Technical Report
1238

Probabilistic Programming with Missing Data

D. Suen
K.L. Nahabedian
M.J.Yee
M.B. Hurley

26 February 2019

Lincoln Laboratory
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
LEXINGTON, MASSACHUSETTS

This material is based upon work supported by the Assistant Secretary of Defense for Research and Engineering under Air Force Contract No. FA8702-15-D-0001.

DISTRIBUTION STATEMENT A. Approved for public release: distribution unlimited.

This report is the result of studies performed at Lincoln Laboratory, a federally funded research and development center operated by Massachusetts Institute of Technology. This material is based upon work supported by the Assistant Secretary of Defense for Research and Engineering under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Assistant Secretary of Defense for Research and Engineering.

© 2018 Massachusetts Institute of Technology.

Delivered to the U.S. Government with Unlimited Rights, as defined in DFARS Part 252.227-7013 or 7014 (Feb 2014). Notwithstanding any copyright notice, U.S. Government rights in this work are defined by DFARS 252.227-7013 or DFARS 252.227-7014 as detailed above. Use of this work other than as specifically authorized by the U.S. Government may violate any copyrights that exist in this work.

Massachusetts Institute of Technology
Lincoln Laboratory

Probabilistic Programming with Missing Data

D. Suen
K.L. Nahabedian
M.J. Yee
M.B. Hurley

Group 104

Technical Report 1238

26 February 2019

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited.

Lexington Massachusetts

This page intentionally left blank.

ACKNOWLEDGMENTS

The authors would like to thank the group office leadership: Paul Metzger, Ben Landon, and Sanjeev Mohindra for their guidance of the program through the year. We would also like to thank Ken Senne for his tireless devotion to our group's summer internship program to identify and recruit young talent who will form the backbone of the U. S. technical workforce in future years.

This page intentionally left blank.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Page

LIST OF FIGURES

LIST OF FIGURES

(Continued)

LIST OF TABLES

This page intentionally left blank.

ABSTRACT

This report summarizes work performed to develop a computer algorithm that is capable of handling missing data fields in multivariate data sets. The results presented here are based upon prior work which examined the applicability of inverse covariance matrices, or precision matrices, to representing missing data as zero eigenvalues in the precision matrices. The prior work used maximum a posteriori (MAP) estimates for a combination of normally distributed multivariate data with normally distributed multivariate measurement errors and assumed that the prior probability distributions for means and precision matrices were uniform.

This work extends the previous technique to one that uses normal-Wishart matrices to describe the prior probability distribution for a normal data distribution and to estimate posterior parameters for these distributions for multivariate data with missing fields. While the integrals to estimate posterior probabilities from likelihood and prior probability distributions may in fact be analytically solvable, the authors were unable to discover such a solution. Instead, analytic integral solutions were used for individual probability measurements and a probabilistic programming language was used to perform numerical integration on the remaining integrals. The chosen solution leverages the strengths of one integration method to address the weakness of the complimentary method: analytic integration is performed where numerical integration cannot be performed, and numerical integration is performed where analytic solutions are currently unknown. Most of the recent work was performed by an undergraduate intern for Group 104 at MIT Lincoln Laboratory during the Summer of 2018. A model for the problem is defined and analyzed mathematically. A discussion of the probabilistic programming languages and programs is also provided, along with results for a number of simulations.

This page intentionally left blank.

1. INTRODUCTION

As the state of the art in data analytics moves to even larger data sets with multi-modal features of greater number of dimensions, the problem of missing data fields is bound to become more prevalent. Fields like medical and social sciences research have struggled with the missing data problem for decades. Solutions have included Multiple Imputation (MI) [1, 2] and Full Information Maximum Likelihood (FIML) [3, 4]. The FIML technique uses a set of boolean values to indicate missing data fields, while the technique developed by the authors uses inverse covariance matrices, or precision matrices (also called information matrices), to represent missing data fields; zero eigenvalues in the appropriate dimensions of the precision matrix represent missing data fields. The technique described in this report can be considered a generalization of FIML, in that the measurement coordinate systems' principal axes are no longer required to be the same for all measurements. Individual measurements can have different principal axes with different subsets of axes containing no information.

From a Bayesian perspective, prior work by the authors ignored the prior distribution functions for the mean and precision matrix. In other words, uniform distribution functions were assumed for the prior distribution for the mean and precision matrix of the data distribution. The prior work also used a maximum likelihood method to estimate a cluster's mean and covariance matrix. This work extends the previous work to use normal-Wishart prior probability distributions for the mean and precision matrix of a cluster and to determine posterior estimates of the expectation values for the mean and precision matrix from the distribution of the data. The normal-Wishart distribution is the conjugate prior for the multivariate normal distribution when precision matrices are adopted instead of covariance matrices. While the model for data generation consists of Gaussian and Wishart distributions, complete analytic integration of the density functions was found to be intractable. This project examined the possibility of using a probabilistic programming language to perform the numerical integrations needed to estimate expectation values for means and covariance matrices. The utility of the approach was demonstrated with a simple data set clustered with a k-means algorithm that relied on a probabilistic program to estimate association likelihoods between measurements and clusters and to update the Gaussian model parameters of the clusters.

