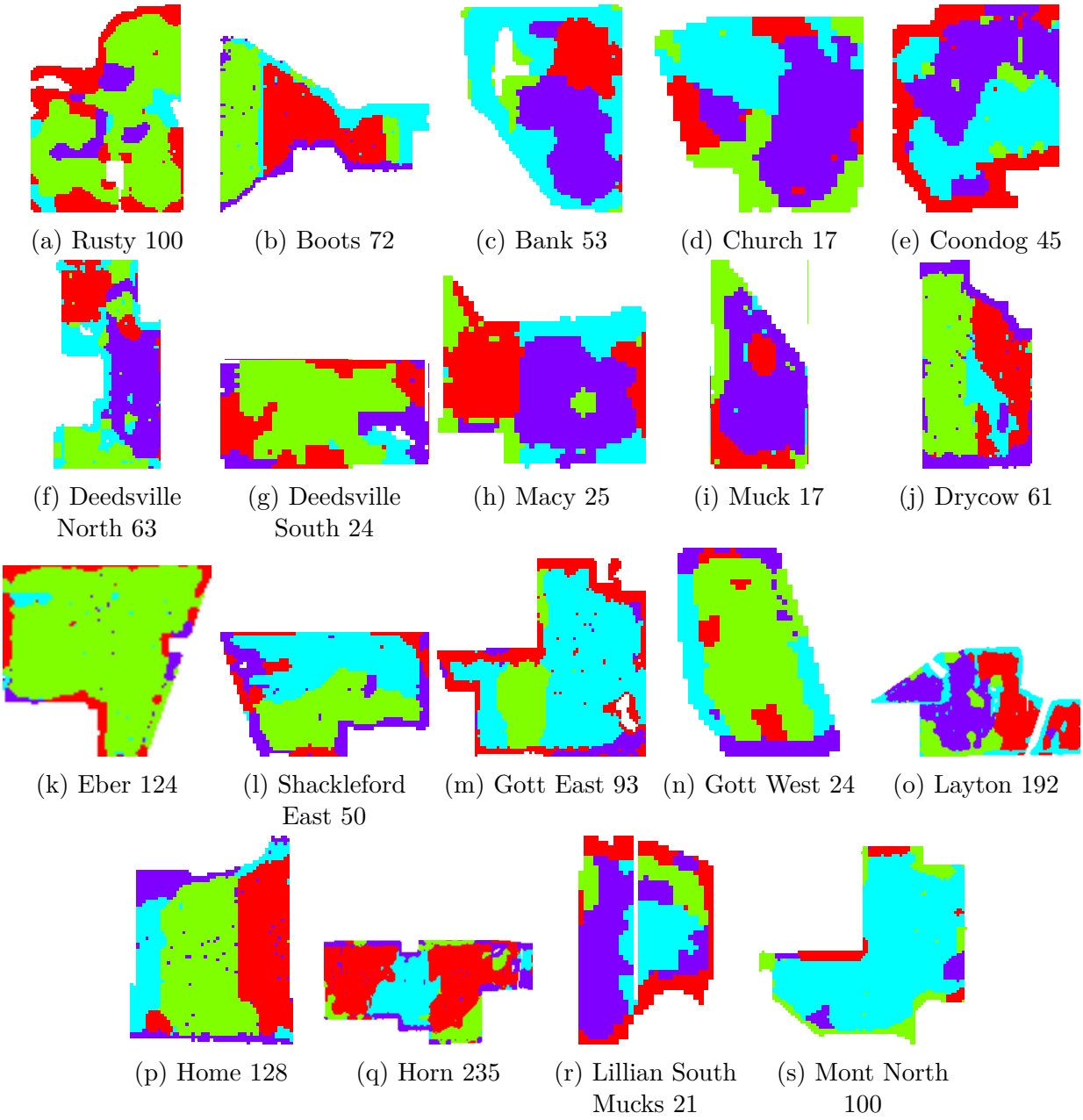
The algorithm was run on the pre-processed corn yield data for each of the fields shown in Fig. 2.1. All of the fields were run once for each value of $K$ from 2 to 10, and each field had at least 2 years of corn harvest data (i.e., $P \geq 2$). The resulting management zone assignments for $K = 4$ are shown in Fig. 5.1, and the standard deviations of the yields for each field and each year of yield data for delineations with 3, 4, and 5 zones respectively are shown in Tables 5.1 to 5.3.

#### 5.1.1.1 Resulting Management Zones

One of the first observations from the results is that the edges of fields tend to be assigned to different management zones than the interiors of fields. This is a positive result because these edges (or "end rows") are known to yield differently and need different management than the interior (due in part to different treatment). The model and algorithm successfully determined this without being given prior knowledge of it.

More importantly, the algorithm does not always assign zones exclusively to the edges. In Fig. 5.1o, the algorithm surrounded the triangular section in the upper left of the field with the "end row" zone. The farmer of this field confirmed that this section is in fact farmed as a separate field from the rest of it and there are end rows there. Conversely, in Fig. 5.1p the algorithm assigned the top and bottom edges as "end row" but not the right and left edges. There is actually more of this field on the right and left but those data had to be excluded because they were missing for some years, while the top and bottom are the real edges of the field.

A shortcoming of the approach that is made evident by these results is not being able to assign a management zone to a location that has missing data for any of the years involved. While some of these holes are actually part of the shape of the field, for example the hole toward the bottom of the field in Fig. 5.1a is a house, other regions should have been included



(a) Rusty 100 (b) Boots 72 (c) Bank 53 (d) Church 17 (e) Coondog 45

(f) Deedsville North 63 (g) Deedsville South 24 (h) Macy 25 (i) Muck 17 (j) Drycow 61

(k) Eber 124 (l) Shackleford East 50 (m) Gott East 93 (n) Gott West 24 (o) Layton 192

(p) Home 128 (q) Horn 235 (r) Lillian South Mucks 21 (s) Mont North 100

**Figure 5.1.** Pictured are the output segmentations resulting from running the presented SEM based algorithm on the real multi-year yield data of each field. Each field was run with $K = 4$. The different colors correspond to different management zone assignments, with white corresponding to no zone assignment.

and segmented. There is one such unassigned region toward the lower right corner of the field in Fig. 5.1m. While that region had yield data in some of the years' data, it did not have data for all years and thus the algorithm could not segment it.

#### 5.1.1.2 Variance Reduction

Shown in Tables 5.1 to 5.3 are the standard deviations of the yields for each field and each year of yield data for delineations with 3, 4, and 5 zones respectively. The total column is the standard deviation of all the recorded yields for that field in that year. The other columns are the standard deviations of the yields grouped by the YPZ to which they were assigned. The highlighted cell in each row is the end row zone for the given field, found by visual inspection of the delineated zones. Zone standard deviations that are not below the corresponding yearly standard deviation are indicated in bold.

Most of the zone standard deviations are below the corresponding total yield standard deviation. Some of the rows have a zone with a higher standard deviation than the total yield, but for the majority of fields this corresponds to the end rows. The zone standard deviations indicated in bold are the one that are greater than their corresponding total yield standard deviation. Out of the 105 trials shown in the tables, only 20 had zones with standard deviations that were not below the total yield standard deviation and were also not end row zones. This means that 81.0% of the time, only our end row zones did not show a reduction in variance. The end rows are expected to be highly variable in terms of yield productivity, owing to highly variable treatment, so this seems a reasonable result.

The end rows indicated in Tables 5.1 to 5.3 were identified visually, since the algorithm does not explicitly differentiate them from other zones. For example, one delineation result is shown in Fig. 5.2. Look at this delineation it can be seen that the green management zone corresponds to the end rows. From the legend, it can be seen that the green zone is zone 1. Based on this, the corresponding column in Table 5.2 was highlighted for the rows corresponding to the field Gott East 93.

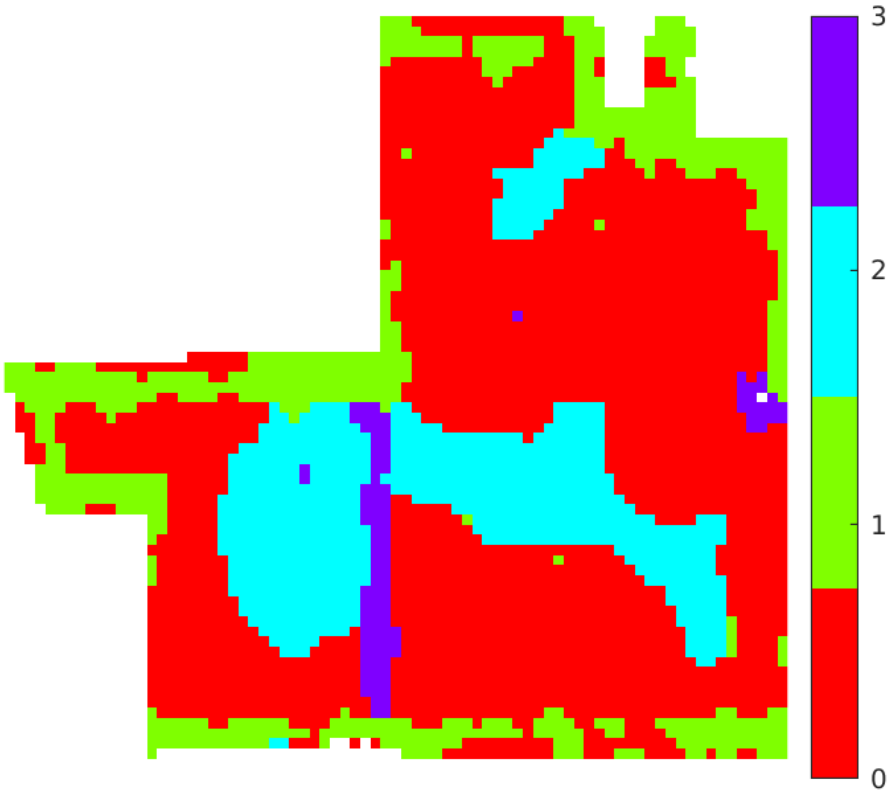



**Figure 5.2.** Management zone delineation for the field Gott East 93 for $K = 4$ with a legend showing the number label assigned to each zone.

**Table 5.1.** Standard deviations of yields for each field in each year when delineated for $K = 3$

| | Total | Zone 0 | Zone 1 | Zone 2 |
|---|---|---|---|---|
| Rusty 100: 2010 | 36.9 | 31.2 | 26.8 | **41.6** |
| Rusty 100: 2013 | 41.7 | 25.1 | 26.7 | **53.9** |
| Bank 53: 2008 | 34.5 | **46.3** | 18.9 | 16.8 |
| Bank 53: 2010 | 44.5 | **57.8** | 22.4 | 17.8 |
| Coondog 45: 2010 | 38.1 | **40.3** | 30.8 | 17.1 |
| Coondog 45: 2011 | 35.1 | **48.7** | 11.8 | 12.6 |
| Deedsville North 63: 2008 | 36.9 | **42.8** | 19.1 | 32.4 |
| Deedsville North 63: 2010 | 45.8 | **47.0** | 17.4 | 20.5 |
| Deedsville South 24: 2008 | 40.3 | 20.4 | 35.7 | 34.7 |
| Deedsville South 24: 2010 | 40.1 | 24.3 | **47.9** | 32.2 |
| Muck 17: 2008 | 40.5 | **45.1** | 13.1 | 19.2 |
| Muck 17: 2010 | 48.1 | **56.9** | 22.9 | 40.9 |
| Drycow 61: 2006 | 29.2 | 18.6 | **47.9** | 10.1 |
| Drycow 61: 2007 | 35.7 | 24.7 | **48.5** | 18.7 |
| Drycow 61: 2008 | 31.3 | 15.4 | **48.5** | 10.3 |
| Drycow 61: 2009 | 41.4 | 31.8 | 40.7 | 17.6 |
| Drycow 61: 2011 | 40.9 | 33.5 | **53.9** | 27.0 |
| Eber 124: 2007 | 33.1 | 20.7 | 32.6 | **46.9** |
| Eber 124: 2009 | 39.5 | 28.6 | 24.2 | **45.4** |
| Shackleford East 50: 2007 | 37.0 | 18.7 | **49.2** | 11.6 |
| Shackleford East 50: 2009 | 43.9 | 22.6 | **47.4** | 20.6 |
| Shackleford East 50: 2010 | 40.2 | 26.2 | **48.7** | 29.1 |
| Gott East 93: 2007 | 26.1 | 10.9 | **40.7** | 7.32 |
| Gott East 93: 2009 | 30.8 | 19.1 | **40.3** | 22.9 |
| Gott East 93: 2011 | 33.9 | 19.4 | **51.3** | 17.7 |
| Gott West 24: 2007 | 33.0 | **42.1** | 15.5 | 15.5 |
| Gott West 24: 2009 | 41.4 | **41.9** | 11.1 | 28.8 |
| Gott West 24: 2011 | 32.2 | **41.8** | 17.8 | 12.7 |
| Layton 192: 2007 | 30.5 | 14.3 | **39.6** | 18.0 |
| Layton 192: 2008 | 30.2 | 15.8 | **37.7** | 18.9 |
| Layton 192: 2009 | 30.8 | 18.7 | **36.1** | 21.6 |
| Home 128: 2008 | 29.8 | **40.1** | 14.5 | 12.5 |
| Home 128: 2009 | 49.3 | **56.7** | 24.5 | 23.1 |
| Lillian South Mucks 21: 2012 | 74.3 | 54.6 | 27.2 | 52.9 |
| Lillian South Mucks 21: 2013 | 52.2 | **62.1** | 22.7 | 28.5 |

**Table 5.2.** Standard deviations of yields for each field in each year when delineated for $K = 4$

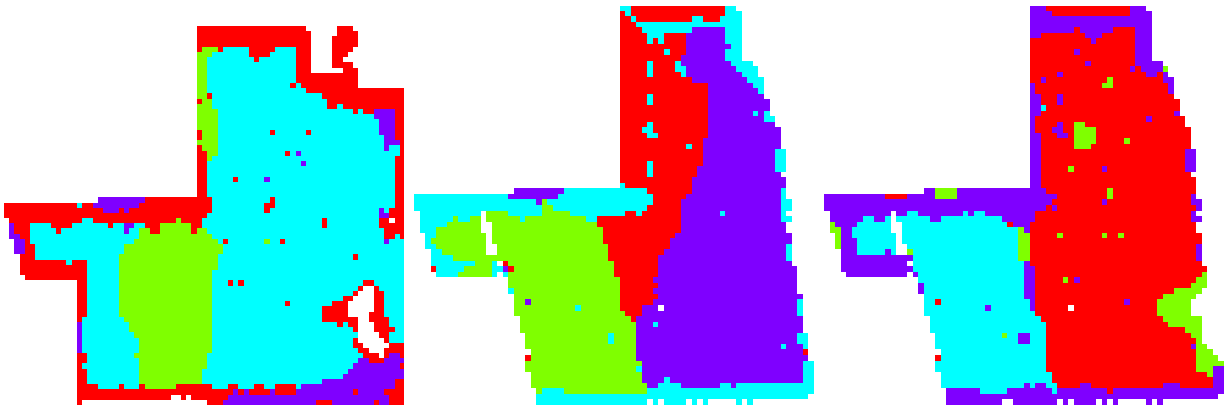
| | Total | Zone 0 | Zone 1 | Zone 2 | Zone 3 |
|---|---|---|---|---|---|
| Rusty 100: 2010 | 36.9 | **39.5** | 25.1 | 21.7 | 21.2 |
| Rusty 100: 2013 | 41.7 | **42.9** | 18.0 | 22.7 | 29.8 |
| Bank 53: 2008 | 34.5 | **45.9** | **38.4** | 16.8 | 19.1 |
| Bank 53: 2010 | 44.5 | **57.8** | 42.3 | 17.8 | 22.6 |
| Coondog 45: 2010 | 38.1 | **40.4** | 24.9 | 19.3 | 16.7 |
| Coondog 45: 2011 | 35.1 | **48.6** | 13.2 | 11.2 | 12.4 |
| Deedsville North 63: 2008 | 36.9 | 17.5 | 19.9 | **38.9** | 24.5 |
| Deedsville North 63: 2010 | 45.8 | 19.2 | 17.0 | **53.7** | 40.3 |
| Deedsville South 24: 2008 | 40.3 | **40.3** | 20.4 | 18.5 | 25.6 |
| Deedsville South 24: 2010 | 40.1 | **40.9** | 23.2 | **42.2** | **54.8** |
| Muck 17: 2008 | 40.5 | **58.1** | 13.3 | 9.76 | 24.0 |
| Muck 17: 2010 | 48.1 | **52.8** | 22.7 | 31.0 | 37.0 |
| Drycow 61: 2006 | 29.2 | 17.2 | 10.8 | 17.9 | **48.9** |
| Drycow 61: 2007 | 35.7 | 23.4 | 18.10 | 18.4 | **46.8** |
| Drycow 61: 2008 | 31.3 | 12.10 | 10.1 | 25.8 | **48.8** |
| Drycow 61: 2009 | 41.4 | 25.2 | 18.1 | **41.5** | 40.2 |
| Drycow 61: 2011 | 40.9 | 25.6 | 27.1 | **44.5** | **53.9** |
| Eber 124: 2007 | 33.10 | **47.1** | 18.9 | 29.9 | 16.5 |
| Eber 124: 2009 | 39.5 | **44.6** | 26.9 | 23.8 | 21.5 |
| Shackleford East 50: 2007 | 37.0 | **49.1** | 11.0 | 18.1 | 13.5 |
| Shackleford East 50: 2009 | 43.9 | **49.9** | 20.1 | 22.8 | 25.8 |
| Shackleford East 50: 2010 | 40.2 | **48.1** | 18.6 | 25.8 | 32.7 |
| Gott East 93: 2007 | 26.10 | 10.9 | **41.5** | 6.75 | **35.0** |
| Gott East 93: 2009 | 30.8 | 20.1 | **39.2** | 22.10 | 20.0 |
| Gott East 93: 2011 | 33.9 | 19.3 | **50.6** | 18.2 | 15.6 |
| Gott West 24: 2007 | 33.0 | **41.7** | 8.27 | 13.9 | 15.5 |
| Gott West 24: 2009 | 41.4 | **43.1** | 28.4 | 22.3 | 11.2 |
| Gott West 24: 2011 | 32.2 | **40.7** | 10.8 | 12.7 | 18.1 |
| Layton 192: 2007 | 30.5 | 17.7 | 20.9 | **43.6** | 10.3 |
| Layton 192: 2008 | 30.2 | 18.8 | 18.9 | **38.9** | 14.8 |
| Layton 192: 2009 | 30.8 | 25.3 | 18.0 | **35.9** | 17.0 |
| Home 128: 2008 | 29.8 | 14.4 | 14.5 | **41.9** | 11.1 |
| Home 128: 2009 | 49.3 | 24.3 | 31.3 | **57.5** | 19.2 |
| Lillian South Mucks 21: 2012 | 74.3 | 60.3 | 45.3 | 35.4 | 25.9 |
| Lillian South Mucks 21: 2013 | 52.2 | **66.9** | 26.9 | 25.2 | 26.1 |

**Table 5.3.** Standard deviations of yields for each field in each year when delineated for $K = 5$

| | Total | Zone 0 | Zone 1 | Zone 2 | Zone 3 | Zone 4 |
|---|---|---|---|---|---|---|
| Rusty 100: 2010 | 36.9 | **44.9** | 22.7 | 24.1 | **37.4** | 21.9 |
| Rusty 100: 2013 | 41.7 | **51.8** | 23.8 | 18.0 | **49.5** | 21.6 |
| Bank 53: 2008 | 34.5 | **45.6** | 21.5 | 18.1 | **51.5** | 16.7 |
| Bank 53: 2010 | 44.5 | **49.9** | 26.3 | 22.4 | **50.2** | 17.5 |
| Coondog 45: 2010 | 38.1 | 16.5 | 30.4 | 19.3 | 27.5 | 25.3 |
| Coondog 45: 2011 | 35.1 | 12.4 | 32.9 | 10.9 | 13.6 | **37.9** |
| Deedsville North 63: 2008 | 36.9 | 17.1 | 34.9 | 24.7 | 31.4 | 14.6 |
| Deedsville North 63: 2010 | 45.8 | 16.1 | **54.9** | 40.8 | 36.6 | 18.7 |
| Deedsville South 24: 2008 | 40.3 | **47.0** | 31.7 | 16.0 | 14.1 | 32.9 |
| Deedsville South 24: 2010 | 40.1 | **43.6** | 32.1 | 18.2 | 28.3 | **54.2** |
| Muck 17: 2008 | 40.5 | 14.8 | 24.2 | 21.4 | **56.1** | 13.5 |
| Muck 17: 2010 | 48.1 | 23.8 | 37.5 | 22.8 | 45.2 | 22.9 |
| Drycow 61: 2006 | 29.2 | 25.1 | **53.5** | 10.2 | 17.7 | 15.5 |
| Drycow 61: 2007 | 35.7 | 28.5 | **53.1** | 17.1 | 19.6 | 20.1 |
| Drycow 61: 2008 | 31.3 | 17.3 | **56.8** | 10.1 | 26.4 | 12.1 |
| Drycow 61: 2009 | 41.4 | 32.9 | **39.5** | 18.5 | 40.8 | 24.9 |
| Drycow 61: 2011 | 40.9 | **41.1** | **60.3** | 26.5 | **44.2** | 23.5 |
| Eber 124: 2007 | 33.1 | **48.8** | 20.2 | 16.8 | 30.0 | 17.5 |
| Eber 124: 2009 | 39.5 | **45.5** | 22.7 | 19.7 | 23.5 | 25.6 |
| Shackleford East 50: 2007 | 37.0 | **48.4** | 13.5 | 11.0 | 18.1 | **56.1** |
| Shackleford East 50: 2009 | 43.9 | **48.6** | 25.8 | 20.1 | 22.9 | **64.9** |
| Shackleford East 50: 2010 | 40.2 | **46.1** | 31.4 | 18.8 | 25.9 | **75.9** |
| Gott East 93: 2007 | 26.1 | 13.7 | 9.32 | **35.0** | **43.7** | 8.86 |
| Gott East 93: 2009 | 30.8 | 25.6 | 13.4 | **32.0** | **40.0** | 20.5 |
| Gott East 93: 2011 | 33.9 | 23.1 | 17.3 | 26.8 | **54.6** | 16.8 |
| Gott West 24: 2007 | 33.0 | **34.2** | 14.1 | **52.1** | 15.1 | 8.56 |
| Gott West 24: 2009 | 41.4 | 36.9 | 22.1 | **39.9** | 11.3 | 27.1 |
| Gott West 24: 2011 | 32.2 | 23.7 | 12.7 | **51.2** | 18.2 | 9.91 |
| Layton 192: 2007 | 30.5 | 17.8 | 20.1 | **44.1** | 10.3 | 15.1 |
| Layton 192: 2008 | 30.2 | 18.7 | 19.3 | **40.6** | 14.0 | 21.0 |
| Layton 192: 2009 | 30.8 | 24.4 | 17.5 | **36.1** | 15.2 | 29.0 |
| Home 128: 2008 | 29.8 | 14.6 | 12.4 | 11.4 | 13.2 | **42.9** |
| Home 128: 2009 | 49.3 | 23.6 | 21.8 | 12.8 | 31.4 | **57.1** |
| Lillian South Mucks 21: 2012 | 74.3 | 62.1 | 36.1 | 35.2 | 33.7 | 51.7 |
| Lillian South Mucks 21: 2013 | 52.2 | 36.2 | 9.26 | 27.1 | 41.6 | **54.2** |

### 5.1.2 Corn and Soybean Yield Results

In order to investigate its performance on different crop yield data, the algorithm was run on the pre-processed soybean yield data for the field Gott East 93, in addition to its corn yield data. This field was chosen because it has 3 years of soybean yield data in addition to its 4 years of corn yield data. Three runs of the algorithm were performed for each value of $K$ from 2 to 10, one using only the corn yields ($P = 4$), one using only the soybean yields ($P = 3$), and one using both the corn and soybean yields ($P = 7$). The resulting management zone assignments for $K = 4$ are shown in Fig. 5.3.