1.1 MOTIVATION AND OBJECTIVES

Over the course of a summer internship, Daniel Suen was tasked to select a suitable probabilistic programming language for the posterior estimation of a cluster's mean and precision matrix from multivariate data with missing parameters. Stan, Edward, and PyMC3 were evaluated for potential use. Probabilistic programs were then to be written such that they could be integrated into a k-means style clustering algorithm to demonstrate the effectiveness of the programs [5–7]. The k-means algorithm consists of two iterated steps: assigning measurements to clusters and estimating the means and precision matrices for each cluster from the assigned measurements. The two steps are repeated until measurements are consistently assigned to the same cluster.

As data sets become increasingly large, it becomes more and more difficult to obtain complete data. For instance, traditional statistical data analyses have often been plagued with problems

such as poor response rates (surveys) and censorship (medical data, survival analysis). Various techniques have emerged to work with the data that have been given. This summer project sought to use the Bayesian approach of probabilistic programs to avoid analytically evaluating integrals that are too challenging to solve. The project was primarily an exploratory one to evaluate the power of Probabilistic Programming Languages (**PPLs**) and how they could be applied to this type of problem.

For readers with previous experience with Markov Chain Monte Carlo (**MCMC**), this report includes a lot of beginner topics that may be uninteresting. A short overview of MCMC is provided for those readers with little exposure to this topic. The report first introduces the generative model that described how data are imagined to be generated and moves to a discussion on the PPLs used. Afterwards, it outlines a variant of the k-means clustering algorithm and presents empirical results. Appendices are provided which contain summary derivations of the update equations associated with the probabilistic programs.

1.2 GENERATIVE MODEL

The generative model for the specific problem of interest for this summer project can be described with Figure 1. Using a probabilistic approach, a given cluster can be represented via a PDF, say a multivariate normal (MVN) distribution with unknown mean vector μ and covariance matrix Σ_μ. Next, a state x_i is sampled from this distribution, generating a measurement for that cluster.

For each x_i, zero-mean additive Gaussian noise is added to model the sensor measurement noise. Under this model, each sensor can have a unique covariance matrix Σ_{z_i} associated with its measurement error, and only z_i is strictly observed.

In other words, the generative model is a convolution of two Gaussian distributions: a distribution from which true sample measurements are drawn and an error model to introduce noise into the actual measurements that are observed. The core aspect of the problem is thus estimating the cluster distribution parameters μ and Σ_μ in the presence of sensor measurement error and missing features. To accomplish this end, Bayesian techniques were adopted to enforce various priors of the form $p(\mu, \Sigma_\mu)$ that represent prior beliefs about the cluster distribution parameters.

Although the generative model uses covariance matrices, the algorithms that were developed use precision matrices for a majority of the calculations. This is especially true for the measurement error matrices Σ_{z_i}. Precision matrices Λ_{z_i} are used instead. For this analysis, it was assumed that the measurement error precision matrices are known. The problem is insolvable if the measurement error matrices are unique and unknown for each measurement. It is possible to solve the problem if the measurement error matrix is the same for all samples. This problem is not studied here because the focus is on developing algorithms that can handle missing data fields, as represented with the individual error matrices.

Throughout the report, the derivations will switch between Σ and Λ notation for covariance and precision matrices, depending on context and convenience. Note that $\Sigma = \Lambda^{-1}$ where the inverse exists.

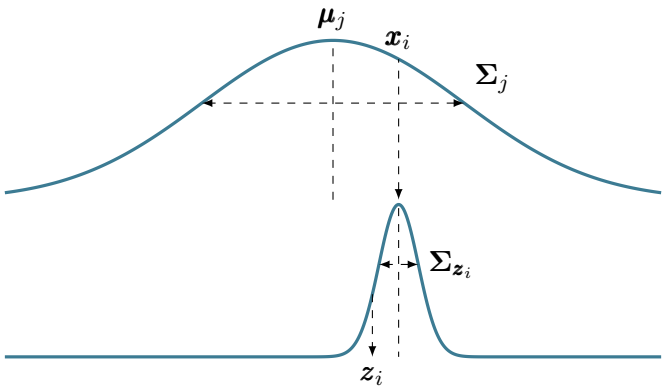

Figure 1. A graphical depiction of the generative model for producing observed samples drawn from a normal distribution of true sample measurements corrupted by sensor noise.

1.3 INTEGRALS FOR THE MODEL

Relying on Bayes' rule, the posterior distribution function for the mean and covariance matrix of the underlying distribution of the measurements is

$$P\left(\mu, \Lambda_\mu | \left\{z_i, x_i; \Lambda_{z_i}\right\}\right) = \frac{P\left(\left\{z_i, x_i; \Lambda_{z_i}\right\} | \mu, \Lambda_\mu\right) P\left(\mu, \Lambda_\mu\right)}{P\left(\left\{z_i, x_i; \Lambda_{z_i}\right\}\right)}. \tag{1}$$

For this model, it is assumed that z_i is only dependent on the cluster parameters μ and $\Sigma_\mu = \Lambda_\mu^{-1}$ through x_i. Some of the notation is relaxed in the following derivations to limit the complexity of the typesetting; for instance, the integrals should be from $-\infty$ to ∞ over all dimensions. It is also assumed that the Λ_{z_i} are known, and are thus fixed parameters rather than random variables. The generative model defines the probability distributions $p(z_i|x_i; \Sigma_{z_i})$ for a specific measurement and $p(x_i|\mu; \Sigma_\mu)$ for a specific cluster. In addition, this report explores the use of the Normal-Wishart prior distribution for this generative model. To implement a k-means style algorithm, estimates of the parameters μ and Σ_μ for particular clusters are desired.

If the posterior probability distribution $P\left(\mu, \Lambda_\mu | \left\{z_i, x_i; \Lambda_{z_i}\right\}\right)$, is available, the expectation values for μ and Σ_μ can be determined by the integrals

$$\langle \mu \rangle = \int_{-\infty}^{\infty} \mu P\left(\mu, \Lambda_\mu | \left\{z_i, x_i; \Lambda_{z_i}\right\}\right) d\mu \; d\Lambda_\mu, \tag{2}$$

$$\langle \Sigma_\mu \rangle = \int_{-\infty}^{\infty} \left(\mu - \langle \mu \rangle\right) \left(\mu - \langle \mu \rangle\right)^\intercal P\left(\mu, \Lambda_\mu | \left\{z_i, x_i; \Lambda_{z_i}\right\}\right) d\mu \; d\Lambda_\mu. \tag{3}$$

These integrals can be used to determine cluster parameters from the associated measurements.

Given the assumption that measurements are independent, other than through μ and Λ_μ, the probability for sets of measurements can be reformulated into a product of probabilities,

$$P\left(\left\{z_i, x_i; \Lambda_{z_i}\right\} | \mu, \Lambda_\mu\right) = \prod_i P\left(z_i, x_i; \Lambda_{z_i} | \mu, \Lambda_\mu\right). \tag{4}$$

The predictive posterior distribution can be used to determine the likelihood that an observed measurement $\tilde{z}$ associates with a given cluster. This is used in the k-means algorithm to assign measurements to clusters.

$$P\left(\tilde{z}|\left\{z_i;\Lambda_{z_i}\right\};\Lambda_{\tilde{z}}\right) = \int P\left(\tilde{z}|\tilde{x};\Lambda_{\tilde{z}}\right) P\left(\tilde{x}|\mu,\Lambda_\mu\right) \times \prod_i P\left(z_i|x_i;\Lambda_{z_i}\right) P\left(x_i|\mu,\Lambda_\mu\right) P\left(\mu,\Lambda_\mu\right) dx_i \; d\tilde{x} \; d\mu \; d\Lambda_\mu. \tag{5}$$

All three integrals in Equations 2, 3, and 5 require solving the integral for a single measurement chain,

$$\begin{aligned} p(z_i|\mu,\Lambda_\mu;\Lambda_{z_i}) &= \int p(z_i,x_i|\mu,\Lambda_\mu;\Lambda_{z_i}) \; dx_i & (6)\\ &= \int p(z_i|x_i,\mu,\Lambda_\mu;\Lambda_{z_i}) p(x_i|\mu,\Lambda_\mu) \; dx_i & (7)\\ &= \int p(z_i|x_i;\Lambda_{z_i}) p(x_i|\mu,\Lambda_\mu) \; dx_i. & (8) \end{aligned}$$

The x_i variables are considered to be nuisance parameters and are removed with the integration. The multivariate Gaussian PDF can be defined as

$$\mathcal{N}(y,\mu,\Lambda) = \frac{|\Lambda|^{1/2}}{(2\pi)^{n/2}} \exp\left(-\frac{1}{2}(y-\mu)^\intercal \Lambda (y-\mu)\right). \tag{9}$$

where n is the number of dimensions. The insertion of Gaussian functions leads to

$$\begin{aligned} p(z_i|\mu,\Lambda_\mu;\Lambda_{z_i}) &= \int \mathcal{N}(z_i,x_i,\Lambda_{z_i}) \mathcal{N}(x_i,\mu,\Lambda_\mu) \; dx_i & (10)\\ &= \frac{(|\Lambda_{z_i}||\Lambda_\mu|)^{1/2}}{(2\pi)^{n/2}} \times & (11)\\ &\quad \int \exp\left(-\frac{1}{2}\left[(z_i-x_i)^\intercal \Lambda_{z_i}(z_i-x_i) + (x_i-\mu)^\intercal \Lambda_\mu (x_i-\mu)\right]\right) \; dx_i. \end{aligned}$$

This integral can be solved analytically (by completing the squares) and the log-likelihood function easily obtained

$$\begin{aligned} \log p(z_i|\mu,\Lambda_\mu;\Lambda_{z_i}) &= C - \frac{1}{2}\log|\Lambda_{z_i}+\Lambda_\mu| + \frac{1}{2}\log|\Lambda_\mu| & (12)\\ &\quad -\frac{1}{2}\left(z_i^\intercal \Lambda_{z_i} z_i + \mu^\intercal \Lambda_\mu \mu - (\Lambda_{z_i} z_i + \Lambda_\mu \mu)^\intercal (\Lambda_{z_i}+\Lambda_\mu)^{-1}(\Lambda_{z_i} z_i + \Lambda_\mu \mu)\right) \end{aligned}$$

with a constant term, $C = \frac{1}{2}\log\left(|\Lambda_{z_i}|\right) - n/2\log\left(2\pi\right)$. For additional discussion on the problem of obtaining analytical solutions, see Appendix 1.