(a) Corn Yields (b) Soybean Yields (c) Both Yields

**Figure 5.3.** Pictured are the management zone delineation for the field Gott East 93 using only corn yields, only soybean yields, and both corn and soybean yields, respectively. The pictured delineations are for $K = 4$.

The first observation is that the zones delineated for corn differ from those delineated for soybeans. This seems reasonable since different crops have different yield characteristics. In practice, it might be useful to run separate delineations for each crop to produce crop-dependent management zones rather than delineating based on multiple crops at once, but the algorithm runs just as well either way.

The previously mentioned shortcoming of the algorithm not being able to assign a management zone to a location that has any missing data, is made more evident by these results. In the corn delineations in Fig. 5.3a there is a hole in the data near the lower right corner, and in the soybean delineations in Fig. 5.3b some of the right edge of the field has no data.

When both corn and soybean data were used, neither the hole nor the right edge of the field could be assigned to a management zone, as seen in Fig. 5.3c.

## 5.2 Simulated Yield Data Based on Real Yield Maps

Since the real data have no ground truth for evaluating the output, simulations were performed. The management zone assignments and distribution parameters from running the algorithm on the real data were used to simulate new yield observations. This simulation was done by drawing sample yields $Y$ according to the conditional distribution defined by (3.4), using the $X$s and $\theta$s from Section 5.1.1. One such simulation was run for each field and each value of $K$ from 2 to 10. These simulations produced $Y$s for which the corresponding $X$s were known, something missing for the real data.

### 5.2.1 Simulated Performance of the Algorithm

Now that there was "ground truth", the performance of the algorithm could be measured using a distance of the output $\hat{X}$ from the correct $X$. The Rand distance [18] was used as an error metric for comparing the output management zone segmentations to the true segmentation. A Rand distance of 0 corresponds to two segmentations with a total agreement, while a Rand distance of 1 corresponds to the least agreement possible.

For brevity, the Rand distance between the true $X$ and the estimate $\hat{X}$ will be referred to as the Rand error. Table 5.4 shows the Rand error of the presented SEM based algorithm when run on the simulated fields. It also shows the average Rand error by field (leftmost column) and by $K$ (bottom row). Additionally, the histogram of these Rand errors is shown in Fig. 5.4.

While the majority of the observed Rand errors were below 0.25, a few were quite large. All of these large errors correspond to running at orders $K \geq 7$ for fields less than 25 acres in size. As the field size decreases and the order increases there are fewer observations for estimating the parameters of each management zone, thus resulting in poorer estimates. While performance, in this case, is indeed poor, having so many management zones in such small fields is not likely to be reasonable from a precision agriculture perspective.

**Table 5.4.** This table shows the simulation Rand errors of the output of the presented SEM based algorithm when run on simulated fields based on real fields, as described in Section 5.2. Also, the leftmost column show the Rand error averaged by field and the bottom row shows the Rand error averaged by order, $K$.

| | $K = 2$ | $K = 3$ | $K = 4$ | $K = 5$ | $K = 6$ | $K = 7$ | $K = 8$ | $K = 9$ | $K = 10$ | Average |
|---|---|---|---|---|---|---|---|---|---|---|
| Rusty 100 | 0.0514 | 0.0364 | 0.0278 | 0.1838 | 0.0330 | 0.0403 | 0.0271 | 0.1600 | 0.0281 | 0.0653 |
| Boots 72 | 0.0294 | 0.0203 | 0.0196 | 0.0203 | 0.0127 | 0.0758 | 0.0305 | 0.0230 | 0.0207 | 0.0280 |
| Bank 53 | 0.0156 | 0.0142 | 0.0215 | 0.0232 | 0.0151 | 0.0395 | 0.0441 | 0.0433 | 0.0145 | 0.0257 |
| Church 17 | 0.0468 | 0.0297 | 0.0254 | 0.0204 | 0.0317 | 0.0367 | 0.0253 | 0.0255 | 0.0465 | 0.0320 |
| Coondog 45 | 0.0147 | 0.0643 | 0.0247 | 0.0282 | 0.0445 | 0.0335 | 0.0385 | 0.0208 | 0.0537 | 0.0359 |
| Deedsville North 63 | 0.0168 | 0.0394 | 0.0247 | 0.0213 | 0.0276 | 0.0183 | 0.0244 | 0.0392 | 0.0863 | 0.0331 |
| Deedsville South 24 | 0.0205 | 0.0216 | 0.0669 | 0.0752 | 0.1218 | 0.0524 | 0.1767 | 0.1091 | 0.0371 | 0.0757 |
| Macy 25 | 0.0468 | 0.0718 | 0.1104 | 0.0416 | 0.0669 | 0.0315 | 0.0719 | 0.0353 | 0.0206 | 0.0552 |
| Muck 17 | 0.0247 | 0.0132 | 0.0168 | 0.0260 | 0.0220 | 0.4644 | 0.0299 | 0.7813 | 0.3403 | 0.1910 |
| Drycow 61 | 0.0128 | 0.0114 | 0.0228 | 0.0800 | 0.0151 | 0.0209 | 0.0376 | 0.0815 | 0.0839 | 0.0407 |
| Eber 124 | 0.0128 | 0.0534 | 0.0248 | 0.1329 | 0.0590 | 0.0767 | 0.0408 | 0.0336 | 0.0531 | 0.0541 |
| Shackleford East 50 | 0.0120 | 0.0151 | 0.0159 | 0.0151 | 0.0317 | 0.0394 | 0.0252 | 0.0429 | 0.0529 | 0.0278 |
| Gott East 93 | 0.0168 | 0.0660 | 0.2167 | 0.0361 | 0.0583 | 0.0766 | 0.0337 | 0.0274 | 0.0271 | 0.0621 |
| Gott West 24 | 0.0230 | 0.0497 | 0.0995 | 0.0291 | 0.0214 | 0.0734 | 0.0370 | 0.7044 | 0.0375 | 0.1194 |
| Layton 192 | 0.0364 | 0.0239 | 0.0162 | 0.0198 | 0.0198 | 0.0184 | 0.0186 | 0.0598 | 0.0149 | 0.0253 |
| Home 128 | 0.0170 | 0.0155 | 0.0226 | 0.0177 | 0.0224 | 0.0235 | 0.0173 | 0.0276 | 0.0156 | 0.0199 |
| Horn 235 | 0.0415 | 0.0196 | 0.0239 | 0.0207 | 0.0199 | 0.0231 | 0.0176 | 0.0229 | 0.0213 | 0.0234 |
| Lillian South Mucks 21 | 0.0157 | 0.0305 | 0.1078 | 0.1180 | 0.0192 | 0.0509 | 0.0641 | 0.0598 | 0.0442 | 0.0567 |
| Mont North 100 | 0.0136 | 0.0186 | 0.0802 | 0.1260 | 0.0430 | 0.0724 | 0.0364 | 0.0382 | 0.1951 | 0.0693 |
| Average | 0.0246 | 0.0323 | 0.0510 | 0.0545 | 0.0361 | 0.0667 | 0.0419 | 0.1229 | 0.0628 | 0.0548 |

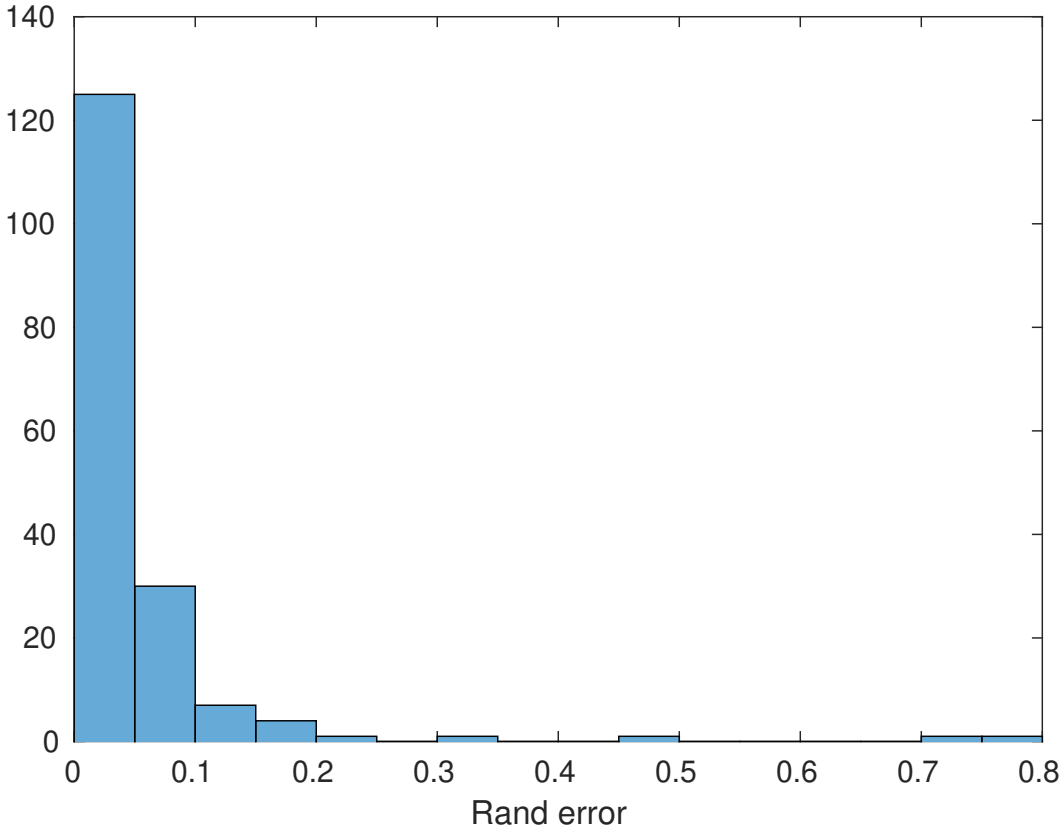



**Figure 5.4.** This histogram shows the occurrences of the Rand errors of the output of the presented SEM based algorithm when run on simulated fields based on real fields, as described in Section 5.2.

### 5.2.2 Comparison to State-of-the-Art

The simulated fields were also run though another management zone delineation algorithm, Management Zone Analyst (MZA)[1] [19]. MZA is commonly used for delineating management zones, particularly from multivariate data [7]. The MZA algorithm was run using Mahalanobis distance as the distance metric because the yields are not assumed to be statistically independent, and with 10,000 as the maximum number of iterations to allow more time for convergence than the default 300 iterations. Also, values of the fuzziness exponent from 1.1 to 5 in increments of 0.1 were tried and the one producing the smallest Rand error (see Section 5.2.1) was used (for each field at each $K$).

#### 5.2.2.1 Multi-Field Simulation

Figure 5.5 shows the Rand error averaged over the 19 fields for both the presented SEM based algorithm and MZA, as a function of $K$. Two versions of MZA are shown, one where the UTM $x$ and $y$ coordinates of each observation were included in the input, and one where the UTM coordinates were not utilized. This was done because both ways of running MZA are used in literature [7], [19].

In these simulations, the presented SEM algorithm outperformed the MZA algorithm for every value of $K$ tested. The gap in the two algorithms' performances seems to decrease as $K$ increases, but $K$s larger than those tested are probably not realistic for actual fields. For the sorts of fields simulated, the proposed algorithm consistently outperformed the state-of-the-art.

#### 5.2.2.2 Multi-Crop Simulation

Another simulated comparison to MZA was performed, as in Section 5.2.2.1 but using the corn and soybean results from Section 5.1.2. Since this time the data were only from one of the fields, Gott East 93, 13 separate realizations of simulated $Y$s were generated for each $K$, and each set of crop yields (corn yields only, soybean yields only, and both corn and

[1]↑The actual MZA program was not used, but the algorithm was implemented as described in [19], using MATLAB.

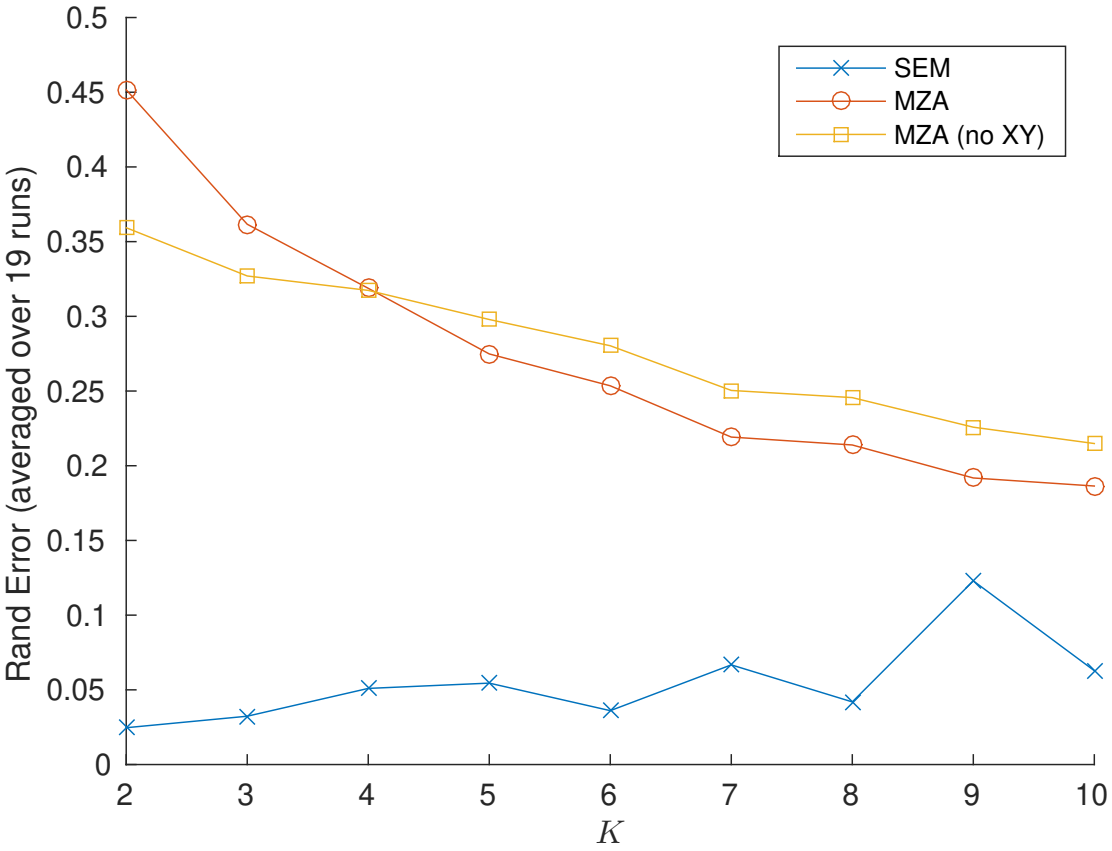



**Figure 5.5.** This figure shows the performances of the presented SEM based algorithm and the MZA algorithm. For each value of $K$ both algorithms were run on the same simulated field, simulated as described in Section 5.2. The MZA algorithm was run twice for each field, once with UTM $x$ and $y$ included in the observations, and once without utilizing the UTM coordinates. The error metric, referred to as Rand error, was computed as the Rand distance between the true $X$ used for simulation and the $\hat{X}$ output by the algorithm.

soybean yields). Figure 5.6 shows the Rand error averaged over the 13 realizations for the presented algorithm and the two methods of running the MZA algorithm.

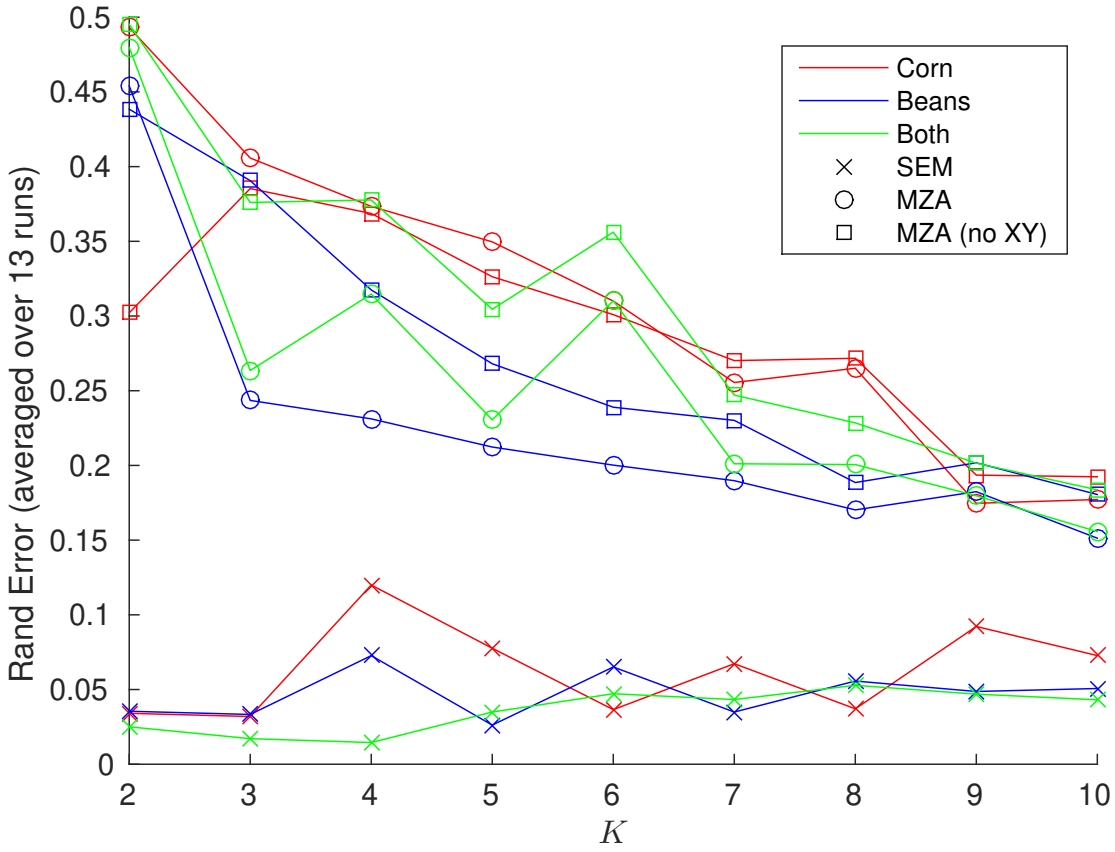



**Figure 5.6.** This figure shows the performances of the SEM algorithm presented by this paper and the MZA algorithm. For each value of $K$ both algorithms were run on the same simulated field, simulated as described in Section 5.2. The error metric, referred to as Rand error, was computed as the Rand distance between the true $X$ used for simulation and the $\hat{X}$ output by the algorithm.

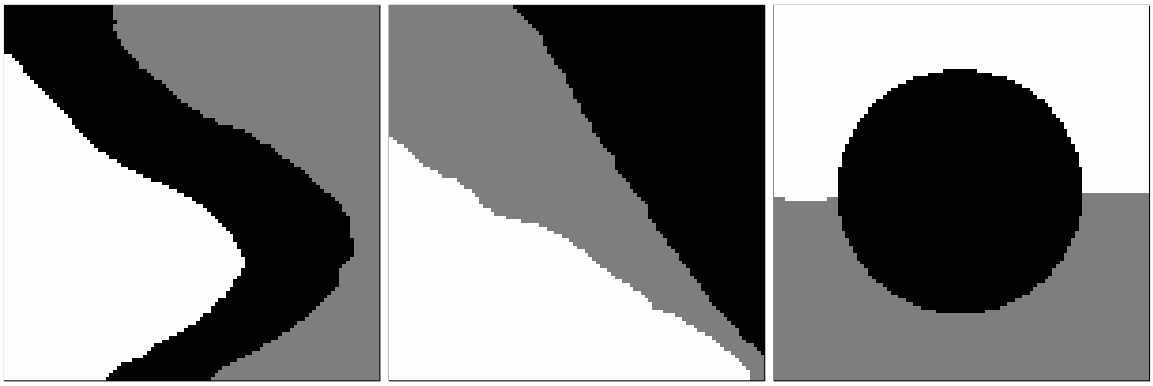
As in the previous simulation, the presented algorithm outperformed the MZA algorithm for every $K$ and every crop tested. An interesting observation is that all three methods tend to perform with soybean yields than with corn yields.

## 5.3 Purely Simulated Yield Data

However, it was desirable to analyze the impact of an increasing number of years of available yield data. The maximum number of years of yield data in our real dataset is 5. Moreover, the ground truth for management zone labels is unknown to us in a real physical cornfield. Therefore, the question was addressed with a simulation-based experiment.

### 5.3.1 Simulating Yield Data for Arbitrary Years

In order to analyze the impact of an increasing number of years of data on our delineations, we created synthetic management zone assignments for which we could simulate a variable number of years of yield maps. Synthetic zone assignments were used because no ground truth zones were available. The zone assignments used in the simulation are shown in Fig. 5.7.



(a) Synthetic field 1 (b) Synthetic field 2 (c) Synthetic field 3

**Figure 5.7.** Management zone assignments for fields in multiple year experiment.

Once we had zone assignments, we could then generate yearly yield maps from these zones. By looking at the distributions of yields across all the fields and all the years in our dataset, it was decided to use a mean of 180 bu/ac and a standard deviation of 30 bu/ac as a representative distribution of corn yields. Total yield was assumed to be normally distributed, and the total yield was divided into 3 equally likely classes. These classes were also assumed to be normally distributed, and were spaced such that the means and variances of the total yield were preserved. These distributions are illustrated in Fig. 5.8. The 3 class

distributions were given equal standard deviations of $\frac{\sigma_{total}}{1.8} = 16.6$, this ratio of total variation to in-zone variation seemed reasonable based on the results of our delineations on real data.

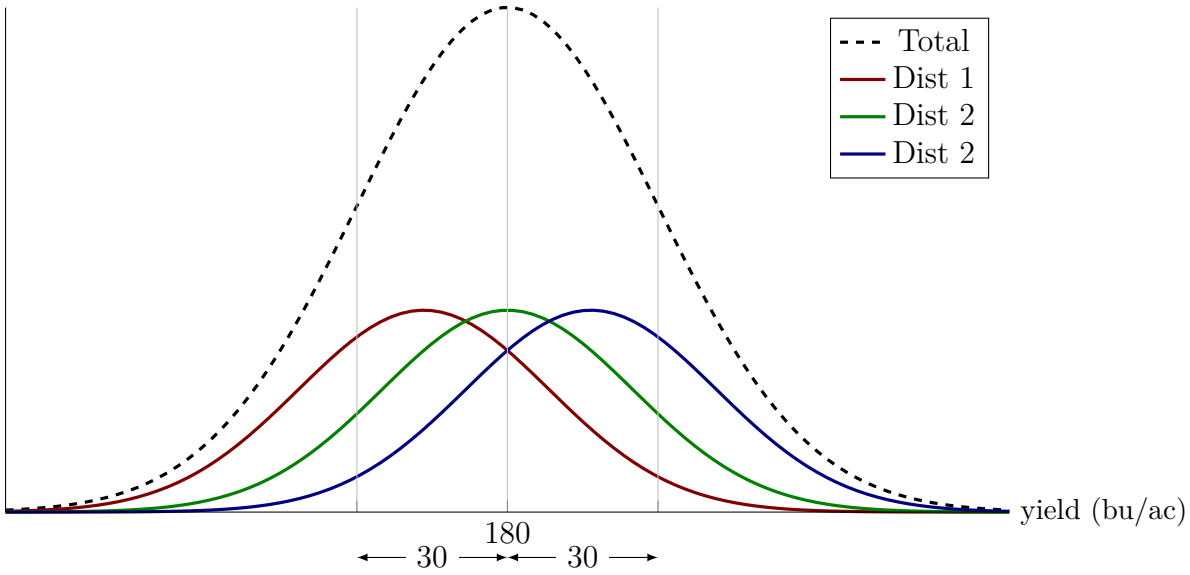



**Figure 5.8.** Illustration of dividing total yield distribution into 3 equally likely zone distributions. The division is done such that the total mean and variance are preserved.

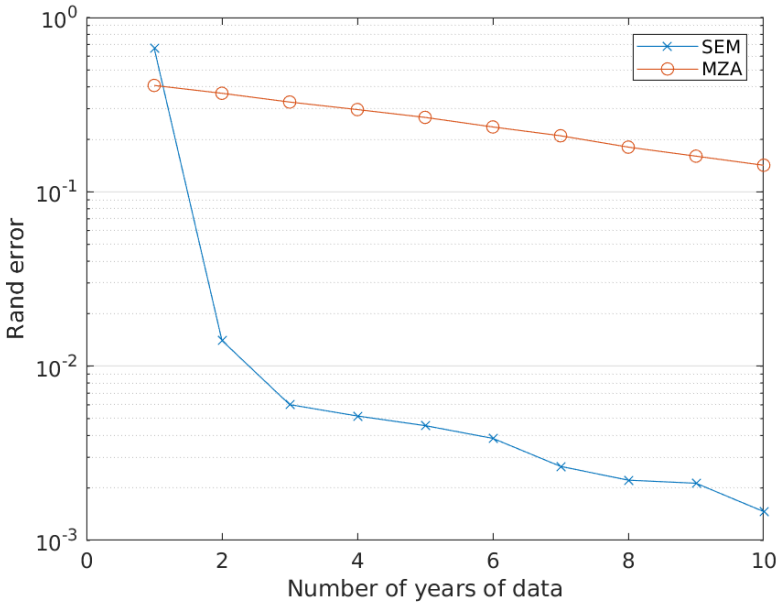
Using the zone assignments of Fig. 5.7 and the yield distributions of Fig. 5.8, yield maps were randomly generated. To generate a yield map for a given set of zones, the three yield distributions were randomly assigned to the three management zones. In this way, we do not artificially prefer one zone by assigning it the higher productivity distribution every year. Then, for each pixel in a zone assignment, a pseudo-random realization of a normally distributed yield was drawn according to the distribution that had been assigned to that zone. In order to create $P$ yield maps, the above procedure was repeated $P$ times. In this way we had multiple yield maps corresponding to fixed zone assignments, but where each zone's behavior varied year-to-year.

### 5.3.2 Simulation Results

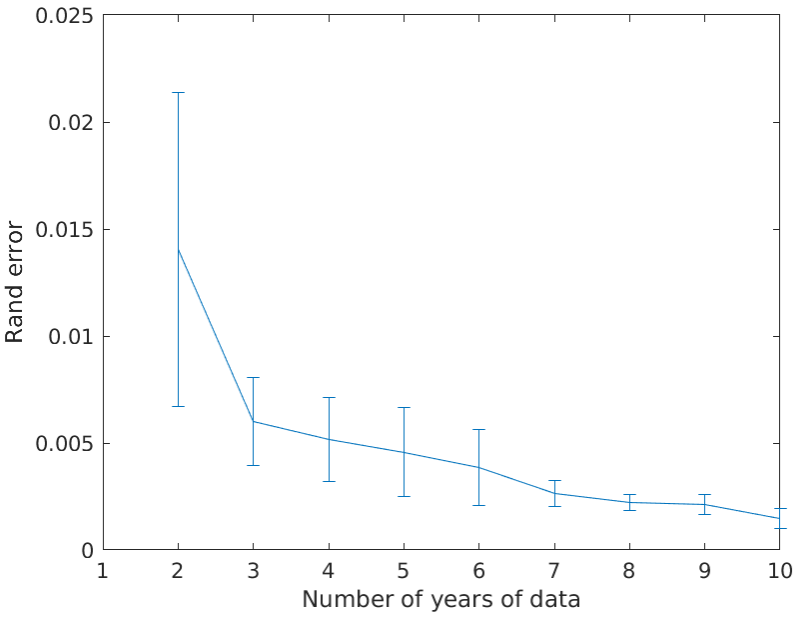
The performance of the proposed delineation method and MZA on these simulated yield maps is shown in Fig. 5.9. The metric used is the Rand error (as described in the manuscript,

i.e., the Rand distance between the output delineations and the simulated delineation from Fig. 5.7), averaged over the three fields used.

The results of the simulation are that the errors in the delineations decrease as more years' yields are used. When only a single year's data are used, MZA performs better than the proposed method. However, the proposed method rapidly converges to very low errors as the number of years of data increases. The average Rand error decreases with $P$, as does the variability of these errors. This result makes sense, as each year's yield map is another observation with which the algorithm can estimate the correct zone assignment for each location. While both MZA and our method perform better the more yield maps are used, the proposed method makes much better use of multiple year's yield maps than MZA.




(a) Error for both the proposed method and MZA.




(b) Error for proposed method when $P \geq 2$.

**Figure 5.9.** Shown are the Rand errors vs number of years of yield data from the simulation experiment to determine the impact of an increasing number of years' data. The values are averaged over the three fields used, and the error bars show the standard deviation of the errors. Errors were calculated for both the proposed method's delineations and delineations from MZA. Both methods are shown in Fig. 5.9a and, since the errors are so much smaller, a zoomed version is shown for $P \geq 2$ is shown in Fig. 5.9b.

# 6. CONCLUSIONS

The proposed model and algorithm can successfully delineate management zones for a field based on multiple years of yield data. They can also delineate based on yield data for multiple crops, both together and separately, without needing any crop-specific calibration. The output delineations are very different for different crops, but this is likely because different crops need different management and not because of a shortcoming of the algorithm or model.

Simulated results show the relative performance of the presented algorithm to the state-of-the-art MZA algorithm. In simulation, the presented algorithm outperforms the MZA algorithm for all orders and all crop types tested. Additionally, simulations showed that the proposed algorithm makes better use of multiple years' data, with it rapidly converging to very low errors as the number of years increases. The relative performances seemed to be converging as the order, $K$, increased, however, since orders higher than those tested are likely not realistic, the results still suggest the presented approach is better than the state-of-the-art for delineating management zones based on multi-year yield data.

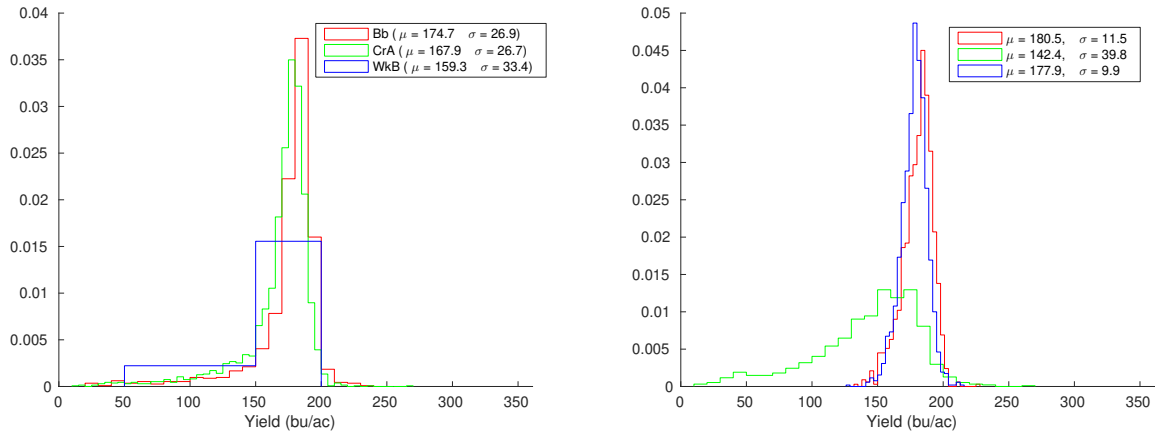
# 7. FUTURE WORK

## 7.1 Investigate Soil Relations

Soils influence crops and are therefore a factor in determining management zones. Aside from clustering based on the observed yields, another method of determining management zones is to use soil maps.

### 7.1.1 Initial Soil Map Unit Comparison

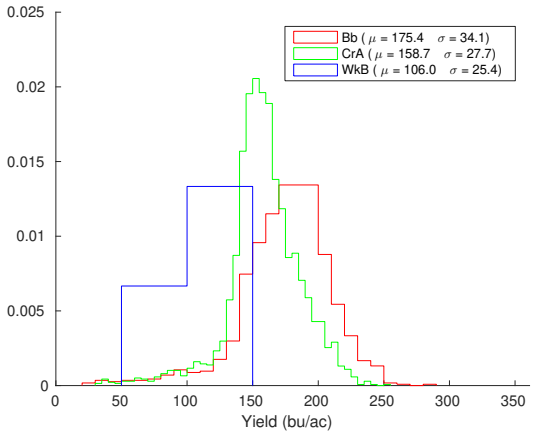
In order to compare the current results and investigate incorporating soils in the model, the SSURGO soil data for the farm fields used were acquired from [20]. As a first look at the soil data, the SSURGO map units were used as management zones as in [21]. In particular, soils for the field Gott East 93 were investigated. Since the field had 3 map units in it, the algorithm results for $K = 3$ were used for comparison. The histograms of yields by map unit and by management zone are shown in Figs. 7.1 to 7.4.



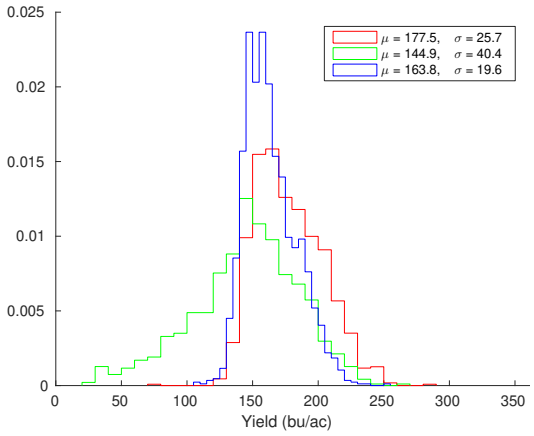

(a) SSURGO map units (b) Delineated zones ($K = 3$)

**Figure 7.1.** Yield histograms for 2007

The yearly distributions of the management zones are more distinct than those of the SSURGO map units. This suggests that the presented method is better than managing by map unit alone. This is likely because map units are defined at too coarse a scale to be

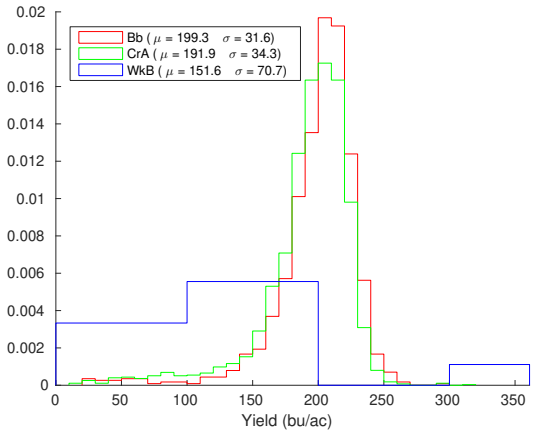



(a) SSURGO map units

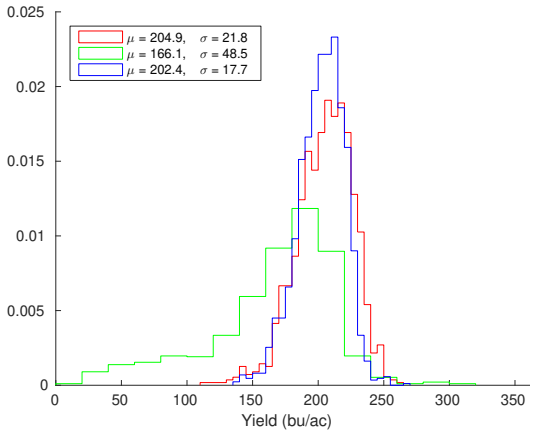



(b) Delineated zones ($K = 3$)

**Figure 7.2.** Yield histograms for 2009

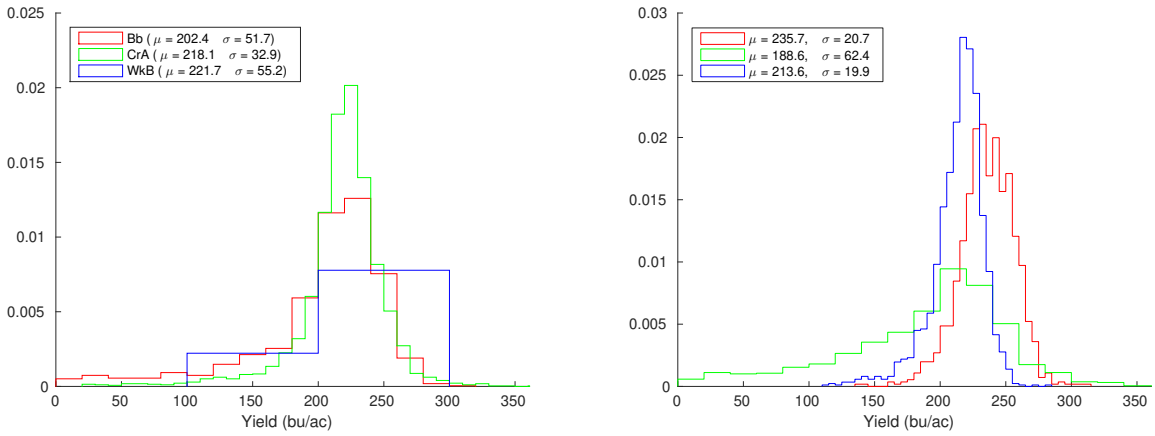



(a) SSURGO map units




(b) Delineated zones ($K = 3$)

**Figure 7.3.** Yield histograms for 2011




(a) SSURGO map units

(b) Delineated zones ($K = 3$)

**Figure 7.4.** Yield histograms for 2013

helpful at the scale of a field. However, finer-scale soil data could be useful in delineating management zones.

### 7.1.2 ERUs

Since the map units seem too coarse for delineating management zones, the next step is to try finer resolution soil data. The environmental response unit (ERU) data is a finer resolution soil data set than SSURGO, which is partly based on [21]. The next step would be to acquire ERU data for the locations in the yield data set. The ERU data could be compared to the delineated management zones, similarly to the SSURGO map units.

In addition to looking at the distributions of yields within the ERUs vs the management zones, a next step would be to compare the reduction in variance of the two as in [21]. Though the algorithm does not necessarily minimize the variance of the management zones, it is still of interest as a performance metric. Since ideally management zones are homogeneous the should have low variance, especially those besides the "end row" zones.

The ERUs could potentially be used in the delineation of management zones, in addition to the yearly yields. They might be useful in determining a suitable number of zones, $K$, for a given field. Also, it would be of interest to try initializing the SEM based on ERUs rather than fuzzy c-means.

## 7.2 Handling Partial Yield Observations

As mentioned, currently the model and algorithm require that each location delineated has a yield observation for all the years of interest. That is, for every location, $s$, in $Y$ all $P$ elements of the corresponding yield vector $Y_s$ must be observed.

However, many of the yield maps had substantial parts of the data missing. This can happen when multiple machines are used to harvest a field but only some of these machines' data are retained. It also happens that some years a field is split in half with one half used to grow one crop and another half used for a different crop. When things such as these happen, the yield map for that year must either be interpolated to fill in the missing sections, or that

year's data can be left out of the delineation entirely. The current algorithm cannot utilize the partial yield map in its delineation.

It would be better if the delineation could be performed on some subset of the full observed $Y$, call it $\tilde{Y}$. For every location $\tilde{Y}$ would have observations for some, but not necessarily all, of the $P$ years. That is,

$$\varnothing \subset \tilde{Y}_s \subseteq Y_s \qquad \forall s \in S. \tag{7.1}$$

This would require deriving new EM update steps for the family of distributions $P(\tilde{Y}, X \mid \theta)$ and the implementation of a Gibbs sampler for sampling from the distribution $P(X \mid \tilde{Y})$. Then the management zones could be estimated with the MPM for the posterior given $\tilde{Y}$, i.e.,

$$\hat{X}_s = \arg\max_{x_s} p\left(x_s \mid \tilde{Y}\right) \qquad \forall s \in S. \tag{7.2}$$

Such an extension would resolve the undelineated regions seen in some of the results. More importantly, it would facilitate simultaneously delineating zones for multiple fields, producing a single set of management zones for a whole farm rather than zones that are unrelated from field to field.

### 7.3 Determining Adequate Convergence

Currently, running the algorithm is somewhat time-consuming. The number of EM iterations being used is likely more than necessary. An initial analysis of the convergence was conducted, and a comparatively large number of iterations was chosen to help ensure convergence.

The metric used for the change in a given parameter from one iteration to the next was

$$D(A_k) \triangleq \frac{\|A_k^{new} - A_k^{old}\|_F}{\|A_k^{old}\|_F}. \tag{7.3}$$

This metric was chosen because it can be thought of as a percent change, under the Frobenius norm. Then the max of this across the $K$ zones, that is,

$$D(A) \triangleq \max_{0 \leq k < K} D(A_k) \tag{7.4}$$

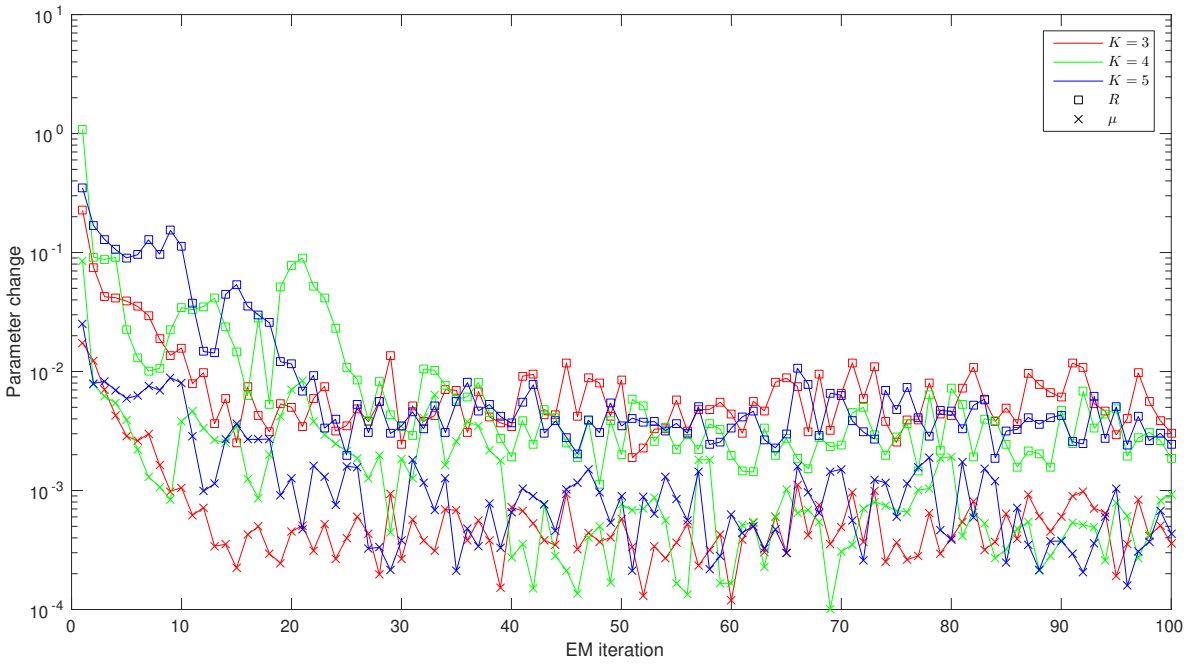
was computed for $R$ and $\mu$. Plots of this change metric for both $R$ and $\mu$ are shown in Figs. 7.5a and 7.5b for 1000 and 10000 Gibbs samples per iteration respectively. Currently, the parameter changes have only been recorded for the field Gott East 93 and only for $3 \leq K \leq 5$. The first step in this future work would be to record these statics for more fields and values of $K$.