While probabilistic programs can numerically integrate the function in Equation 10 for positive definite precision matrices, it becomes problematic when one of the precision matrices is no

longer positive definite. This would effectively require sample draws from an infinite space. Analytic integration leads to an apparent problematic term with $|\Lambda_{z_i}|$. This term is zero if any of the eigenvalues are zero. However, the term can be cancelled in the equations that are involved in the clustering algorithm described here. The term cancels via the normalization terms, such as the divisor term in the posterior probability equations. The association cost functions can be considered to be log-likelihood ratio tests to determine cluster assignment and the term can again be canceled. Arguments can be made that these measures remain finite in the limit of eigenvalues of Λ_{z_i} approaching zero. This argument assumed that the combination $\Lambda_{z_i} + \Lambda_\mu$ remains positive definite. The selection of the normal-Wishart prior is to ensure that Λ_μ remains positive definite.

Fortuitously, the log-likelihood function in Equation 12 can easily be implemented in Stan. Stan also provides primitives for the Wishart distribution so that a normal-Wishart prior distribution can easily be constructed. The remaining integrals for estimating expectation values for μ and Λ_μ and for the posterior predictive log likelihood ratio test can be easily obtained through numerical integration of the remaining integrals.

1.4 PRIOR DISTRIBUTIONS

The uniform distribution prior was pursued in previous work. The first probabilistic programs that were written this summer also used a uniform prior for the cluster mean and covariance matrix. To ensure that the sampled matrix was positive definite, eigenvalues $\Lambda_1, \Lambda_2 \sim U(0, 100)$ and a rotation angle $\Theta \sim U(0, 2\pi)$ were sampled uniformly and independently. In the case of a 2-D covariance matrix, a sampled matrix was then constructed by

$$\Sigma = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}, \tag{13}$$

where λ_1, λ_2, and θ are all realizations of the random variables Λ_1, Λ_2, and Θ, respectively.

For the Normal-Wishart prior distribution, the following definition is provided. Because of the simple definition, it should be easily definable from the primitives provided by probabilistic programs, assuming the Wishart distribution and MVN parameterized by a precision matrix are in the set of primitives.

Definition

A mean vector and precision matrix pair $(\boldsymbol{\mu}, \boldsymbol{\Lambda}) \sim \text{NW}(\boldsymbol{\mu}_0, \lambda, \mathbf{W}, \nu)$ is equivalent to

1. $\boldsymbol{\Lambda} \sim \mathcal{W}(\mathbf{W}, \nu)$
2. $\boldsymbol{\mu} \sim \mathcal{N}(\boldsymbol{\mu}_0, (\lambda\boldsymbol{\Lambda})^{-1})$

One can interpret the Wishart distribution function as a multidimensional analog to the chi-squared distribution function. For the Wishart distribution, it is required that $\nu > \dim(\mathbf{W}) - 1$, and that the expected value is $\text{E}[\boldsymbol{\Lambda}] = \nu\mathbf{W}$. The larger the value of ν, the more concentrated the distribution is around $\nu\mathbf{W}$. Hence, if one has a reasonable estimate $\boldsymbol{\Lambda}_0$ for $\boldsymbol{\Lambda}$, one can select ν

based on his/her level of confidence on the estimate $\mathbf{\Lambda}_0$. The term $\mathbf{W} = \mathbf{\Lambda_0}/\nu$ can then be set accordingly.

Additionally, it is a good practice to set $\boldsymbol{\mu}_0$ based on the prior belief on where $\boldsymbol{\mu}$ may lie. The parameter λ reflects one's confidence about the parameter $\boldsymbol{\mu}_0$; the larger the value of λ, the larger the values of the precision matrix $\lambda\Lambda$, and so indicates increased confidence in the likely value of $\boldsymbol{\mu}_0$.

2. MARKOV CHAIN MONTE CARLO (MCMC)

The No-U-Turn Sampler (**NUTS**) is an auto-tuned Hamiltonian Monte Carlo (**HMC**)[1] extension developed by Hoffman and Gelman [8]. There are advantages of NUTS over traditional MCMC methods such as the Metropolis algorithm [8]; namely, the Metropolis algorithm performs poorly with high-dimensional data sets due to it purely being a random walk (see curse of dimensionality). As such, whenever possible, the authors usually opted for using the NUTS algorithm.

Under certain regularity conditions, the posterior distributions can exhibit asymptotic normality (with a convergence in distribution). See the Bernstein-von Mises theorem for more details [9]. The paper titled "Applications of the theory of Martingales" by Doob is also applicable to regularity conditions; it is one of the first papers to discuss the Bernstein-von Mises theorem, although the paper contains heavy mathematics [10].

[1] http://arogozhnikov.github.io/2016/12/19/markov_chain_monte_carlo.html

This page intentionally left blank.