Initially, with only 1000 Gibbs samples per iteration, some of the parameter changes continued to fluctuate above 0.01. Therefore, the number of samples was increased to 10000. With the increased number of Gibbs samples, the fluctuations stayed below 0.01 after 30 iterations even for $K = 5$. To be conservative, and because higher $K$s would converge more slowly, 100 iterations were used for all the delineation results.

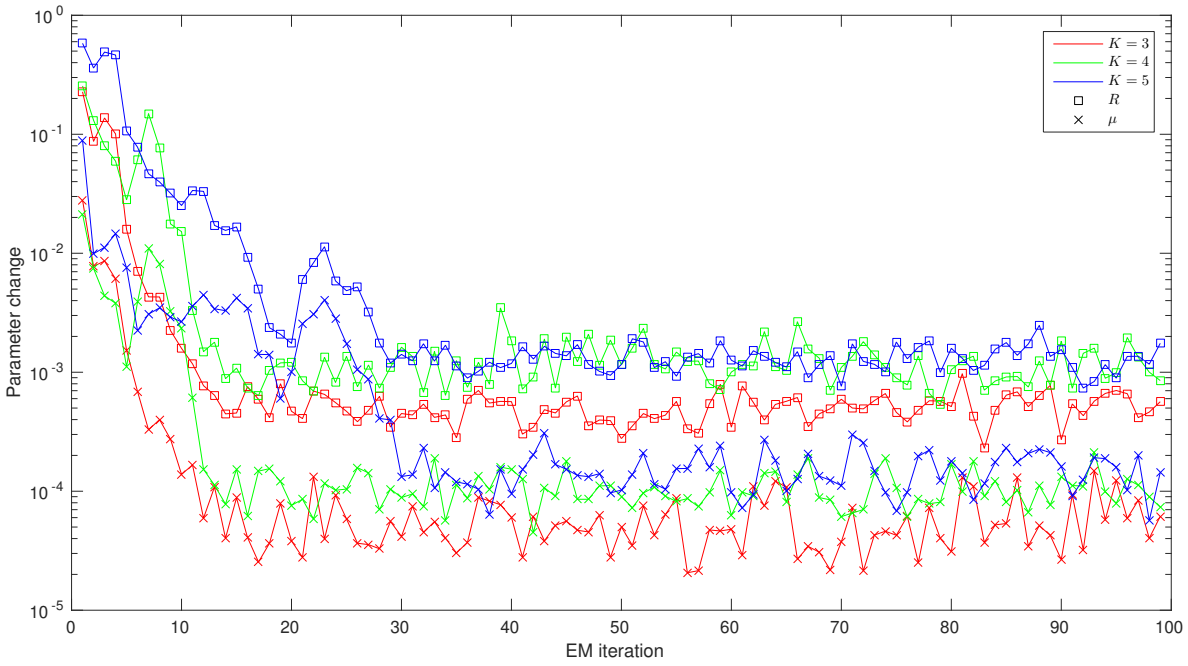
Another potential convergence statistic to look at is the Rand error vs iteration when running on simulated data. This would require rerunning the simulations since only the final outputs were recorded for the initial simulations. Such a metric could not be used as a stopping criterion, since the Rand error is not known except in simulation. However, analyzing the behavior of it in simulation could provide insight into the orders of Gibbs samples and EM iterations needed, and thresholds on parameter convergence might be practical.

### 7.4 Accounting for Variable Rate Inputs

The current model does not account for varying inputs such as plant population or fertilizer amount across a field within a year. It is assumed that any variation within a field is due to either the natural variability of the crop or differences in the management zones. If the farmer is already managing according to a set of "management zones", the algorithm cannot distinguish this from the management zones inherent in the field from unmanaged inputs (e.g., from soil type and elevation). For the remainder of this section "managed zones"




(a) 1000 Gibbs samples per EM iteration




(b) 10000 Gibbs samples per EM iteration

**Figure 7.5.** Recorded parameter changes for the field Gott East 93, computed according to (7.3) and (7.4)

will refer to the zones by which the field was actually managed, in contrast to the underlying management zones the delineation seeks to find.

Today, with precision agriculture, many fields today are planted non-uniformly. If a field is already managed according to a set of zones with the zones being managed differently, this is likely to create a sort of feedback loop between the management and the delineated zones. The algorithm differentiates the zones based on them yielding differently, but managing the zones differently is also likely to make them yield differently. Thus the management of the field based on zones determined with yields managed by zones is likely to become a self-fulfilling prophecy.

### 7.4.1 Delineate Based on Periodic Uniform Management

One solution which is good from a data collection perspective would be to periodically (say every 5 years) manage a field uniformly. Then only the data of these uniformly managed years' yields would be used to delineate management zones via the proposed algorithm. This would remove any effects from the management of the field (assuming the lingering effects from the previous years' management is negligible) to find the inherent management zones.

However, this method has the practical drawback of requiring uniform management for a year rather than the preferred management practices of the farmer. Yields for the uniformly managed years are likely to be less than if the zones had been well managed with variable rates. There would be a trade-off between the loss of the uniform years' yields and the increased yields for other years based on better zone delineations.

### 7.4.2 Delineate Managed Zones Separately

What initially seems like a simple way to avoid the effects of the managed zones skewing the resulting delineations is to simply delineate each zone separately rather than delineating the whole field together. However, there are two major issues with this.

The first issue would be that the managed zones can change from year to year. This means that even if you separate the data by a given years' managed zones, the other years' data are still potentially in multiple managed zones in previous years. The only way to really

separate out the effects by delineating only one managed zone would be to use only the one year's data. This would remove much of the benefit of the proposed model and algorithm.

Another issue with delineating the managed zones separately is that once the zones are delineated one would need a meaningful way to combine these disparate delineations. The labels between the delineations would not be equivalent, that is, there is no reason to think label 1 from one delineation is in any way similar to label 1 of another delineation.

### 7.4.3 Incorporate Management Effects into the Model

Another solution would be to try to incorporate the knowledge of the zones by which the field was managed into the model and/or algorithm. A simple first step might be to model the management effects as a scaling factor on what the unmanaged yield would have been.

Incorporating this into the model is more involved than it initially seems though. Since the zones a field is managed by can change each year, the scaling factors would need to be estimated separately for each year which has yield data.

## 7.5 Characterizing Yield Map Errors

One major question for the determination of management zones from yield data is how good those data are. There is certainly error involved in the creation of the yield map. Characterizing the various errors involved, both measurement errors and errors induced by the nonlinear behavior of the combine [22], is a next step. An understanding of these errors could be used to improve the data pre-processing and the model in order to better delineate management zones.

# Part II

# THE OADA API FRAMEWORK

# 8. INTRODUCTION

While the Internet has largely solved the problem of how to achieve scalable, widespread exchange of human-consumable data such as web pages, the seamless exchange of machine-usable data is still an area of ongoing development.

Many challenges remain in the Application Programming Interface (API) design space, especially with regards to finding and reusing APIs, and embracing the temporal evolution that is natural in products [23]. Many industries, such as agriculture and manufacturing, still struggle to achieve automated, cross-platform, and cross-organizational data exchange. Unsupervised machines, unlike humans, require more than simple format standardization. The machine must know how to reliably authorize, discover, identify, understand, ingest, and synchronize data in order to successfully perform most of its required tasks.

Many, if not most, modern systems that exchange data use a Representational State Transfer (REST) [24] API (Google, Twitter, Facebook, etc.). These APIs are usually specific, one-off designs. For example, the Google API for information about users is specific to Google and different from the corresponding Facebook API. In industries dominated by one or two large players, their APIs become de facto standard APIs.

In industries such as agriculture, manufacturing, and shipping, it is difficult to achieve API-level standardization due to the technical complexity of the problem itself and the large number of possible communication pathways between participants. In these industries, it is the user or data owner who must connect data platforms. The organizations behind data platforms cannot be expected to sign prior agreements or dedicate developers to each individual integration.

For example, in the context of agriculture, a farmer may want collected data (e.g., data collected by an open-source planting app tracking progress throughout the season) to flow to multiple other platforms that can utilize it. In the context of the food supply chain, a food processing company may need to automatically receive safety certifications from upstream vendors and route them to downstream customers. In the event of a food safety problem, this could help regulators to crawl backward through the supply chain data systems to trace the source of an outbreak.

This configuration of a web of interconnected platforms (i.e., "clouds") is known as an *intercloud* [25]. While micro-service style architectures *within* a platform's boundaries have become quite popular as a means of dealing with complexity, the intercloud model extends the concepts of micro-service architectures beyond platform boundaries. Figure 8.1 illustrates a potential intercloud scenario involving multiple APIs. The black text and icons are the involved clouds, devices, etc., while blue text and arrows show the movement of data. The following subsections detail the core concepts of the Open Ag Data Alliance (OADA) API Framework. Intercloud is an active area of research [23], [26], [27].

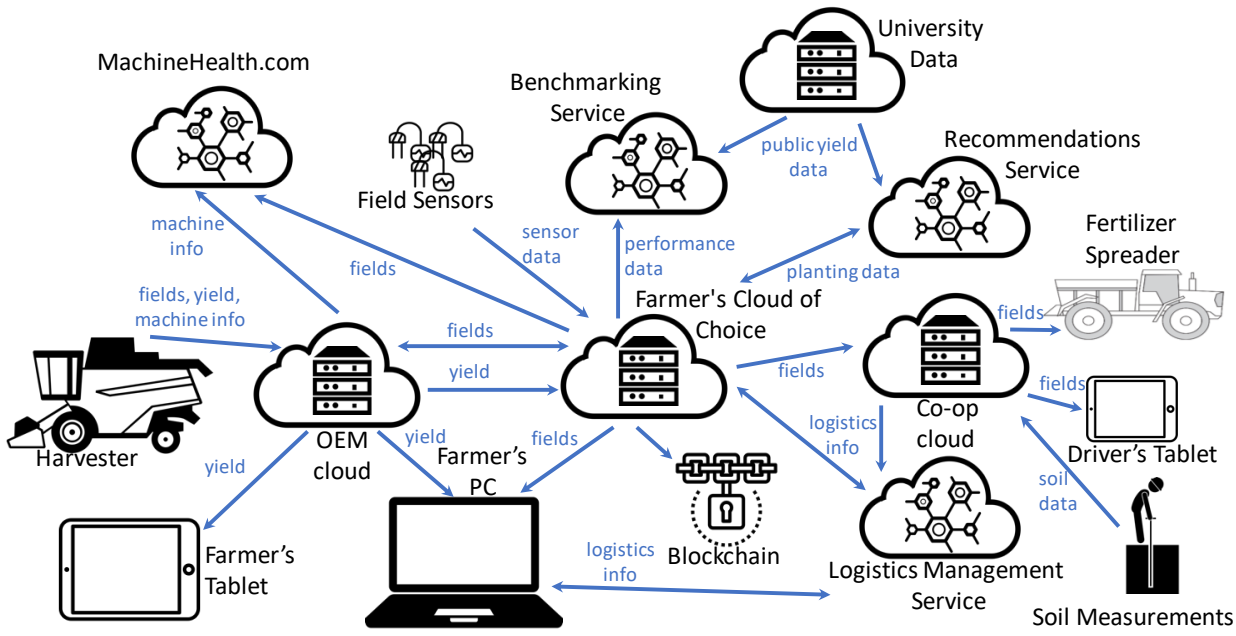



**Figure 8.1.** Illustration of an intercloud scenario involving multiple APIs

This part of the manuscript presents work on a new framework for solving these problems of automated, real-time, and historical data exchange between systems that is initiated by data owners. It provides pathways to adoption-based API standardization in emerging and heterogeneous industries. The goal of this work is to facilitate a common API framework, enabling a sort of micro-services engine that supports ad-hoc architectures without pre-defined boundaries.

The following sections present a review of related work, a detailed description of the framework, a Proof-of-Concept (PoC) implementation utilizing this framework, a discussion

of some real-world use cases already employing the framework, and finally conclusions on the utility of the framework.

## 8.1 Related Work for OADA

Open Data Protocol (OData) is a set of best practices for making and using RESTful APIs [28]. This work focuses on producing machine-readable descriptions of APIs' data models. OData has been approved as a standard by the International Organization for Standardization (ISO) [29].

OpenAPI (formerly Swagger) works on defining a standard interface to RESTful APIs [30]. Its work focuses on producing a description that allows *both* machines and humans to discover and understand a RESTful API, as opposed to producing strictly machine-readable information.

GraphQL takes a client developer-focused approach for creating web APIs [31]. GraphQL instead enables API clients to define the parts of the data structure relevant to them such that the API will only return or mutate data of interest to the client, rather than the traditional REST model in which the server defines the format of the data. This enables painless evolution of the API.

AsyncAPI works on defining a specification for documenting and describing message-driven APIs [32]. It is protocol-agnostic (can be applied to Hypertext Transfer Protocol (HTTP)), and seeks to be as compatible as possible with the OpenAPI specification [30] mentioned above.

## 8.2 Representational State Transfer (REST)

REST [24] is hard to precisely define because it is an architectural style, rather than a published standard or protocol like SOAP (formerly Simple Object Access Protocol) [33]. The main requirements of REST are client-server architecture, stateless, cacheability, layered system, and uniform interface.

REST defines resources as the elements which are communicated between server and client. These resources can represent any information of interest (e.g., images, queries,

collected data). Implementation details such as how the resources are actually stored and what database is used are hidden from clients. Clients access resources on the server only using Uniform Resource Identifiers (URIs). Each resource has its own unique URI.

The communication between client and server should use a stateless protocol. Statelessness here refers to the high-level protocol used between client and server, it is not saying that the lower network layers (e.g., transport and media layers) involved in getting requests from client to server and back are necessarily stateless. Here, stateless means that every request received by the server is independent of every other request and can be interpreted without any awareness of other requests sent. This enables handling requests in distributed and concurrent fashions, which allows for what is known as horizontal scaling [34]. Horizontal scaling is increasing the capacity (i.e., number of clients which can be handled simultaneously) by using an increasing number of separate nodes (i.e., server instances). The independent requests can be handed off to different server nodes without requiring the nodes to be aware of the other nodes or other requests.

Clients, as well as any intermediate parties (e.g., proxy servers), can cache server responses. Server responses need to indicate themselves as cacheable or non-cacheable, in order to prevent clients from using stale responses. Cacheing can reduce the number of client-server interactions needed, improving scalability.

Clients generally cannot tell if they are connected directly to the end server or an intermediary. These intermediaries can be things like proxies, load balancers, or security layers placed between clients and the server. These intermediaries do not affect communication between client and server. Therefore such layers can be added without needing to update server or client, enabling improved scalability with things like load balancing and shared caches.

A REST API provides uniform representation of the exposed resources, which does not correspond directly to how the server internally stores and represents the data. Two servers implementing the same REST API would expose the same resource representations to clients, but could store and internally represent those resources in entirely different ways. These representations of resources are what clients use to manipulate the state of the resources,

such as modifying or deleting the resource. This distinction decouples the server and the client, allowing for their implementations to evolve independently.

## 8.3 Hypertext Transfer Protocol (HTTP)

HTTP is a client-server communications protocol [35]. It is commonly used as the application layer protocol for APIs. While the two are often conflated, HTTP is distinct from REST and neither one requires the other. REST is a methodology for designing APIs and how they ought to function, whereas HTTP is a specific communication protocol APIs can use.

### 8.3.1 Requests

HTTP is a request-response protocol. The protocol is initiated by a client sending a request to a server. A request consists of up to four parts. First, the request line is sent which contains the method, the target, and the protocol version, terminated with a carriage return and a linefeed (e.g., GET /resources/123 HTTP/1.1). Second, the zero or more header lines may be sent each consisting of a field name, a colon, and a field value, terminated with a carriage return and a linefeed (e.g., Content−Type: application/json). Then an empty line must be sent, consisting of only a carriage return and linefeed, to indicate the end of the header section. Finally, an optional request body may be sent.

### 8.3.2 Methods

HTTP defines various methods as well as allowing for the use of new methods [36]. The relevant standard HTTP methods for this manuscript are GET, HEAD, PUT, POST, and DELETE. GET is used to request the current state of the target from the server. HEAD is like GET but tells the server to not send a response body. This is useful for retrieving metadata in the response header without having to transfer all of the data. PUT is used to update the state of the target on the server based on the request body. POST is used to request the server process the request body according to the rules of the target. For

example, a POST request to a /users might be used to request the server to create a new user. DELETE is used to request the server to delete the state of the target.

### 8.3.3 Responses

For every request from a client, the server sends a corresponding response. A response consists of: A response consists of up to four parts. First, the status line is sent which contains the protocol version, the status code, and an optional reason phrase, terminated with a carriage return and a linefeed (e.g., HTTP/1.1 200 OK). Second, the zero or more header lines may be sent each consisting of a field name, a colon, and a field value, terminated with a carriage return and a linefeed (e.g., Content−Type: application/json).Then an empty line must be sent, consisting of only a carriage return and linefeed, to indicate the end of the header section. Finally, an optional response body may be sent.

### 8.3.4 Status Codes

A status code is a three-digit integer code (e.g., 200) for the result of the server's attempt to process and fulfill the client's corresponding request [36]. The first digit of a status code defines its class. Clients may not understand all status codes, but they must understand all classes and treat unknown codes as being equivalent to the X00 status for the corresponding class (e.g., if 234 is unknown to the client, it shall treat it as 200). The 1XX (Informational) status class means the request was received and understood. It is a provisional response while processing on the request continues. The 2XX (Successful) status class means the request was received and accepted. The 3XX (Redirection) status class means further client action is needed in order to complete the request. The 4XX (Client Error) status class means the request was invalid or not understood. The 5XX (Server Error) status class means the server failed to process a request despite the request appearing to be valid.

# 9. OADA API CORE CONCEPTS

The OADA API was designed with the idea that there will be multiple clouds and APIs. This is why rather than being a *specific* API, OADA is a framework for designing APIs which lend themselves to interoperability in an evolving intercloud environment.

## 9.1 RESTful Design

The first core concept in OADA is adherence to REST design principles, discussed previously in Section 8.2. Any OADA conformant API is therefore a RESTful API. Many APIs today utilize the concept of REST [37]. While REST is not specific to HTTP [24], OADA calls for using Hypertext Transfer Protocol Secure (HTTPS) and WebSockets in its implementations.

With REST in mind, OADA represents the API data as a set of resources that can be found, read, modified, deleted, shared, connected linked, or watched for changes. A resource is conceptually a collection of bytes, but OADA gives JavaScript Object Notation (JSON) formatted resources a higher level of functionality than other formats. In particular, JSON resources are internally addressable, that is, Uniform Resource Locators (URLs) can be constructed to refer to internal parts of a JSON resource, and a part of one JSON resource can link directly to another resource. This functionality can theoretically be implemented for other formats, for example, Extensible Markup Language (XML), but OADA only requires JSON support.

### 9.1.1 Resources

Resources are uniquely identified with a URL comprised of a protocol such as https://, a domain, such as example.com, and a canonical home rooted at the /resources path. For example, the resource with identifier abc123 at domain example.com has a canonical URL of https://example.com/resources/abc123.

For JSON resources, the URL can be extended beyond the canonical resource to include paths within the JSON itself. For example, the URL https://example.com/resources/abc123

/key1 refers to the key1 part of the abc123 resource. Additionally, if the value at the key1 key of the abc123 resource is itself a link to another resource, then the URL extends to refer further into parts of the linked resource.

In OADA, the current state of a resource can be retrieved with a GET request to its URL. A resource may change over time, but a GET request to the resource always returns its current state as of the time the request is processed by the platform. HEAD requests and conditional GET requests [38] are also supported in OADA, to facilitate client caching. If the requested resource has changed from what is in the client's cache, the server will respond with the new version of the resource, otherwise, the server will send a response indicating the cache is valid.

One of the main innovations of OADA is the ability to have live data graphs, that is, resources are not only available for reading via an HTTP GET, but also as a log of all changes to any arbitrary sub-tree of resources. For details on this protocol, please see Section 9.6 below.

### 9.1.2 Resource Mutation Methods

In a RESTful API, the state represented in a resource can be changed through particular methods. In an OADA API, the specific methods for changing resource state are: Create (uses the HTTP POST method), Upsert (uses the HTTP PUT method), and Delete (uses the HTTP DELETE method), as described below.

#### 9.1.2.1 Create

Performing the Create method on a URL results in placing the body of the request at a new randomly generated key within the addressed resource. This effectively appends a random string to the end of the specified URL and places the new data there. As a result, the Create method is non-idempotent, that is, executing Create multiple times with the same data will result in repeatedly adding the data at different random keys each time. This prevents accidental collisions when multiple resources are created concurrently.

Performing this method on the global resources endpoint (i.e., POST /resources) creates a new resource with a random resource identifier (i.e., it results in the resource /resources /<random new key> being created). Performing this request on any existing resource (e.g., POST /resources/abc123) results in the new data being placed at a random string key within the resource. The newly-created canonical path is returned in the Content−Location HTTP response header [36].

#### 9.1.2.2 Upsert

In an OADA Upsert operation, the body of the request is merged with the current state of the addressed part of a resource to create the new state. Here, merge means recursively adding the keys from the request body to the target resource, overwriting any existing keys. If the resource did not exist at all prior to the upsert, it is created transparently with a state equal to the request body.

The Upsert operation is idempotent, that is, executing the same Upsert operation multiple times results in the same final state. For example, if the resource did not exist before the first Upsert, it was created and its state was made exactly equal to the request body. A second identical request will tell the server to modify the now-existing resource by replacing its contents with the identical contents of a repeated request body, resulting in the same state as before this second request.

For JSON resources, only the parts of the JSON object that exist in the request body are merged (unmentioned keys are not modified). In this way, an Upsert can be thought of as an idempotent merge operation. The merge itself is a recursive merge, that is, if other keys exist at any given level of a JSON resource, only the keys at that level mentioned in the request body will be overwritten or created. In this type of merge, it is impossible to completely replace the contents of a JSON object at any given level. One can only ensure that the specified keys will exist, and they will point to objects which at least contain the further specified keys, but may contain additional keys. For non-object values such as strings, numbers, and booleans, this merge will completely replace the original value with the new

value regardless of the type of the new value (e.g., a string would be completely replaced with a new object).

#### 9.1.2.3 Delete

Although Upsert can set the value corresponding to a given key, creating that key if it does not yet exist, it cannot remove a key. Therefore, OADA has a Delete method that explicitly removes a state element. A Delete can be performed on a resource as a whole, in which case it will be entirely removed from the current state. Additionally, a Delete can be performed on a specific element of a resource, removing only the specified path within the resource while leaving the rest of the resource unchanged.

Delete is also an idempotent operation, that is, multiple uses of Delete on the same URL will result in the same final state of the resource (i.e., the specified key will be absent).

## 9.2 User-Centric REST APIs

Each OADA resource is owned by a user account. The corresponding user can choose to give other local accounts on the same cloud access to the resource, and the user can also sync the resource to another cloud that supports the OADA API. For clarity, a user is a real person, not code interacting directly with an API.

### 9.2.1 User-driven Connections

The OADA framework aims to allow users to use, share, and authorize their data as they see fit rather than relying a priori interoperation agreements between cloud providers. A client written to use an OADA conformant API can be used with any OADA-conformant platform, even if the client and platform do not have advanced knowledge of each other. To that end, the OADA framework supports Dynamic Client Registration [39]. For the purposes of this section, a client is any application which needs to ask for an authorized token to use on behalf of a particular user at an OADA conformant cloud. The user interacts with a client (e.g., a website), and that client interacts with the OADA API.

Dynamic Client Registration allows clients to register themselves with platforms automatically, obtaining a client identifier in response that can be used in future OAuth 2.0 requests for tokens. The platforms still have the ability to manage these clients, such as blacklisting clients which misbehave in some way. However, OADA encourages letting users pick clients as they see fit. Even after registering, a client can only access data to which a user has granted it access, and a user cannot grant a client access to data to which that user does not have access.

Dynamic Client Registration poses a hurdle to traditional OAuth 2.0 token distribution in that it cannot use the common method of pre-shared client secrets. Therefore, during the registration process, the client provides a JSON document describing itself which is digitally signed by a trusted signature authority, and contains a public key for the client. When the client performs an OAuth 2.0 request, it must create a client secret on-the-fly in the form of a JSON Web Token (JWT) that is signed with its corresponding private key as proof it is whom it says it is.

### 9.2.2 Users and User Accounts

It is important to distinguish between users and user accounts because the mapping is not necessarily one-to-one. For the purpose of this work, the term user refers to an actual person (or another real-world entity like a company with a representative). A user may have more than one account within a given OADA conformant cloud.

A person or company may use multiple OADA conformant clouds, and will have a distinct user account on each. From the perspective of an OADA API, two user accounts are distinct entities, even if they happen to belong to the same actual user. The example of a single user with two user accounts corresponding to two different OADA conformant clouds, is illustrated in Fig. 9.1.

### 9.2.3 Federated Identity / Universal Login

A user can use an identity from one OADA cloud to log into another. This results in the second cloud transparently creating a new account and linking it to the original account.

This facilitates users sharing and migrating their data between clouds, or even utilizing multiple OADA conformant platforms concurrently in an intercloud scenario without the need to manage many identities. A user can choose any platform acting as an OADA identity provider to handle his/her identity, using an OADA-conformant version of the OpenID Connect protocol [40].

### 9.2.4 Sharing and Permissions vs Sync

User-driven data sharing can happen between user accounts within an OADA cloud, or from a user account on one cloud to a user account on another cloud. While one might think of both things simply as sharing, they are actually separate operations.

The relation between permissions and sync is illustrated in Fig. 9.1. While a single physical user is involved, that user has two user accounts (one in each cloud). Permissions within a cloud control what other user accounts within that cloud can access the user's data, whereas sync is used to move the user's data between the clouds.

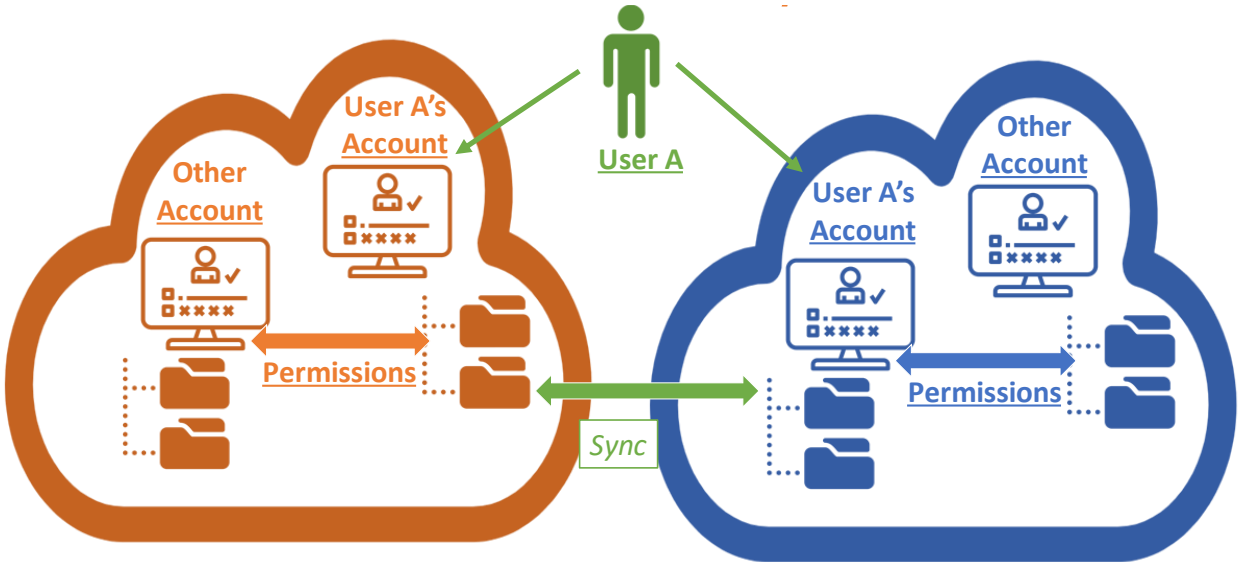



**Figure 9.1.** Illustration of intra-cloud vs intercloud sharing

Permissions pertain to a user accounts *within* a given platform and which data each account may access. For example, within platform 1 User A can grant read permissions to User B's account for a set of data belonging to User A. However, in platform 2 these

permissions from platform 1 would have no effect, even if both User A and User B also have accounts at platform 2, because they were only applied on platform 1. User A could separately share his/her data within platform 2 to User B's account on platform 2 as well.

Sync is *between* two platforms and is distinct from permissions. As shown in Fig. 9.1 if User A has accounts on two OADA platforms, the user could sync data from the account on platform 1 to the account on platform 2. This is effectively sharing between the two accounts, though the accounts belong to the same actual user. Sharing between two different users on two different platforms thus involves a combination of both permissions and sync.

## 9.3 Leverage Existing Standards

OADA seeks to avoid reinventing the wheel where possible. This increases the probability of being able to leverage existing tools. Strictly speaking, many of the standards utilized in OADA are not approved standards, but they are de facto standards with pre-existing implementations and adoption. What is meant by this is that they have a published Request for Comments (RFC) and already have widespread adoption, but are not officially adopted as a standard [41]. These are listed in Table 9.1 along with their corresponding defining documents.

## 9.4 Resource Meta-Data

OADA also has a special class of resource, meta resources, which hold meta-data about resources. Every normal resource has its own meta resource (but a meta resource does not have an associated meta resource). Normal resources contain a _meta key whose value is a link to that resource's meta resource.

This allows for storage of data about a resource which do not necessarily belong within the resource. Having a separate meta resource is especially important for storing formats that do not allow adding arbitrary keys. For example, if a resource is an image format, the meta resource provides a place to store arbitrary data about it.

**Table 9.1.** De Facto Standards Utilized in OADA

| Name | Document(s) |
|---|---|
| Hypertext Transfer Protocol (HTTP) 1.1 | RFC 7230–7237 [35], [36], [38], [42]–[46] |
| Hypertext Transfer Protocol Secure (HTTPS) | RFC 2818 [47] |
| OAuth 2.0 | RFC 6749 [48] |
| Dynamic Client Registration | RFC 7591 [39] |
| JavaScript Object Notation (JSON) | RFC 8259 [49] |
| JSON Web Signature (JWS) | RFC 7515 [50] |
| JSON Web Token (JWT) | RFC 7519 [51] |
| JSON Web Key (JWK) | RFC 7517 [52] |
| OpenID Connect | OpenID Connect Core 1.0 [40] |
| Web discovery | RFC 8615 [53] |
| GeoJSON | RFC 7946 [54] |

**Listing 9.1**. Example OADA resource state

```
{
  "_id": "resources/123",
  "_rev": 1,
  "_type": "application/vnd.foo+json",
  "_meta": {
    "_id": "resources/123/_meta",
    "_rev": 1
  },
  "foo": {
    "bar": {
      "a": 1
    }
  }
}
```

OADA stores information such as create/modify date, owner, and user access for a given resource in its meta resource. Also, a resource's change feed (detailed in Section 9.6) is accessible via a link under the _changes key of the corresponding meta resource.

## 9.5 Graph-Based Data Representation

OADA resources can arbitrarily link to other resources, forming an overall traditional URL-driven API structure. In other words, part of the state of one resource can be a reference to another.

These links between resources create a graph. This graph of data can be traversed using the URLs of API requests. The traversal via URL is described in the subsections that follow.

### 9.5.1 Resource Fields as Children

OADA not only treats explicit links between resources as a graph but also the keys and sub-keys of resources can be accessed directly. For example, consider an OADA-conformant platform where GET /resources/123 returns the state shown in Listing 9.1.

**Listing 9.2**. State of key `foo` of `resources/123`

```
{
  "bar": {
    "a": 1
  }
}
```

**Listing 9.3**. State of key `bar` of `resources/123/foo`

```
{
  "a": 1
}
```

**Listing 9.4**. Example JSON of an OADA link

```
{
  "_id": "resources/456",
  "_rev": 1
}
```

The keys within this resource may be accessed directly as if they were their own resource. For example, the key foo maybe be accessed directly with the request GET /resources/123/foo. The state returned by that request in this case is shown in Listing 9.2.

This traversal of keys is not limited to the first-level keys of a resource. The URL of a request can refer to an arbitrarily deep sub-key of a resource, and the traversal as above will be carried out in a recursive fashion. For example, the request GET /resources/123/foo/bar, would produce the state of the key bar of the state from Listing 9.2. The result of this request is shown in Listing 9.3.

### 9.5.2 Links and Link Traversal

The links between resources are followed transparently (i.e., the client does not need to know about them), and traversed similarly to the traversal described in Section 9.5.1. The representation of a link with JSON in OADA is shown in Listing 9.4.

**Listing 9.5**. Example of OADA resource containing a link to another resource

```
{
  "_id": "resources/foo",
  "_rev": 2,
  "_type": "application/vnd.foo+json",
  "_meta": {
    "_id": "resources/foo/_meta",
    "_rev": 2
  },
  "bar": {
    "_id": "resources/baz"
  }
}
```

**Listing 9.6**. Example of an OADA versioned link

```
{
  "_id": "resources/111",
  "_rev": 9001
}
```

The _id key is required and its presence is what makes a key of a resource into a link. The _rev key is optional, OADA links are traversed in the same manner regardless of the presence of that _rev key.

Shown in Listing 9.5 is an example of a link occurring within a resource. In the example, the resource with id resources/foo has a key bar which is a link to another resource with id resources/baz. What this means in terms of OADA APIs is that the request GET~/resources/foo/bar is equivalent to the request GET~/resources/baz since they resolve to the same resource, and the two requests will return the same state.

### 9.5.3 Versioned and Unversioned Links

As mentioned in Section 9.5.2, OADA links have an optional _rev key. A link with a _rev key is a versioned link, and a link without one is an unversioned link. Examples of a versioned link and a unversioned link are shown in Listing 9.6 and Listing 9.7, respectively.

**Listing 9.7**. Example of an OADA unversioned link

```
{
  "_id": "resources/111"
}
```

In a versioned link, the _rev key tracks the _rev of the linked resource. When the _rev of the versioned link changes, it is considered a change to the resource containing the link (i.e., the parent). This in turn makes the _rev of that parent resource update after the child is changed. Figure 9.2 illustrates this upward propagation of _revs through the versioned link resource graph. A path with three levels is shown, but the propagation holds for any number of versioned links.

_rev updates can come at a cost of increased processing required per write, so they should be used with care where necessary and not as a default for all links in a model. Please note that _rev updates are eventually consistent in an OADA conformant API (a write to a leaf node is not required to be immediately reflected in a parent node). It is expected that deeper graphs will have higher rev update latency, although the batching of changes discussed in Section 9.6.3 alleviates this under high write loads.

## 9.6 Live Data Graphs and Change Feeds

Resources in an OADA API have a history of the changes made to them. This history allows a client to keep track of the past changes and synchronize an external state to the server's current state by requesting and receiving only changes. Importantly, changes can be conveniently tracked, pushed, and replicated for any arbitrary sub-graph using versioned links as described above. This provides the OADA concept of a *live data graph*.

In an OADA API, the history of modifications to a resource is called a *change feed*. A change feed is an ordered stream of *change documents*, which represent idempotent *changes* of resources given in JSON arrays. Applying a given change to a resource multiple times in a row results in the same net state of the resource. This enables an at-least-once delivery semantic for changes. The changes in a change feed are indexed by the resource's revision number.

### 9.6.1 Change Types

A change to a resource is represented by a JSON document and is either a *merge change* or a *delete change*. A merge change represents new or modified properties of the resource by

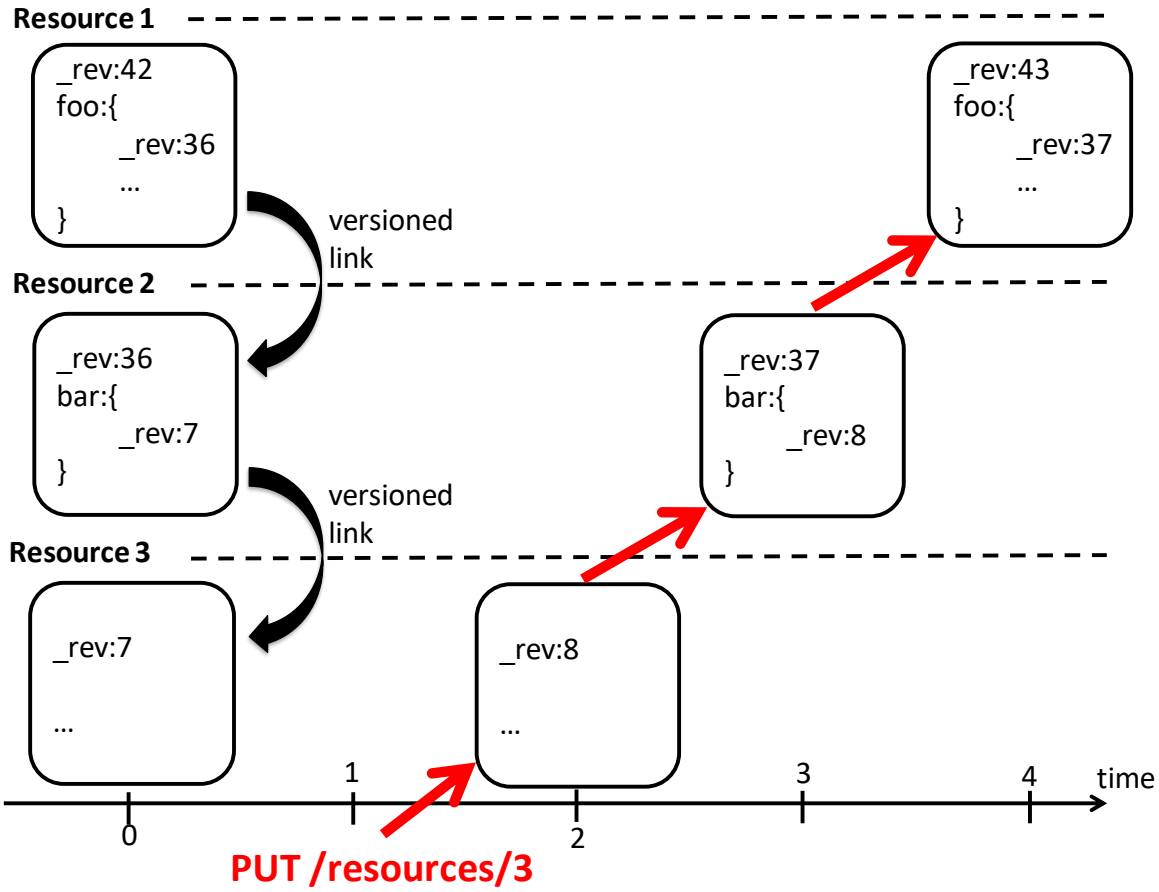



**Figure 9.2.** Illustration of a change to a resource causing the upward propagation of rev changes

only containing the new key and value pairs (i.e., the contents of the change body are the same as the contents of the body of the Upsert required to change the resource from its state before the change to its state after). Note that deleted properties are not represented by a merge change for the same reason that Upset and Delete are separate operations. Therefore, OADA represents a change of removing a value in JSON by using null to represent the value that was removed along with specifying a type of "delete" in order to explicitly differentiate the two cases.

### 9.6.2 Change Trees

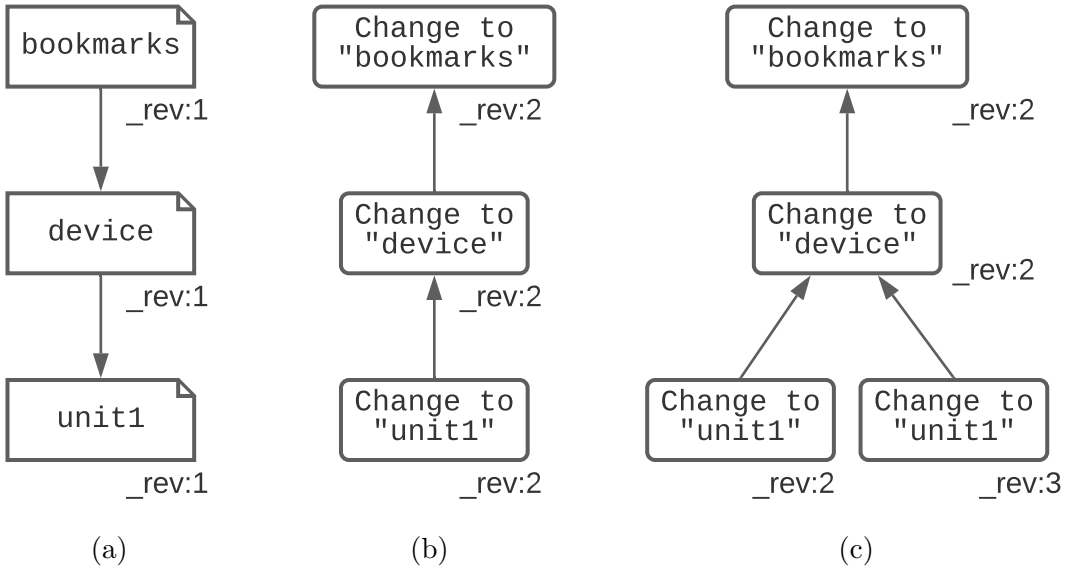



**Figure 9.3.** Example resource tree and change trees. (a) A resource tree containing three resources and two versioned links. (b) The resulting change tree after unit1 is modified once. (c) The resulting change tree after unit1 is modified twice with the batched change feature enabled.

Any changes made to a resource will propagate to its parents connected with a versioned link, as mentioned previously in Section 9.5.3. This chain of propagation is described by a *change tree* in which a node represents a single change to a specific resource in the graph and an edge represents change propagation over a versioned link. These change trees are the language by which a live data graph can be streamed.

A change tree is applied to a set of resources by traversing it depth-first, visiting children in order of increasing revision. Applying the changes in this order will transition a resource from the state immediately before it to the state after.

For example, consider the case shown in Fig. 9.3a of making changes to the path /bookmarks/devices/unit1, where bookmarks, devices, and unit1 are all resources and connected with versioned links. In this scenario, a single change to the leaf resource unit1 creates a change tree with three change nodes as shown in Fig. 9.3b.

### 9.6.3 Batched Changes

The number of new change nodes increases with the depth of the resource being modified since versioned links propagate changes up the graph. Creating many new change nodes can be costly since it requires both modifying the change graph and notifying any clients watching the relevant live data graph section.

The *batched changes* feature reduces the number of new change nodes by allowing an OADA platform to merge multiple changes for a live data graph into a single change tree. Notice that the most important (i.e., originating) change is always the leaf node, which corresponds to an API request. The other nodes are just to notify ancestors of the original change. These non-leaf nodes can be merged to represent a single change of the ancestor indicating that one *or more* changes have occurred to descendants.

Consider the previous example in which we make changes to a path /bookmarks/devices /unit1. Without batched changes, two change requests to unit1 create two change trees with a total of six change nodes. With batched changes enabled, the changes to bookmarks and device are merged respectively, and a single tree with a total of only four nodes is created as shown in Fig. 9.3c. This batching can be adjusted dynamically based on system load to prioritize either latency or throughput.

## 9.7 Generic Intercloud Data Sync

A user can set up a connection between two OADA-conformant platforms such that all (or a subset) of that user's data from cloud 1 can be automatically (i.e., without further user

action) pushed to cloud 2 over time. Any additions or changes to data after the connection is established will be pushed without further user intervention.

Users (via clients) are able to set up such connections between any two OADA APIs for which they have accounts without coordination with the maintainers of the APIs or the source platform. With OADA, cloud 1 need not even know cloud 2 exists (and vice versa) before the user sets up this connection.