3. DESCRIPTIONS OF PROBABILISTIC PROGRAMMING LANGUAGES

Probabilistic programming languages are programming languages that can be used to specify a probabilistic model and then perform inference. Inference can be as simple as attempting to obtain more accurate estimates for some of the model parameters.

3.1 PYMC3

The incorporation of Edward into TensorFlow initially made working with Edward appealing, but ultimately, it was decided to skip the use of Edward because of its relative immaturity and to avoid any potential obstacles during setup that could delay progress.

On that note, PyMC3 was chosen as the first language to apply to the generative model. PyMC3 required the installation of *theano* to be able to work with tensors. PyMC3 was the first language that was implemented for a few reasons:

1. It is easy to import into Python.
2. There are no difficult dependencies to manage during installation.
3. There is excellent user support and documentation.

The first PPL programs that were written assumed uniform priors so prior distribution functions could be ignored. To combat the risk of possibly generating samples with square matrices containing non-positive eigenvalues, the eigenvalues and a rotation angle were independently drawn from uniform distributions of finite extent to ensure that the eigenvalues were positive definite. The covariance matrices were composed via the matrix decomposition outlined in Section 2.

Unfortunately, a roadblock was encountered while trying to advance beyond uniform prior distributions and implement Normal-Wishart prior distributions in PyMC3. As of the summer of 2018, there were no MVN primitives in PyMC3 that used precision matrices. The decision was made to transition to Stan because Normal-Wishart distributions are contained in its set of primitives.

3.2 STAN

First, there are a few helpful references that readers may use to familiarize themselves with Stan in general. The Stan Reference Manual provides some overview of the Stan syntax and example applications [11]. An article by Annis, et al. [12] gives good examples of how to implement user-specified distributions in Stan. Additionally, the "Brief Guide to Stan's Warnings" provides an excellent summary of the meanings of many of the common Stan warnings[2] .

The first difficulty with Stan occurred while trying to set up PyStan to work in Python, likely due to a dependency issue. After a few attempts at reinstallation, this approach was abandoned

[2] http://mc-stan.org/misc/warnings.html

in favor of utilizing RStan with R. The remainder of the probabilistic programs were written in R and RStan. The web page, "Installing RStan on Windows" was a straightforward guide for getting RStan running on MS Windows[3].

Essentially, one of the key aspects of Stan that set it apart from other aforementioned PPLs is that every Stan program defines a model and is saved in a .stan file. This means that models can easily be ported between popular encapsulating languages such as R, Python, and MATLAB. In addition, PPLs such as Edward and PyMC3 were specifically developed to work with Python. Their functions are inherently 'Pythonic' in nature; hence, if one uses those languages, one must also use Python. The installation attempts of this summer suggest that a PyStan installation may not be completely straightforward, whereas the RStan installation was fairly simple; no effort was made to use Stan with MATLAB.

As mentioned earlier, NUTS is one of the state-of-the-art MCMC algorithms, which Stan provides. The Stan manual provides details on how to write programs for log probability densities. This capability is essential for writing modern MCMC algorithms [11]. The Stan files written for this project defined models specifying the prior probability distributions and the likelihood functions. As long as the model remained the same, the same model could be repeatedly executed to analyze different data sets. When a Stan program is first called in R, the model is compiled into C++ code. Assuming the model remains the same, the model can be saved as a .Rda file and then loaded into future instantiations of R to avoid duplicative compilations (which can take up to a few minutes to complete).

When diagnosing the convergence of MCMC, it is useful to run multiple chains at the same time - the idea being that if there is decent convergence, multiple chains starting at different points in state space should arrive at roughly the same region. The Gelman-Rubin diagnostic compares the within-chain variance to the between-chain variance; if they are similar, then it can be said that the chains appear to be well-mixed; values close to 1 are the most ideal [13].

To see how well a model is performing, it is useful to plot the autocorrelation as a function of the lag, and examine the traceplots. A good autocorrelation function should have a value of 1 at a lag of 0 and values close to 0 everywhere else. A good traceplot should appear fuzzy, and there should be little evidence of correlation. Thinning (keeping every kth sample) can be used to reduce the autocorrelation of the output sample sequence.

[3] https://github.com/stan-dev/rstan/wiki/Installing-RStan-on-Windows

4. STABILITY MEASUREMENTS

The stability of the estimates to changes in sample size or measurement error level was examined. One would expect that a decrease in sample size or an increase in measurement noise would result in loss of accuracy. The experiments that were run indicate that lower sample sizes or higher measurement errors result in posterior estimates that are more biased towards the prior distribution parameter values, as expected.

Experiments were also conducted to evaluate the sensitivity to changes in sample size. For each Rda file, 50 MCMC runs were performed. After each run, the mean of the posterior distribution samples was calculated to get sample means for the mean vector and the precision matrix. Hence, there are a total of 50 mean vector estimates and 50 precision matrix estimates for data sets with different sample sizes.

Similarly, experiments were conducted to evaluate the sensitivity to changes in measurement error levels. Like before, for each Rda file, 50 MCMC runs were performed, and after each run, the mean of the posterior distribution samples was calculated to get sample means for the mean vector and the precision matrix. As before, there are a total of 50 mean vector estimates and 50 precision matrix estimates for data sets with different measurement error levels.