There are two main parts of OADA facilitating this data movement. The first is the change feeds representing the live data graph, as discussed in Section 9.6. The second is the OADA methods of communicating these changes through intercloud updates.

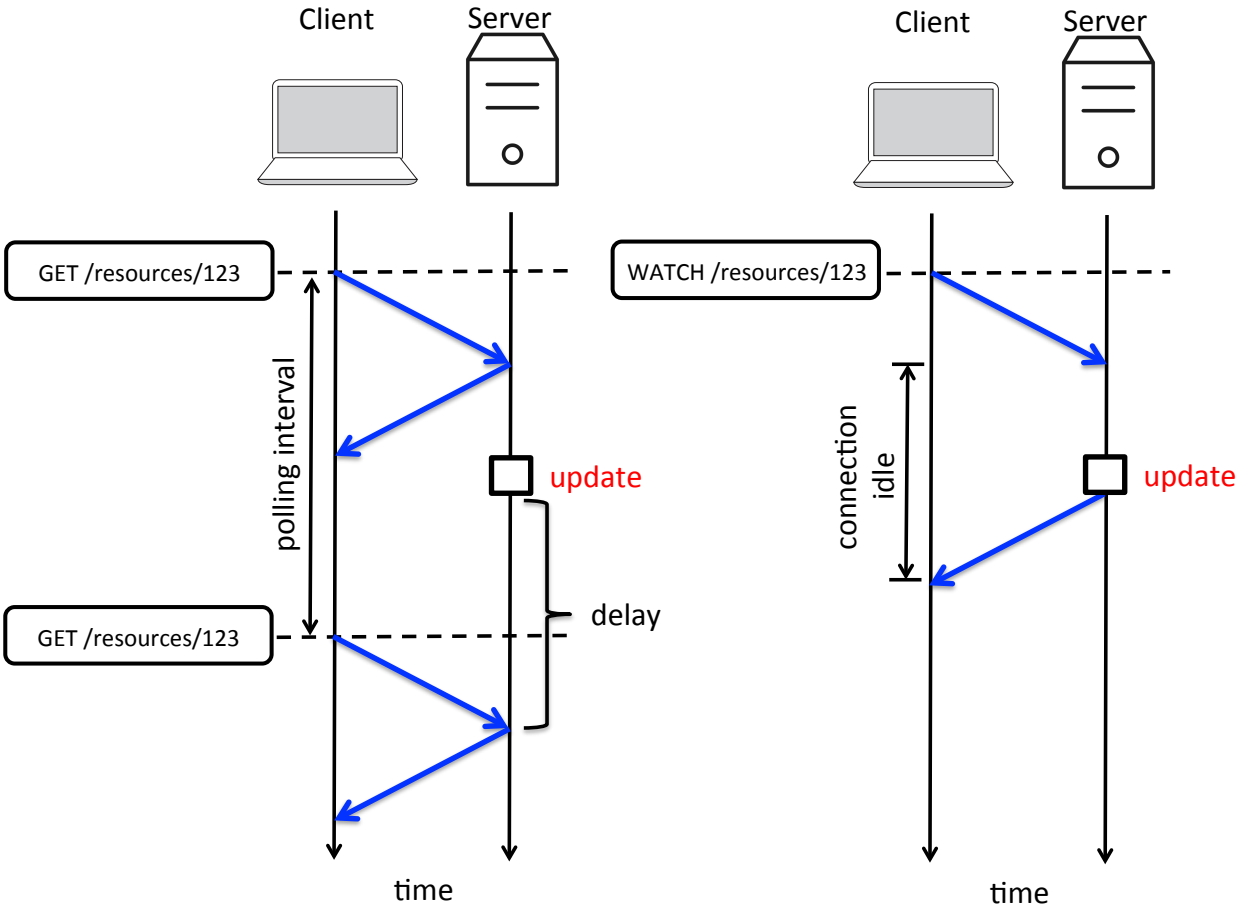



**Figure 9.4.** Illustration of push and poll models of updates from server to client

At a high level, there are two main ways for the updates to the live data graph to be sent from the platform where they are happening to the platform (or user device) consuming

them. These are referred to as “push” and “poll”. OADA supports both of these models. In fact, OADA has three variations of “push” available.

### 9.7.1 Polling

Polling is simply having the client periodically perform a GET on the endpoint of interest to see if it has changed since the last time the client retrieved it. Figure 9.4 shows the push and poll models of updates from server to client. When changes are infrequent, polling can result in unnecessarily long delays and/or excessive HTTP requests and therefore should be avoided in favor of push. However, polling is conceptually simpler to implement for a client as a starting point.

### 9.7.2 Webhooks

Webhooks is a solution that works well for pushing updates between two servers. It requires that the server can make REST requests to the client. Often this is not possible, but it is possible when the client is another server as opposed to a web browser or mobile app. In an intercloud scenario, both client and server are often servers. A client can register a webhook on any resource, and that webhook will be triggered on every change to the live data graph rooted at that node (i.e., every rev update to the resource). The webhook is configurable to make an HTTP request to any URL with statically-defined headers. In this way, any external service that can receive arbitrary HTTP requests can react to live event triggers from the remote live data graphs of interest to it.

### 9.7.3 OADA Sync Webhooks

While it is useful to notify an arbitrary HTTP API of changes to a live data graph, it is even more useful if the destination API is standardized (such as an OADA API). In other words, the webhook can be smarter if it knows the destination has an OADA-conformant API. The source platform can simply replay the same change at the destination resource and one-way replication is automatically enabled.

Since the OADA Upsert is idempotent, a platform can receive synchronization streams from multiple sources and trivially merge them creating a resource with data from all the sources. This allows clients to arbitrarily setup live data graph syncs where multiple streams can coalesce into a single resource and then be filtered, split, or otherwise redistributed elsewhere. For example, if a sequence of keys at cloud A are created as random identifiers, and another sequence of keys at cloud B are created also as random identifiers, then replaying the creating of the random keys from cloud A and cloud B at a resource in destination cloud C will result in cloud C containing all the data found in the resources on both cloud A and cloud B without collisions.

OADA sync webhooks also perform recursive synchronization of live data graphs rooted at any resource. This replicates a live data graph at a destination platform (i.e., changes to the source automatically flow to the destination). To achieve this, OADA maintains a mapping on the source platform between the resource ids in the source live data graph to the resource ids in the destination data graph.

### 9.7.4 WebSockets

WebSockets [55] is a widely used protocol which allows a persistent connection between a server and a client, enabling a server to push data to a connected client. While not strictly necessary for the sending updates from a server to a client, WebSockets are a good alternative to requiring the client to poll the server for updates.

WebSockets are especially useful for the case where the client is *not* an OADA conformant platform but is instead something like a browser, smart phone app, or micro-service as they do not require the client to expose a network-accessible REST API of its own. The connection between client and server is maintained for either side to initiate asynchronous communication, and it is simple for the client to know when the connection has dropped.

The key method for a WebSockets-based live data graph is WATCH. Setting a WATCH on a resource causes the change feed of the live data graph rooted at that resource to be streamed over WebSockets. The WATCH can be started at the current rev, or a past rev

can be specified from which to resume. The OADA platform will start streaming changes for that live data graph from the specified rev.

Micro-service architectures can utilize WATCH to react in real-time to things in particular parts of an OADA bookmarks tree.

### 9.8 Format Agnostic

While the discussion of OADA thus far has all been in the context of using JSON as the primary data serialization format (due to its widespread use in APIs [56]), any other formats are also supported and are referred to as *binary* formats.

OADA does not make any requirements on the semantic structure of data. It simply requires that the Content−Type be defined for every resource. Content−Types are strongly encouraged to specify a particular schema (e.g., application/vnd.oada.bookmarks.1+json) rather than simply a serialization (e.g., application/json). These semantic schemata, when combined in the graph, form a fully-defined API definition for live data graphs. Through the live data graphs themselves, live graph transformations can be maintained which can convert from one schema to another.

# 10. PROOF-OF-CONCEPT AND REFERENCE IMPLEMENTATION

While OADA itself is a framework, or API specification, rather than a specific piece of software, an OADA server implementing a base OADA conformant API has been developed [57]. This implementation has been used as a PoC, and as an open example of how to use OADA. It is also currently used in production environments and can easily add OADA conformance to a platform.

## 10.1 Open-Source

All of the code written in relation to OADA, both the PoC server and associated libraries, is open source. The code is openly available on the OADA GitHub [58] under permissive licenses.

## 10.2 Portable

The reference implementation is written in JavaScript, so it can be run on many platforms. The core micro-services are all written as Node.js [59] packages. This was chosen because it is supported on many platforms and the high availability of libraries for Node.js and JavaScript in general.

The reference implementation is written to not need a specific host OS or cloud provider and therefore is run via Docker [60]. Even though all of the core micro-services use Node.js, other services can be written in whatever programming language the writers prefer. Micro-services communicate over HTTP (or Kafka if in the core), so any languages and runtimes that support HTTP can be used to implement micro-services.

## 10.3 Architecture

The architecture of the reference OADA implementation is illustrated in Fig. 10.1. It is a micro-service based architecture, designed to allow for horizontal scaling and adding extra micro-services for new features. Each box is a separately running micro-service, database,

or server. Only one of each micro-service is pictured for clarity, but multiple instances of a given micro-service can be used to scale the platform horizontally.

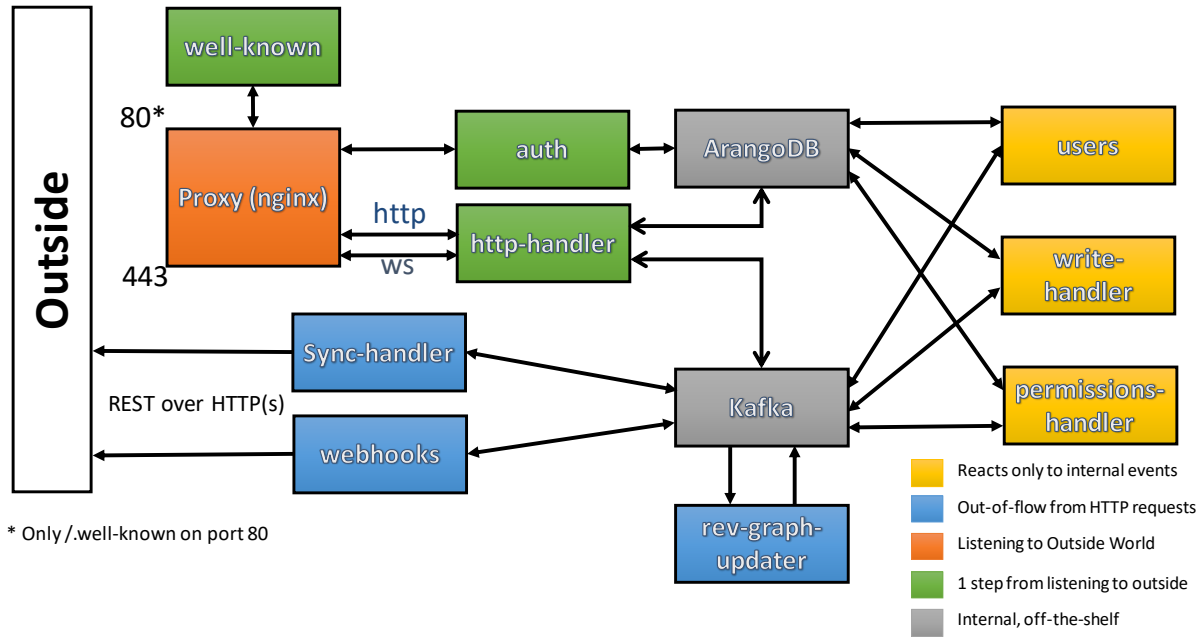



**Figure 10.1.** Architecture of the OADA PoC implementation

### 10.3.1 Core Micro-services

#### 10.3.1.1 HTTP Handler

The micro-service for interpreting and responding to incoming HTTP requests is http–handler. It checks the permissions and authorizations of incoming requests based on a token. It then retrieves the resource from the database (ArangoDB) for reads, or talks to the write handler over Kafka to execute writes.

#### 10.3.1.2 Auth

The micro-service for authenticating users using OAuth 2.0 [48] is auth. Clients perform the OAuth flow with the auth server and receive a token that can be used for HTTP requests.

It has its own collection in the ArangoDB database where it stores authentication information for clients and tokens.

#### 10.3.1.3 Users

The micro-service for adding and initializing new users is users. It handles things such as creating /bookmarks and /shares resources for new users.

#### 10.3.1.4 Write Handler

The micro-service for modifying the resources in the database for the Create, Upsert, and Delete methods is write−handler. It is designed to work with multiple instances running concurrently by partitioning the write requests such that all writes to a particular resource are always sent to the same instance of write−handler.

#### 10.3.1.5 Rev Graph Updater

The micro-service for propagating changes to _rev keys up versioned links is rev−graph −updater. It watches Kafka for any time a write to a resource occurs. When this happens, it queries ArangoDB for all parents of that resource which have a versioned link to it. For each parent found, it sends an Upsert to write the new value of the resource's _rev to the versioned link of the parent resource. This asynchronously implements the upward propagation of changes described in Section 9.6.

### 10.3.2 Kafka

The OADA reference implementation uses Kafka [61] as a message queue for the communication between the core micro-services.

### 10.3.3 ArangoDB

The underlying database for the PoC implementation is ArangoDB [62]. It is where the actual JSON documents containing the state of the API's resources, the graph of resource connections, and the changes graph are stored.

ArangoDB was chosen for the database for two main reasons:

1. It has support for JSON storage and querying.

2. It has support for graph queries.

The reference implementation also stores other data in ArangoDB besides just resources, but under different collections.

### 10.3.4 NGINX

Incoming HTTP and HTTPS traffic is routed through NGINX [63]. It is used as a reverse-proxy, to delegate requests to the appropriate micro-service (http−handler, auth, or well−known) and more importantly to load balance across multiple instances of a given micro-service and handle TLS.

# 11. OADA API APPLICATION RESULTS

There are already a few successful use cases of the OADA framework and the reference implementation mentioned in Chapter 10. Some of these use cases are discussed in the following subsections. As with the reference implementation, the apps in the following sections are all available open-source on GitHub.

## 11.1 Field Work App

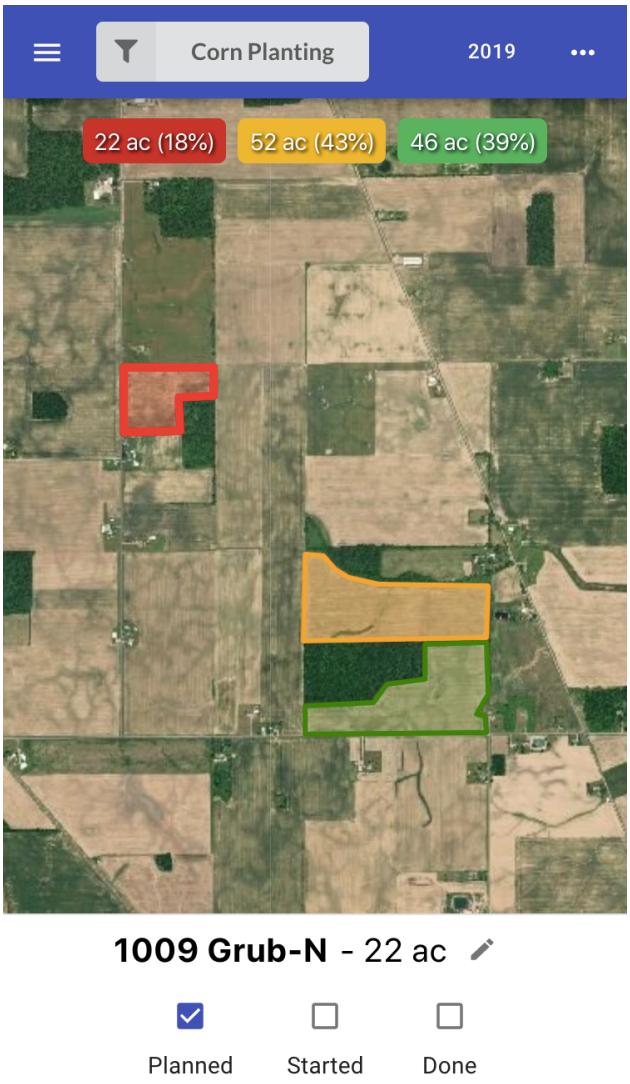



**Figure 11.1.** Screenshot of Field Work App, a web app designed to help farmers keep track of the status of operations in their fields.

The Field Work App [64] is a web-based application for farmers to track the progress of field operations such as seed planting, tillage, chemical applications, and harvest. It serves to aid in logistics planning as it presents both a geospatial view of progress on a map as well as a numerical summary in terms of the acreage completed and remaining. A screenshot of it is shown in Fig. 11.1.

While users may create field boundaries by drawing them within the Field Work App itself, they may also import them into OADA using a separate web-based import tool. An operation is created by providing a text title for the operation (e.g., "Soybean planting 2020") and setting a selection of fields to have a status of "planned". As operations proceed, users then advance the status of each field from "started" to "done" in order to generate summaries of progress. Operations can also be shared. For example, a harvest operation might be shared with an agronomist in order to notify him/her when a particular set of fields are ready to have post-harvest soil samples collected.

An integration was performed with a third-party data lake to provide a cloud-based data management platform with a functioning OADA-compliant API. A service was developed to perform a two-way sync of field boundary data between the data lake and the third-party's OADA-compliant platform. The syncing service consisted of three separate micro-services — one responsible for maintaining a clone of the relevant parts of the data lake in OADA, one for translating that dataset into the appropriate formats, and a third responsible for propagating changes made in OADA back to the data lake. The separation of the sync operation into three services allowed for simpler isolation from circular updates while offering improved debugging capabilities.

## 11.2 Trials Tracker App

The Trials Tracker [65] App is a web-based application for row crop farmers and similar agriculturalists to manage planned or impromptu yield trials. Through a simple interface, users can take note of yield trials, view mean yield values and compare trials to the remainder of the field or other trials. The Trials Tracker App recognizes the prominent role of geospatial data in the agricultural data ecosystem. The app leverages stream-computed aggregations

of the raw yield data from the live data graph that are geospatially-indexed in order to match the user's current zoom level and geospatial extent. As a result, all of the user's yield data are rendered on a seamless mapping interface that does not rely on the selection of a particular field. This simplifies the user experience. Apart from rendering visualizations, these aggregates can be used to compute several statistics and present them to the user.

Trials Tracker served as a pilot application for several OADA technologies that would benefit future OADA-driven applications. One example is the indexing services used to transform or re-index a given dataset such that it is more readily consumed by another application. A re-indexing service was used to translate the raw harvest data into the geospatially-indexed aggregates used by Trials Tracker. This was accomplished by subscribing to the live data graph where the raw data arrived, then ensuring the necessary resources holding the yield data aggregates existed. The re-indexing service was also responsible for computing running sums and other statistics as each raw data point is added to each aggregate. This was done in a stream-processing style for real-time data which was able to update the statistical totals based only on the contents of the change feeds rather than requiring reprocessing the large underlying datasets.

Trials Tracker also drove the development of additional front-end libraries and functionalities. Motivated by the need to update the yield data renderings and trial statistics in real-time within the app, an aid was developed to maintain subscriptions to push notifications for all of the proper OADA resources. Additionally, a caching layer making full use of OADA live data graph features improved the performance of the app given its data-heavy operations. Such functionalities were generalized into a client-side library for interacting with an OADA-compliant server [66].

### 11.3 Trellis Supply Chain Sovereign Data Automation

The Trellis Framework [67] is a brand name for OADA used in the food supply chain industry. Using OADA, the flow of relevant data (e.g., food safety audits) that are needed between trading partners along a supply chain (e.g., from grower to packager to retailer) is automated with selectable sharing rules and fine-grain privacy controls. This is achieved

without requiring global coordination in the industry; the only coordination required is between business partners who already coordinate to do business. As long as any two players have OADA-conformant platforms for their data, the syncing abilities of OADA can be leveraged to create an ad-hoc, automated data pipeline that supports the privacy requirements of its users.

For example, a farmer with food safety audits stored in Trellis can pre-configure that the food safety audits relating to cucumbers should go to three particular processors' Trellis platforms. Those processors can configure their platforms to automatically sync their food safety audits to any downstream distributors which receive product from them. The same holds from distributor to retailer to consumer.

A unique feature of this automatic, real-time, peer-to-peer supply chain data exchange model is that the existence of a standardized API enables new privacy controls such as the ability to Mask & Link [68]: replace sensitive data in a document with a hash of the original data and a Trellis URL pointing to where to retrieve it if you have permission, acting as an auditable, automated redaction engine.

While the work of this use case focused on food safety, the method applies to other supply chain scenarios such as advance shipment notices or sustainability tracing. It is not specific to the data being transferred nor their format.

### 11.4 ISOBlueApp

ISOBlueApp [69] is a web-based application that shows current and historical telemetry data collected by ISOBlue devices [70], [71] connected to agricultural vehicles in real-time. A screenshot of it is shown in Fig. 11.2.

The telemetry data are sent from ISOBlue devices to an OADA-conformant platform in real-time and indexed by the device name, date, and hour. For example, the location data collected by unit1 on October 16th, 2020 at 10:20 AM UTC will be stored to the path `/bookmarks/isoblue/device-index/unit1/location/day-index/2020-10-16/hour-index/10`. ISOBlueApp allows a user to monitor the updates on the server using a WATCH request and visualize the retrieved information in real-time.

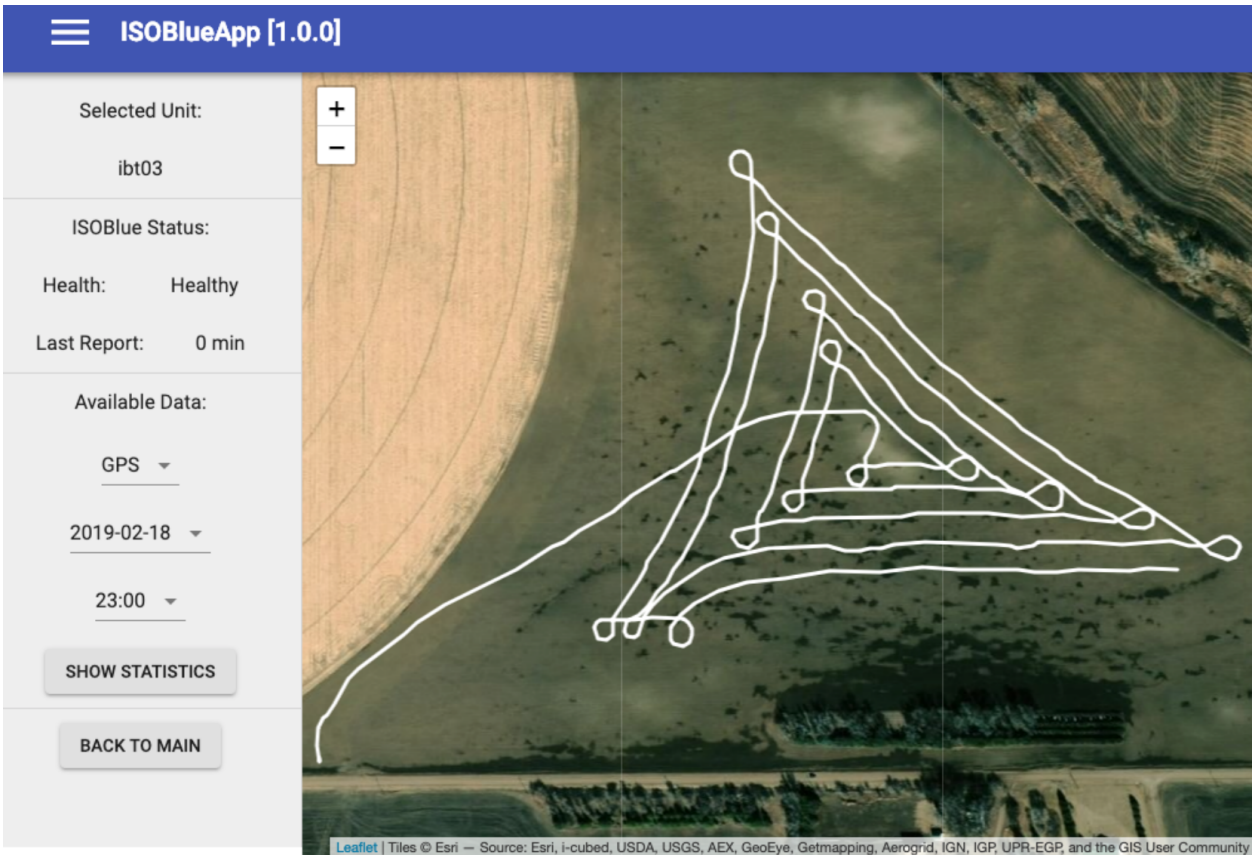



**Figure 11.2.** Screenshot of ISOBlueApp, an interactive tracking application for agricultural telemetry devices. The application retrieves real-time information from an OADA-conformant platform.

# 12. CONCLUSIONS

In this work a common API framework for intercloud environments was detailed, and a reference implementation of it was presented. Multiple cases of the framework already being used were also discussed. The framework covered in this work successfully facilitates the development and data flow for the discussed uses.

It is certainly the case that designing a specific server and API from the ground up for a particular fixed graph would be more performant than using OADA for that same graph. However, OADA allows for greater flexibility and code reuse than creating new servers and API for each use case (i.e., each graph). By using this suggested framework and its live data graphs when developing APIs, the resulting APIs will be significantly more useful for intercloud scenarios than using REST alone. This is especially true for user-centered scenarios, and scenarios with multiple OADA-conformant platforms.

# 13. FUTURE WORK

## 13.1 Using the Part I Data with OADA

The flexibility of OADA allows clients to decide how best to organize their data. The optimum organization of data will depend on the specific use case. In the case of the presented stochastic expectation-maximization (SEM) based algorithm from Part I, the client would need to:

- get data by the crop type,
- get data by harvest year,
- and get data within a certian boundary (i.e., within a given field).

Based on these requirements of the application, one way to organize the data with OADA would organize the yield data is to first index by year, then by crop type, and then by geohash. A geohash is a public domain system for encoding geographic coordinates into short strings [72].

The root resource, /bookmarks, would link to a harvest resource as shown in Listing 13.1. The harvest resource, /bookmarks/harvest, would contain a list of links to year resources as shown in Listing 13.2. This list is indexed by year so, for example, the key 2020 is a link to a resource related to the 2020 harvest data. The 2020 harvest resource, /bookmarks/harvest/years/2020, would contain a list of links to crop resources as shown in Listing 13.3. This list is indexed by crop so, for example, the key corn is a link to a resource related to the

**Listing 13.1**. Response to `GET /bookmarks`

```
{
  "_id": "resources/123",
  "_rev": 1,
  "harvest": {
    "_id": "resources/456",
    "_rev": 1
  }
}
```

**Listing 13.2**. Response to `GET /bookmarks/harvest`

```
{
  "_id": "resources/456",
  "_rev": 1,
  "years": {
    ...,
    "2020": {
      "_id": "resources/2020",
      "_rev": 1
    },
    "2021": {
      "_id": "resources/2021",
      "_rev": 1
    }
  }
}
```

**Listing 13.3**. Response to `GET /bookmarks/harvest/years/2020`

```
{
  "_id": "resources/789",
  "_rev": 1,
  "crops": {
    "corn": {
      "_id": "resources/corn",
      "_rev": 1
    },
    "soy": {
      "_id": "resources/soy",
      "_rev": 1
    }
  }
}
```

**Listing 13.4**. Response to `GET /bookmarks/harvest/years/2020/crops/corn`

```
{
  "_id": "resources/789",
  "_rev": 1,
  "geohashes": {
    ...,
    "abcdefg": {
      "_id": "resources/abc",
      "_rev": 1
    },
    "hijklmn": {
      "_id": "resources/hij",
      "_rev": 1
    }
  }
}
```

2020 corn harvest data. /bookmarks/harvest/years/2020 The 2020 corn harvest resource, /bookmarks/harvest/years/2020/crops/corn, would contain a list of links to geohash tiles resources as shown in Listing 13.4. This list is indexed by geohash so, for example, the key abcdef is a link to a resource related to the 2020 corn harvest data within a given tile of latitude and longitude. The best geohash tile size depends on the specific use case. A geohash 7 characters long gives roughly 150m worst-case tile length, which is a good starting point when working at field-scale. Finally, the resource for the example geohash abcdef for corn in 2020, /bookmarks/harvest/years/2020/crops/corn/geohashes/abcdef, would contain a list of yield data points as shown in Listing 13.5.

Using this scheme, the algorithm code could be modified to dynamically fetch the needed yield data via an OADA API. To delineate zones for a given field, the code would first compute all the geohashes of the chosen length (7 in this example) which intersect the field. Then for a given crop, and year, each geohash would be requested with earropGET /bookmarks/harvest/years/y/crops/c/geohashes/abcdef. This would be repeated for each year and crop of interest. After receiving the responses to those requests, the algorithm would then have all the needed data to estimate the management zones.

**Listing 13.5**. Response to
`GET /bookmarks/harvest/years/2020/crops/corn/geohashes/abcdef`

```
{
  "_id": "resources/789",
  "_rev": 1,
  "data": {
    "sadasda": {
      "lat": 40.4293272,
      "lon": -86.9123666,
      "alt": 100,
      "yield": 200,
      "crop": "corn",
      ...
    },
    ...
  }
}
```

# REFERENCES


[1] C. Gorse, D. Johnston, and M. Pritchard, "Universal transverse mercator," in *A Dictionary of Construction, Surveying and Civil Engineering*, Oxford University Press, 2012, ISBN: 9780199534463.

[2] D. Shepard, "A two-dimensional interpolation function for irregularly-spaced data," in *Proceedings of the 1968 23rd ACM National Conference*, ser. ACM '68, New York, NY, USA: ACM, 1968, pp. 517–524. DOI: 10.1145/800186.810616. [Online]. Available: http://doi.acm.org/10.1145/800186.810616.

[3] "IEEE standard for floating-point arithmetic - redline," *IEEE Std 754-2008 (Revision of IEEE Std 754-1985) - Redline*, pp. 1–82, Aug. 2008. DOI: 10.1109/IEEESTD.2008.5976968. [Online]. Available: http://dx.doi.org/10.1109/IEEESTD.2008.5976968.

[4] E. G. Souza, C. L. Bazzi, R. Khosla, M. A. Uribe-Opazo, and R. M. Reich, "Interpolation type and data computation of crop yield maps is important for precision crop production," *Journal of Plant Nutrition*, vol. 39, no. 4, pp. 531–538, 2016. DOI: 10.1080/01904167.2015.1124893. eprint: http://dx.doi.org/10.1080/01904167.2015.1124893. [Online]. Available: http://dx.doi.org/10.1080/01904167.2015.1124893.

[5] R. J. Elliott, *Hidden Markov models: estimation and control*. New York: Springer, 2008, ISBN: 978-0-387-94364-0. DOI: 10.1007/978-0-387-84854-9. [Online]. Available: http://dx.doi.org/10.1007/978-0-387-84854-9.

[6] A. A. Farooque, Q. U. Zaman, A. W. Schumann, A. Madani, and D. C. Percival, "Delineating management zones for site specific fertilization in wild blueberry fields," *Applied Engineering in Agriculture*, vol. 28, no. 1, pp. 57–70, 2012. DOI: 10.13031/2013.41286. [Online]. Available: http://elibrary.asabe.org/abstract.asp?aid=37255%5C&t=5.

[7] N. R. Kitchen, K. A. Sudduth, D. B. Myers, S. T. Drummond, and S. Y. Hong, "Delineating productivity zones on claypan soil fields using apparent soil electrical conductivity," *Computers and Electronics in Agriculture*, vol. 46, no. 1 - 3, pp. 285–308, 2005, Applications of Apparent Soil Electrical Conductivity in Precision Agriculture, ISSN: 0168-1699. DOI: 10.1016/j.compag.2004.11.012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0168169904001346.

[8] K. Diker, D. F. Heermann, and G. W. Buchleiter, "Analysis of multi year yield data for delineating yield response zones," in *2003 ASAE Annual Meeting*, ser. 031086, 2003. DOI: 10.13031/2013.13974. [Online]. Available: http://elibrary.asabe.org/abstract.asp?aid=13974%5C&t=5.

[9] N. R. Kitchen, K. A. Sudduth, B. Myers, S. T. Drummond, and S. Y. Hong, "Site-specific productivity zones delineated using bulk soil electrical conductivity," in *2003 ASAE Annual Meeting*, ser. 032340, 2003. DOI: 10.13031/2013.15322. [Online]. Available: http://elibrary.asabe.org/abstract.asp?aid=15322%5C&t=5.

[10] S. Song, B. Si, J. M. Herrmann, and X. Feng, "Local autoencoding for parameter estimation in a hidden Potts-Markov random field," *IEEE Transactions on Image Processing*, vol. 25, no. 5, pp. 2324–2336, May 2016, ISSN: 1057-7149. DOI: 10.1109/TIP.2016.2545299. [Online]. Available: http://dx.doi.org/10.1109/TIP.2016.2545299.

[11] Y. Zhang, M. Brady, and S. Smith, "Segmentation of brain MR images through a hidden Markov random field model and the expectation-maximization algorithm," *Medical Imaging, IEEE Transactions on*, vol. 20, no. 1, pp. 45–57, Jan. 2001, ISSN: 0278-0062. DOI: 10.1109/42.906424. [Online]. Available: http://dx.doi.org/10.1109/42.906424.

[12] A. Banerjee and P. Maji, "Rough sets and stomped normal distribution for simultaneous segmentation and bias field correction in brain MR images," *Image Processing, IEEE Transactions on*, vol. 24, no. 12, pp. 5764–5776, Dec. 2015, ISSN: 1057-7149. DOI: 10.1109/TIP.2015.2488900. [Online]. Available: http://dx.doi.org/10.1109/TIP.2015.2488900.

[13] F. Guastaferro, A. Castrignanò, D. De Benedetto, D. Sollitto, A. Troccoli, and B. Cafarelli, "A comparison of different algorithms for the delineation of management zones," *Precision Agriculture*, vol. 11, no. 6, pp. 600–620, Dec. 2010, ISSN: 1573-1618. DOI: 10.1007/s11119-010-9183-4. [Online]. Available: https://doi.org/10.1007/s11119-010-9183-4.

[14] S. Miyamoto, *Algorithms for fuzzy clustering methods in c-means clustering with applications*. Berlin: Berlin: Springer, 2008, Includes bibliographical references (pages 235-243) and index., ISBN: 978-3-540-78736-5. DOI: 10.1007/978-3-540-78737-2. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-78737-2.

[15] M. A. Tanner, *Tools for Statistical Inference Methods for the Exploration of Posterior Distributions and Likelihood Functions*. New York, NY: New York, NY: Springer New York, 1996, ISBN: 978-1-4612-8471-0. DOI: 10.1007/978-1-4612-4024-2. [Online]. Available: http://dx.doi.org/10.1007/978-1-4612-4024-2.

[16] C. A. Bouman, *Model-Based Image Processing*. [Online]. Available: http://eng.purdue.edu/~bouman/publications/pdf/MBIP-book.pdf.

[17] M. L. Comer and E. J. Delp, "The EM/MPM algorithm for segmentation of textured images: Analysis and further experimental results," *IEEE Transactions on Image Processing*, vol. 9, no. 10, pp. 1731–1744, Oct. 2000, ISSN: 1057-7149. DOI: 10.1109/83.869185. [Online]. Available: http://dx.doi.org/10.1109/83.869185.

[18] W. M. Rand, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical Association*, vol. 66, no. 336, pp. 846–850, 1971. DOI: 10.1080/01621459.1971.10482356. eprint: http://www.tandfonline.com/doi/pdf/10.1080/01621459.1971.10482356. [Online]. Available: http://www.tandfonline.com/doi/abs/10.1080/01621459.1971.10482356.

[19] J. J. Fridgen, N. R. Kitchen, K. A. Sudduth, S. T. Drummond, W. J. Wiebold, and C. W. Fraisse, "Management zone analyst (MZA): Software for subfield management zone delineation," *Agronomy Journal*, vol. 96, no. 1, pp. 100–108, Jan. 2004. DOI: 10113/8380. [Online]. Available: http://handle.nal.usda.gov/10113/8380.

[20] Soil Survey Staff, *Soil Survey Geographic (SSURGO) database*. Natural Resources Conservation Service, United States Department of Agriculture.

[21] C. W. Bobryk, D. B. Myers, N. R. Kitchen, *et al.*, "Validating a digital soil map with corn yield data for precision agriculture decision support," *Agronomy Journal*, vol. 108, no. 3, pp. 957–965, 2016. DOI: 10.2134/agronj2015.0381. [Online]. Available: http://dx.doi.org/10.2134/agronj2015.0381.

[22] E. M. Hawkins, "Organizing historical agricultural data and identifying data integrity zones to assess agricultural data quality," Ph.D. dissertation, Purdue University, 2016. [Online]. Available: http://docs.lib.purdue.edu/dissertations/AAI10172347/.

[23] K. M. Sim, "Intelligent resource management in intercloud, fog, and edge: Tutorial and new directions," *IEEE Transactions on Services Computing*, pp. 1–1, 2020, ISSN: 1939-1374. DOI: 10.1109/TSC.2020.2975168. [Online]. Available: http://dx.doi.org/10.1109/TSC.2020.2975168.

[24] R. T. Fielding, "REST: architectural styles and the design of network-based software architectures," Doctoral dissertation, University of California, Irvine, 2000. [Online]. Available: http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

[25] D. Bernstein, E. Ludvigson, K. Sankar, S. Diamond, and M. Morrow, "Blueprint for the intercloud - protocols and formats for cloud computing interoperability," in *2009 Fourth International Conference on Internet and Web Applications and Services*, May 2009, pp. 328–336. DOI: 10.1109/ICIW.2009.55. [Online]. Available: http://dx.doi.org/10.1109/ICIW.2009.55.

[26] S. Sotiriadis, N. Bessis, C. Amza, and R. Buyya, "Elastic load balancing for dynamic virtual machine reconfiguration based on vertical and horizontal scaling," *IEEE Transactions on Services Computing*, vol. 12, no. 2, pp. 319–334, Mar. 2019, ISSN: 1939-1374. DOI: 10.1109/TSC.2016.2634024. [Online]. Available: http://dx.doi.org/10.1109/TSC.2016.2634024.

[27] V. D. Justafort, R. Beaubrun, and S. Pierre, "A hybrid approach for optimizing carbon footprint in intercloud environment," *IEEE Transactions on Services Computing*, vol. 12, no. 2, pp. 186–198, Mar. 2019, ISSN: 1939-1374. DOI: 10.1109/TSC.2016.2638900. [Online]. Available: http://dx.doi.org/10.1109/TSC.2016.2638900.

[28] D. Chappell, "Introducing OData data access for the web, the cloud, mobile devices, and more," Chappell & Associates, Tech. Rep., May 2011.

[29] ISO 20802-1:2016, "Information technology — Open data protocol (OData) v4.0 — Part 1: Core," International Organization for Standardization, Geneva, CH, Standard, Dec. 2016.

[30] OpenAPI, "OpenAPI specification," SmartBear Sofrware, Specification, version 3.0.3, Feb. 2020. [Online]. Available: https://swagger.io/specification/.

[31] GraphQL, "GraphQL specification," Facebook Inc., Specification, Jun. 2018. [Online]. Available: https://spec.graphql.org/June2018.

[32] AsyncAPI, "AsyncAPI specification," Specification, version 2.0.0, 2020. [Online]. Available: https://www.asyncapi.com/docs/specifications/2.0.0.

[33] M. Gudgin, M. Hadley, N. Mendelsohn, *et al.*, "SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)," World Wide Web Consortium, W3C Recommendation, Apr. 2007. [Online]. Available: https://www.w3.org/TR/soap12/.

[34] H. El-Rewini and M. Abd-El-Barr, *Advanced Computer Architecture and Parallel Processing*, ser. Wiley Series on Parallel and Distributed Computing. Wiley, 2005, ISBN: 9780471478393. [Online]. Available: https://books.google.com/books?id=7JB-u6D5Q7kC.

[35] R. Fielding (Ed.) and J. Reschke (Ed.), *Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing*, RFC 7230 (Proposed Standard), RFC, Updated by RFC 8615, Fremont, CA, USA: RFC Editor, Jun. 2014. DOI: 10.17487/RFC7230. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7230.txt.

[36] R. Fielding (Ed.) and J. Reschke (Ed.), *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content*, RFC 7231 (Proposed Standard), RFC, Fremont, CA, USA: RFC Editor, Jun. 2014. DOI: 10.17487/RFC7231. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7231.txt.

[37] S. M. Sohan, F. Maurer, C. Anslow, and M. P. Robillard, "A study of the effectiveness of usage examples in REST API documentation," in *2017 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, Oct. 2017, pp. 53–61. DOI: 10.1109/VLHCC.2017.8103450. [Online]. Available: http://dx.doi.org/10.1109/VLHCC.2017.8103450.

[38] R. Fielding (Ed.) and J. Reschke (Ed.), *Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests*, RFC 7232 (Proposed Standard), RFC, Fremont, CA, USA: RFC Editor, Jun. 2014. DOI: 10.17487/RFC7232. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7232.txt.

[39] J. Richer (Ed.), M. Jones, J. Bradley, M. Machulak, and P. Hunt, *OAuth 2.0 Dynamic Client Registration Protocol*, RFC 7591 (Proposed Standard), RFC, Fremont, CA, USA: RFC Editor, Jul. 2015. DOI: 10.17487/RFC7591. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7591.txt.

[40] N. Sakimura, J. Bradley, M. Jones, B. de Medeiros, and C. Mortimore, *OpenID Connect Core 1.0*, Nov. 2014. [Online]. Available: https://openid.net/specs/openid-connect-core-1%5C_0.html.

[41] C. Huitema, J. Postel, and S. Crocker, *Not All RFCs are Standards*, RFC 1796 (Informational), RFC, Fremont, CA, USA: RFC Editor, Apr. 1995. DOI: 10.17487/RFC1796. [Online]. Available: https://www.rfc-editor.org/rfc/rfc1796.txt.

[42] R. Fielding (Ed.), Y. Lafon (Ed.), and J. Reschke (Ed.), *Hypertext Transfer Protocol (HTTP/1.1): Range Requests*, RFC 7233 (Proposed Standard), RFC, Fremont, CA, USA: RFC Editor, Jun. 2014. DOI: 10.17487/RFC7233. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7233.txt.

[43] R. Fielding (Ed.), M. Nottingham (Ed.), and J. Reschke (Ed.), *Hypertext Transfer Protocol (HTTP/1.1): Caching*, RFC 7234 (Proposed Standard), RFC, Fremont, CA, USA: RFC Editor, Jun. 2014. DOI: 10.17487/RFC7234. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7234.txt.

[44] R. Fielding (Ed.) and J. Reschke (Ed.), *Hypertext Transfer Protocol (HTTP/1.1): Authentication*, RFC 7235 (Proposed Standard), RFC, Fremont, CA, USA: RFC Editor, Jun. 2014. DOI: 10.17487/RFC7235. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7235.txt.

[45] J. Reschke, *Initial Hypertext Transfer Protocol (HTTP) Authentication Scheme Registrations*, RFC 7236 (Informational), RFC, Fremont, CA, USA: RFC Editor, Jun. 2014. DOI: 10.17487/RFC7236. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7236.txt.

[46] J. Reschke, *Initial Hypertext Transfer Protocol (HTTP) Method Registrations*, RFC 7237 (Informational), RFC, Fremont, CA, USA: RFC Editor, Jun. 2014. DOI: 10.17487/RFC7237. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7237.txt.

[47] E. Rescorla, *HTTP Over TLS*, RFC 2818 (Informational), RFC, Updated by RFCs 5785, 7230, Fremont, CA, USA: RFC Editor, May 2000. DOI: 10.17487/RFC2818. [Online]. Available: https://www.rfc-editor.org/rfc/rfc2818.txt.

[48] D. Hardt (Ed.), *The OAuth 2.0 Authorization Framework*, RFC 6749 (Proposed Standard), RFC, Updated by RFC 8252, Fremont, CA, USA: RFC Editor, Oct. 2012. DOI: 10.17487/RFC6749. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6749.txt.

[49] T. Bray (Ed.), *The JavaScript Object Notation (JSON) Data Interchange Format*, RFC 8259 (Internet Standard), RFC, Fremont, CA, USA: RFC Editor, Dec. 2017. DOI: 10.17487/RFC8259. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8259.txt.