4.1 STABILITY TO SAMPLE SIZE

Figure 2 shows the resulting estimates of the mean components from 50 MCMC runs for each of seven sample sizes ranging from 2 to 1000. The simulated data were generated to describe a cluster with a mean of $[1, 2]^\intercal$ and a covariance matrix of $[5,\ -1; -1,\ 2]$. The eigenvalues for this matrix are approximately 5.30 and 1.70. The measurement noise model's precision matrix was set to $\Lambda_{z_i} = [1, 0; 0, 1]$. There were no missing measurements in this analysis. The prior parameters were set to $\mu_0 = [0, 0]$, $\lambda = 2$, $W = [1, 0; 0, 1]$ and $\nu = 3$. Figure 3 show histograms of the 50 estimates of the eigenvalues. As could be expected, the estimates improve as sample size increases. For low sample sizes, the estimates are very poor and biased toward the prior distribution.

4.2 STABILITY TO MEASUREMENT NOISE

A similar test was run to assess the sensitivity to different measurement noise levels. The same basic measurement noise precision matrix was used but scaled across a range of factors: $\Lambda_{z_i} = a\,[1, 0; 0, 1]$, where a ranged from 0.05 to 20. The sample size for this test was set at 100 samples. The results are shown in Figures 4 and 5. Again, as expected, as the precision improves, the more reliable estimates are obtained for the mean and eigenvalues for the covariance matrix. Although not as extreme for this series of runs as compared to the runs with varying sample sizes, the estimates are more biased toward the prior parameters for the runs with larger noise levels (smaller a).

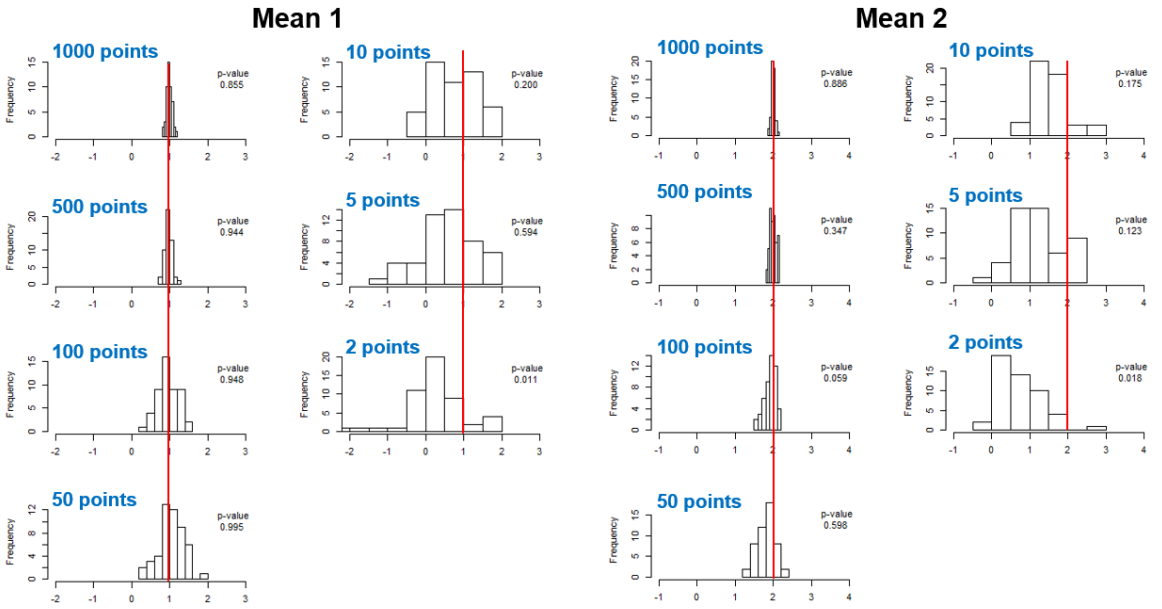

Figure 2. Results from estimating the 2-D mean from different numbers of data samples.

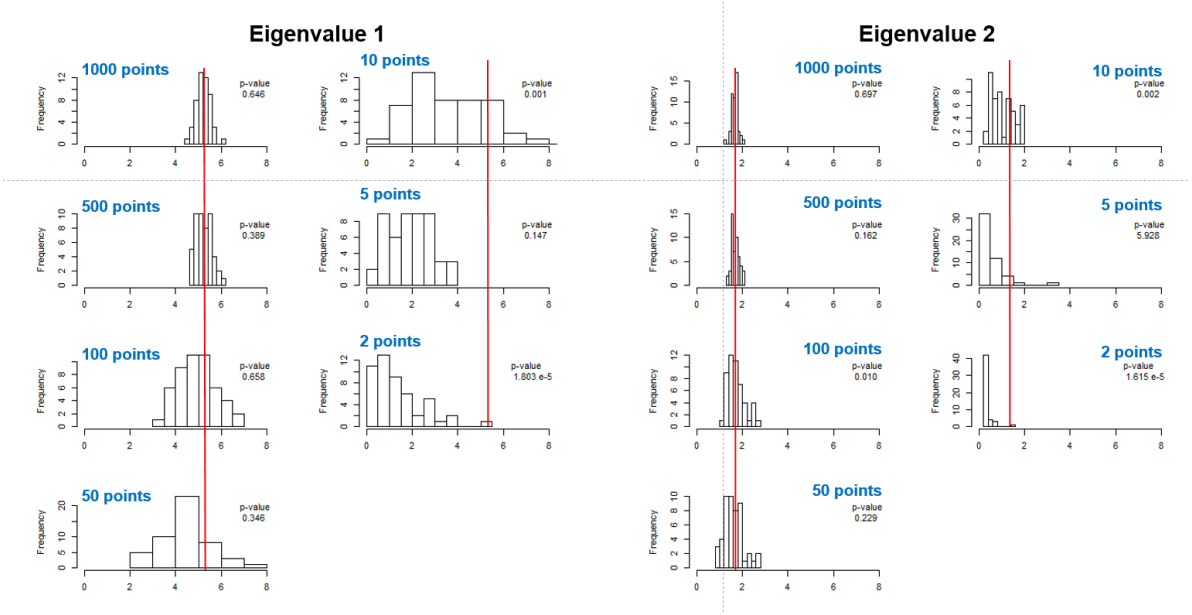

Figure 3. Results from estimating the eigenvalues from different numbers of data samples.

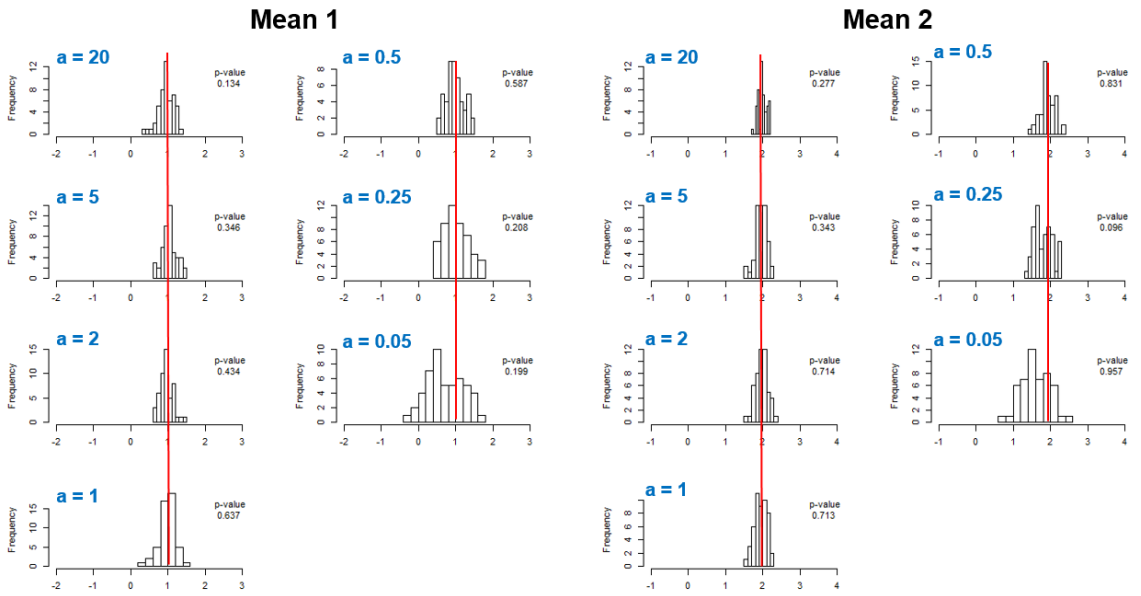

Figure 4. Results from estimating the 2-D mean from different measurement noise levels.