[50] M. Jones, J. Bradley, and N. Sakimura, *JSON Web Signature (JWS)*, RFC 7515 (Proposed Standard), RFC, Fremont, CA, USA: RFC Editor, May 2015. DOI: 10.17487/RFC7515. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7515.txt.

[51] M. Jones, J. Bradley, and N. Sakimura, *JSON Web Token (JWT)*, RFC 7519 (Proposed Standard), RFC, Updated by RFCs 7797, 8725, Fremont, CA, USA: RFC Editor, May 2015. DOI: 10.17487/RFC7519. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7519.txt.

[52] M. Jones, *JSON Web Key (JWK)*, RFC 7517 (Proposed Standard), RFC, Fremont, CA, USA: RFC Editor, May 2015. DOI: 10.17487/RFC7517. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7517.txt.

[53] M. Nottingham, *Well-Known Uniform Resource Identifiers (URIs)*, RFC 8615 (Proposed Standard), RFC, Fremont, CA, USA: RFC Editor, May 2019. DOI: 10.17487/RFC8615. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8615.txt.

[54] H. Butler, M. Daly, A. Doyle, S. Gillies, S. Hagen, and T. Schaub, *The GeoJSON Format*, RFC 7946 (Proposed Standard), RFC, Fremont, CA, USA: RFC Editor, Aug. 2016. DOI: 10.17487/RFC7946. [Online]. Available: https://www.rfc-editor.org/rfc/rfc7946.txt.

[55] I. Fette and A. Melnikov, *The WebSocket Protocol*, RFC 6455 (Proposed Standard), RFC, Updated by RFCs 7936, 8307, 8441, Fremont, CA, USA: RFC Editor, Dec. 2011. DOI: 10.17487/RFC6455. [Online]. Available: https://www.rfc-editor.org/rfc/rfc6455.txt.

[56] A. Neumann, N. Laranjeiro, and J. Bernardino, "An analysis of public REST web service APIs," *IEEE Transactions on Services Computing*, pp. 1–1, 2018, ISSN: 1939-1374. DOI: 10.1109/TSC.2018.2847344. [Online]. Available: https://dx.doi.org/10.1109/TSC.2018.2847344.

[57] OADA, *OADA API reference implementation*. [Online]. Available: https://github.com/OADA/server.

[58] OADA, *Open Ag Data Alliance*. [Online]. Available: https://github.com/OADA.

[59] Joyent, Inc., *Node.js®*. [Online]. Available: https://nodejs.org.

[60] Docker, Inc., *Docker*. [Online]. Available: https://www.docker.com.

[61] Apache Software Foundation, *Apache Kafka*. [Online]. Available: https://kafka.apache.org.

[62] ArangoDB, Inc., *ArangoDB*. [Online]. Available: https://www.arangodb.com.

[63] I. Sysoev and NGINX, Inc., *NGINX*. [Online]. Available: https://www.nginx.org.

[64] OpenATK, *FieldWorkApp*. [Online]. Available: https://github.com/OpenATK/FieldWorkApp.

[65] OpenATK, *TrialsTracker*. [Online]. Available: https://github.com/OpenATK/TrialsTracker.

[66] OADA, *OADA cache*. [Online]. Available: https://github.com/OADA/oada-cache.

[67] A. W. Layton, S. Noel, Y. Wang, *et al.*, "The Trellis framework for automatic food safety data transfer," in *2018 ASABE Annual International Meeting*, American Society of Agricultural and Biological Engineers, 2018, p. 1. DOI: 10.13031/aim.201801248. [Online]. Available: https://dx.doi.org/10.13031/aim.201801248.

[68] Trellis Framework, *Trellis mask & link*. [Online]. Available: https://github.com/trellisfw/trellisfw-masklink.

[69] OpenATK, *ISOBlueApp*. [Online]. Available: https://github.com/OpenATK/ISOBlueApp.

[70] A. W. Layton, A. D. Balmos, S. Sabpisal, A. Ault, J. V. Krogmeier, and D. Buckmaster, "ISOBlue: An open source project to bring agricultural machinery data into the cloud," in *2014 ASABE Annual International Meeting*, Jul. 2014. DOI: 10.13031/aim.20141929380. [Online]. Available: http://dx.doi.org/10.13031/aim.20141929380.

[71] Y. Wang, H. Liu, J. Krogmeier, A. Reibman, and D. Buckmaster, "ISOBlue HD: An open-source platform for collecting context-rich agricultural machinery datasets," en, *Sensors*, vol. 20, no. 20, p. 5768, Oct. 2020, ISSN: 1424-8220. DOI: 10.3390/s20205768. [Online]. Available: https://dx.doi.org/10.3390/s20205768.

[72] G. Niemeyer, *Geohash*, 2008. [Online]. Available: http://geohash.org.

# A. PROOF THAT HMRF MODEL IS AN EXPONENTIAL FAMILY

**Theorem A.0.1.** *The joint model of $X$ and $Y$ forms an exponential family of distributions parameterized by $\theta$. That is, the joint distribution can be written in the following form from the definition of an exponential family.*

$$P(Y, X \mid \theta) = b(Y, X) \exp\left\{\langle \eta(\theta), T(Y, X)\rangle\right\} / \alpha(\theta) \tag{A.1}$$

*Proof.* From Bayes' theorem, the joint model can be broken down thusly.

$$P(Y, X \mid \theta) = P(Y \mid X, \theta) P(X) \tag{A.2}$$

First, writing out the conditional probability density function (pdf) of $Y$ given $X$ as the product of the conditionally independent Gaussians and then grouping by management zone yields the following,

$$\begin{aligned} P(Y \mid X, \theta) &= \prod_{s \in S} P(Y_s \mid X_s, \theta) \\ &= \prod_{s \in S} (2\pi)^{-\frac{P}{2}} |R_{X_s}|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(Y_s - \mu_{X_s})^\intercal R_{X_s}^{-1} (Y_s - \mu_{X_s})\right\} \\ &= (2\pi)^{-\frac{P|S|}{2}} \prod_{k=0}^{K-1} \prod_{s \in S(k)} |R_k|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(Y_s - \mu_k)^\intercal R_k^{-1} (Y_s - \mu_k)\right\} \end{aligned} \tag{A.3}$$

where

$$S(k) \triangleq \{s : s \in S,\, X_s = k\}.$$

Focusing for now on the part inside $\prod_{k=0}^{K-1}$ we change the product over $s$ to a sum in the exponent.

$$\prod_{s \in S(k)} |R_k|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(Y_s - \mu_k)^\intercal R_k^{-1} (Y_s - \mu_k)\right\} = \exp\left\{\sum_{s \in S(k)} -\frac{1}{2}\left((Y_s - \mu_k)^\intercal R_k^{-1} (Y_s - \mu_k) + \log|R_k|\right)\right\} \quad \text{(A.4)}$$

Then multiplying out the quadratic term in the sum yields,

$$\begin{aligned}(Y_s - \mu_k)^\intercal R_k^{-1} (Y_s - \mu_k) &= Y_s^\intercal R_k^{-1} Y_s - 2Y_s^\intercal R_k^{-1}\mu_k + \mu_k^\intercal R_k^{-1}\mu_k \\ &= \langle Y_s Y_s^\intercal, R_k^{-1}\rangle - 2Y_s^\intercal R_k^{-1}\mu_k + \mu_k^\intercal R_k^{-1}\mu_k \quad \text{(A.5)}\end{aligned}$$

because $Y_s^\intercal R_k^{-1} Y_s$ is a scalar, and thus

$$\begin{aligned}Y_s^\intercal R_k^{-1} Y_s &= \mathrm{Tr}\left\{Y_s^\intercal R_k^{-1} Y_s\right\} \\ &= \mathrm{Tr}\left\{Y_s Y_s^\intercal R_k^{-1}\right\} \\ &= \langle Y_s Y_s^\intercal, R_k^{-1}\rangle\end{aligned}$$

.

Plugging (A.5) and (A.4) into (A.3) yields,

$$P(Y \mid X, \theta) = (2\pi)^{-\frac{P|S|}{2}} \prod_{k=0}^{K-1} \exp\left\{-\frac{1}{2}\left(\langle S_k, R_k^{-1}\rangle - 2b_k^\intercal R_k^{-1}\mu_k + N_k\mu_k^\intercal R_k^{-1}\mu_k + N_k \log|R_k|\right)\right\} \quad \text{(A.6)}$$

where

$$N_k \triangleq |S(k)| \qquad = \sum_{s \in S} \delta(X_s = k) \tag{A.7}$$

$$b_k \triangleq \sum_{s \in S(k)} Y_s \quad = \sum_{s \in S} Y_s \delta(X_s = k) \tag{A.8}$$

$$S_k \triangleq \sum_{s \in S(k)} Y_s Y_s^\intercal = \sum_{s \in S} Y_s Y_s^\intercal \delta(X_s = k). \tag{A.9}$$

Finally, combining (A.6) and (A.2) one can obtain the form of (A.1) with

$$b(Y, X) = P(X)$$

$$\eta(\theta) = \begin{bmatrix} -\frac{1}{2}\left(\log|R_0| + \mu_0^\intercal R_0^{-1}\mu_0\right) \\ R_0^{-1}\mu_0 \\ -\frac{1}{2}R_0^{-1} \\ \vdots \\ -\frac{1}{2}\left(\log|R_{K-1}| + \mu_{K-1}^\intercal R_{K-1}^{-1}\mu_{K-1}\right) \\ R_{K-1}^{-1}\mu_{K-1} \\ -\frac{1}{2}R_k^{-1} \end{bmatrix} \tag{A.10}$$

$$T(Y, X) = \begin{bmatrix} N_0 \\ b_0 \\ S_0 \\ \vdots \\ N_{K-1} \\ b_{K-1} \\ S_{K-1} \end{bmatrix} \tag{A.11}$$

$$\alpha(\theta) = (2\pi)^{\frac{P|S|}{2}}. \tag{A.12}$$

Therefore, the joint model of $X$ and $Y$ forms an exponential family of distributions parameterized by $\theta$. □

# B. DERIVATION OF EM STEP EQUATIONS

As is proven in Appendix A, the model used is an exponential family of $\theta$, i.e., the parameters being estimated. Therefore, as shown in [16], the expectation-maximization (EM) updates for maximum likelihood (ML) parameter estimation equate to the following. For the E-step, compute the expected value of the sufficient statistics. For the M-step, compute the ML estimate of $\theta$ replacing the statics $T(X, Y)$ with their expected values.

## B.1 E-Step

According to the form of an exponential family (A.1), the sufficient statistics $T(Y, X)$, are shown in (A.11). Therefore, their expected value is,

$$E[T(Y,X) \mid Y,\theta] = \left[\bar{N}_0, \;\; \bar{b}_0, \;\; \bar{S}_0, \;\; \cdots, \;\; \bar{N}_{K-1}, \;\; \bar{b}_{K-1}, \;\; \bar{S}_{K-1}\right]^{\intercal} \tag{B.1}$$

where

$$\bar{N_k} \triangleq E[N_k \mid Y,\theta] = E\left[\sum_{s\in S} \delta(X_s = k) \mid Y,\theta\right] \tag{B.2}$$

$$\bar{b_k} \triangleq E[b_k \mid Y,\theta] = E\left[\sum_{s\in S} Y_s \delta(X_s = k) \mid Y,\theta\right] \tag{B.3}$$

$$\bar{S_k} \triangleq E[S_k \mid Y,\theta] = E\left[\sum_{s\in S} Y_s Y_s^{\intercal} \delta(X_s = k) \mid Y,\theta\right] \tag{B.4}$$

are the expected values of the individual statistics for each management zone $k$.

Thus, the E-step for the model is to evaluate (B.2), (B.3), and (B.4) for $k = 0, \ldots, K-1$ using the current estimate of $\theta$.

## B.2 M-Step

From (A.1) and because (A.12) does not depend on $\theta$, the ML estimate of $\theta$ can be expressed as,

$$\begin{aligned}\hat{\theta}_{ML} &= \arg\max_{\theta} \log P(Y, X \mid \theta) \\ &= \arg\max_{\theta} \langle \eta(\theta), T(Y, X) \rangle \end{aligned} \tag{B.5}$$

where $\eta(\theta)$ and $T(Y, X)$ are given in (A.10) and (A.11) respectively.

Fist, taking the derivative of the function being maximized in (B.5) w.r.t. $\mu_k$ yields

$$\frac{\partial}{\partial \mu_k} \left\{ -\frac{1}{2} \left( \log |R_k| + \mu_k{}^\intercal R_k^{-1} \mu_k \right) N_k + b_k^\intercal R_k^{-1} \mu_k \right\} = -R_k^{-1} \mu_k N_k + R_k^{-1} b_k \tag{B.6}$$

for all $k$. Then, setting the above equal to 0, and solving for $\mu_k$ gives

$$\hat{\mu}_k^{ML} = \frac{1}{N_k} b_k \tag{B.7}$$

for the ML estimate of $\mu_k$.

Next, taking the derivative of the function being maximized in (B.5) w.r.t. $R_k$ yields

$$\begin{aligned}\frac{\partial}{\partial R_k} \left\{ -\frac{1}{2} \left( \log |R_k| + \mu_k^\intercal R_k^{-1} \mu_k \right) N_k + b_k^\intercal R_k^{-1} \mu_k - \frac{1}{2} \operatorname{Tr} \left\{ S_k R_k^{-1} \right\} \right\} = \\ - \frac{1}{2} \left( 2 R_k^{-1} - R_k^{-1} \circ I - R_k^{-1} \mu_k \mu_k^\intercal R_k^{-1} \right) N_k - R_k^{-1} b_K \mu_k^\intercal R_k^{-1} + \frac{1}{2} \left( R_k^{-1} S_k R_k^{-1} \right)\end{aligned} \tag{B.8}$$

for all $k$. Then, setting the above equal to 0, plugging in (B.7) for $\mu_k$, and solving for $R_k$ gives

$$\hat{R}_k^{ML} = \frac{1}{N_k} S_k - \frac{1}{N_k^2} b_k b_k^\intercal \tag{B.9}$$

for the ML estimate of $R_k$.

Thus, the M-step for the model is to evaluate

$$\hat{\mu}_k = \frac{1}{\bar{N}_k} \bar{b}_k \tag{B.10}$$

$$\hat{R}_k = \frac{1}{\bar{N}_k} \bar{S}_k - \frac{1}{\bar{N}_k^2} \bar{b}_k \bar{b}_k^\intercal \tag{B.11}$$

for $k = 0, \ldots, K-1$.

# C. GIBBS SAMPLER

The Gibbs sampler allows generating realizations of the Markov random field (MRF) $X$ that follow the probability mass function (pmf) $P(X|Y,\theta)$ while only having to evaluate the simpler pmf $P(X_s \mid X_{\partial s}, Y_s, \theta)$. The derivation of an expression for this conditional pmf and the specifics of the Gibbs sampling done here are described in the following sections.

## C.1 Conditional pmf of $X_s$

A simple expression for the conditional marginal of $X_s$ needs to be found. Starting with Bayes' rule, we get

$$P(X_s \mid Y_s, \theta) = P(Y_s|X_{\partial s}, \theta)^{-1} P(X_s \mid X_{\partial s}) P(Y_s \mid X_s, X_{\partial s})$$

then from our hidden Markov random field (HMRF) structure, we get

$$= P(Y_s|X_{\partial s}, \theta)^{-1} P(X_s \mid X_{\partial s}) P(Y_s \mid X_s)$$

and finally, from the Potts pmf of (3.2)

$$= P(Y_s|X_{\partial s}, \theta)^{-1} z(\beta, X_{\partial s})^{-1} \mathrm{e}^{-\beta \sum_{r \in \partial s} b_{|s-r|} \delta(X_r \neq X_s)} P(Y_s \mid X_s). \tag{C.1}$$

Since the first two terms in (C.1) do not depend on $X_s$ and this pmf will only be used to compute a cumulative distribution function (CDF), these terms can be ignored. Thus,

$$P(X \mid Y, \theta) \propto \mathrm{e}^{-\beta \sum_{r \in \partial s} b_{|s-r|} \delta(X_r \neq X_s)} P(Y_s \mid X_s) \tag{C.2}$$

can be used as the improper pmf when implementing the Gibbs sampler, where $P(Y_s|X_s)$ is a multivariate Gaussian pdf as given by (3.4).

## C.2 Configuration Specifics

The specific Gibbs sampler used in this work actually consists of 10 parallel sampling chains. The separate sampling chains were employed to decrease runtime. By using a GPU all 10 can be run simultaneously in roughly the same time it would take to run 1. Each chain is essentially its own simpler Gibbs sampler that is initialized with its own random starting point. When all 10 chains are done, the samples are concatenated and returned. The pseudocode for one such sampler is shown in Algorithm 1.

**Algorithm 1** Single Chain Gibbs sampler

**Require:** $Y, \theta, K$
  **for all** $s \in S$ **do**
    Draw $X_s^{(0)} \sim \mathcal{U}[0, K-1]$
  **end for**
  **for** $l = 1, \ldots, L$ **do**
    $X^{(l)} \leftarrow X^{(l-1)}$
    **for all** $s \in S$ **do**
      Compute CDF $F(x_s) = P(X_s \leq x_s \mid X_{\partial s}^{(l)}, Y_s, \theta)$
      Draw $u \sim \mathcal{U}(0, 1)$
      Update $X_s^{(l)} \leftarrow F^{-1}(u)$
    **end for**
  **end for**
  **return** $X^{(1)}, \ldots, X^{(L)}$

At the start of each chain, each element of $X^{(0)}$ is initialized with an independent and identically distributed (i.i.d.) uniform random integer in the range $[0, K-1]$. Then, in a loop, each element of $X$ is randomly updated using the inverse CDF method to drawn from its marginal distribution. After the loop is done, the current value of the $X$ is taken as a sample. This loop is repeated $L$ times, where here $L$ is the number of samples from the given chain. Since there are 10 sampling chains and 10,000 samples are wanted, the $L$ for each chain is 1,000.

# VITA

Alex Layton was born in Indianapolis, Indiana on July 4th 1989. Alex works as a software engineer as a partner of a small company. Alex received a B.S. in computer engineering from Purdue University, West Lafayette, Indiana in May 2012. Alex was a member of a finalist team in the 2014 DARPA spectrum challenge. Alex received the Bilsland Dissertation Fellowship in 2018. Research interests include statistical modeling, machine learning, and RESTful web services.

# PUBLICATIONS

- A. Layton, T. Arakawa, S. Noel, et al., "The OADA API: An ag-inspired intercloud REST-like API framework for increasing data freedom and interoperability," IEEE Transactions on Services Computing, (manuscript submitted).
- A. Layton, J. V. Krogmeier, A. Alut, and D. R. Buckmaster, "From yield history to management zone identification with hidden Markov random fields," Precision Agriculture, vol. 21, pp. 762–781, Aug. 2020. doi: 10.1007/s11119-019-0694-2.
- A. W. Layton, S. Noel, Y. Wang, et al., "The Trellis framework for automatic food safety data transfer," in 2018 ASABE Annual International Meeting, American Society of Agricultural and Biological Engineers, 2018, p. 1.
- A. W. Layton, Y. Zhang, J. V. Krogmeier, and D. R. Buckmaster, "Determining harvesting efficiency via multiple combine GPS logs," in 2017 ASABE Annual International Meeting, American Society of Agricultural and Biological Engineers, 2017, p. 1.
- Y. Wang, A. D. Balmos, A. W. Layton, et al., "An open-source infrastructure for real-time automatic agricultural machine data processing," in 2017 ASABE Annual International Meeting, American Society of Agricultural and Biological Engineers, 2017, p. 1.
- J. Grossman, A. Layton, J. Krogmeier, and D. M. Bullock, "Traffic signal detector error identification using Kolmogorov-Smirnov test," in Transportation Research Board 95th Annual Meeting, 2016.
- Y. Wang, A. D. Balmos, A. W. Layton, et al., "CANdroid: Freeing ISOBUS data and enabling machine data analytics," in 2016 ASABE Annual International Meeting, American Society of Agricultural and Biological Engineers, 2016, p. 1.
- A. W. Layton, Y. Wang, J. V. Krogmeier, and D. Buckmaster, "Robust estimation of field management zones using multi-year yield data and a hidden Markov random

field," in 2016 ASABE Annual International Meeting, American Society of Agricultural and Biological Engineers, 2016, p. 1. 117

- A. D. Balmos, A. W. Layton, A. Ault, J. V. Krogmeier, and D. R. Buckmaster, "Toward understanding the errors in online air-ride suspension based weight estimation," in 2014 Montreal, Quebec Canada July 13–July 16, 2014, American Society of Agricultural and Biological Engineers, 2014, p. 1.

- J. Y. Kim, A. C. Marcum, A. D. Balmos, et al., "Implementation and analysis of energy detection-based sensing using USRP/SBX platform," in Military Communications Conference (MILCOM), 2014 IEEE, IEEE, 2014, pp. 1504–1509.

- A. C. Marcum, A. D. Balmos, S. G. Larew, et al., "Low SINR synchronization for the DARPA spectrum challenge scenario," in Military Communications Conference (MILCOM), 2014 IEEE, IEEE, 2014, pp. 1447–1453.

- A. W. Layton, A. D. Balmos, S. Sabpisal, A. Ault, J. V. Krogmeier, and D. Buckmaster, "ISOBlue: An open source project to bring agricultural machinery data into the cloud," in 2014 ASABE and CSBE/SCGAB Annual International Meeting, 2014.

- A. D. Balmos, A. W. Layton, A. Ault, J. V. Krogmeier, and D. R. Buckmaster, "Investigation of bluetooth communications for low-power embedded sensor networks in agriculture," in 2013 Kansas City, Missouri, July 21-July 24, 2013, American Society of Agricultural and Biological Engineers, 2013, p. 1.

- J. Welte, A. Ault, C. Bowman, et al., "Autogenic mobile computing technologies in agriculture: Applications and sensor networking for smart phones and tablets," in 2013 EFITA Conference, 2013.

- A. W. Layton, A. D. Balmos, D. L. Hancock, A. C. Ault, J. V. Krogmeier, and D. R. Buckmaster, "Wireless load weight monitoring via a mobile device based on air suspension pressure," in 2012 ASABE Annual International Meeting, 2012.