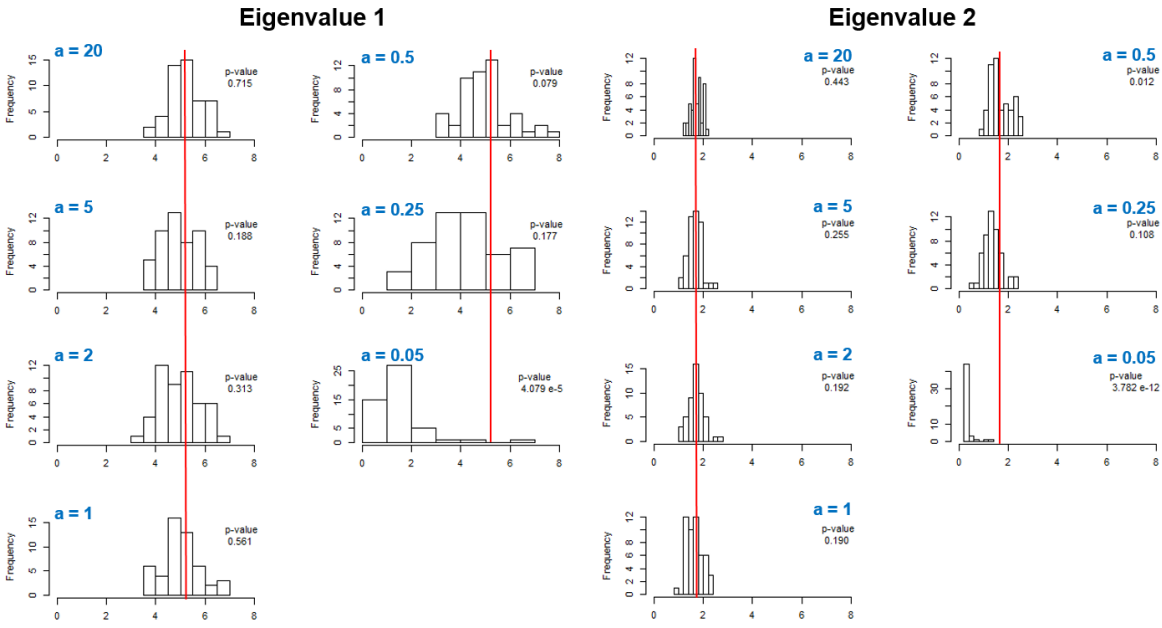

Figure 5. Results from estimating the eigenvalues from different measurement noise levels.

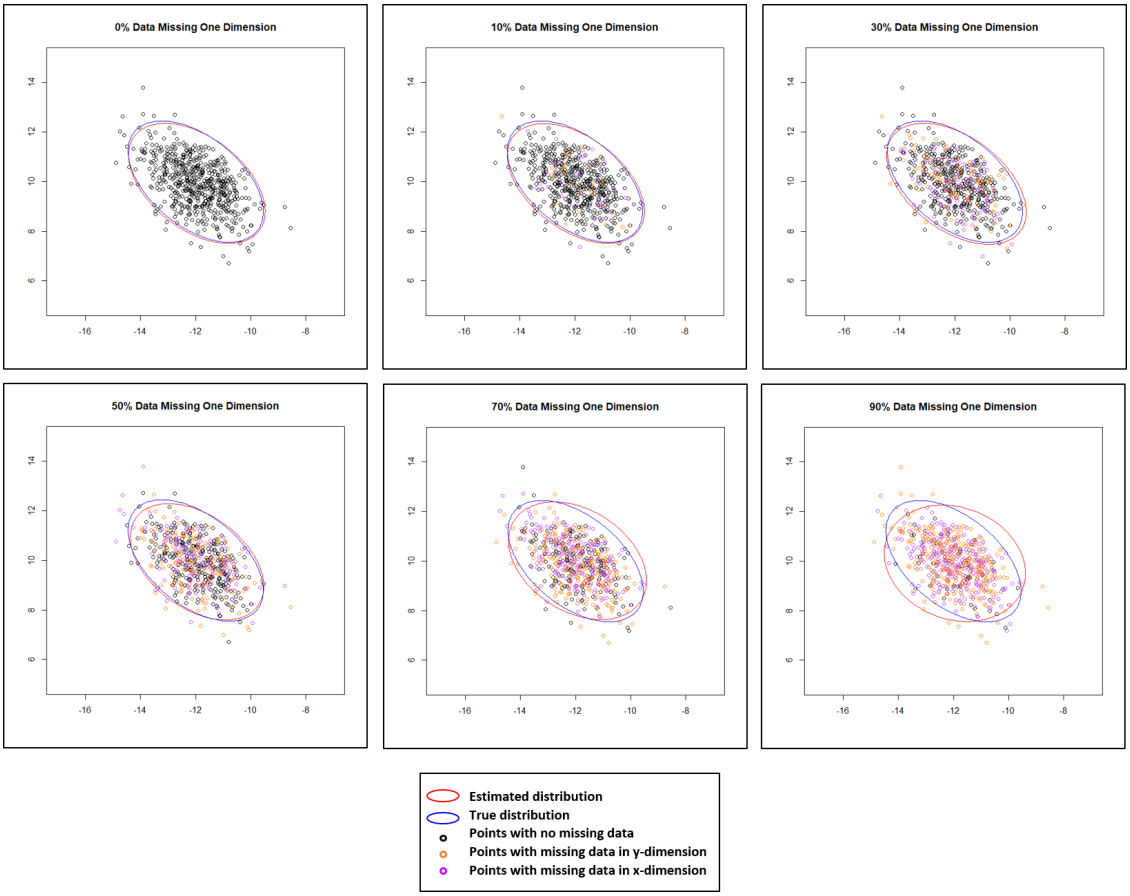

Figure 6. Results from estimating the true distribution while varying the number of points missing data in one dimension.

4.3 STABILITY TO MISSING DATA FRACTIONS

A test was run to assess the quality of mean and covariance estimates as the amount of data with missing values increases. There were 600 points sampled from a distribution with mean $[-12, 10]^\intercal$ and a covariance matrix of $[1, \ -0.5; -0.5, \ 1]$. Measurement noise was set so low as to be negligible, in order to simplify the study of the effects of the variable in question: the percentage of missing data.

Figure 6 shows that as the percentage of points with missing data increases, the quality of the distribution estimate decreases. While the estimate of the mean remains reasonably accurate, the estimate of covariance matrix degrades and becomes more "circular". Figure 7 shows 1) the eigenvalues of the estimated covariance matrix, 2) the rotation angles of the matrix, and 3) the estimated mean, all as a function of percentage of points with missing data. Although the estimate of the mean remains reasonably accurate as the percentage of points with missing data increases, the eigenvalues and rotation angles show that the covariance estimate degrades.

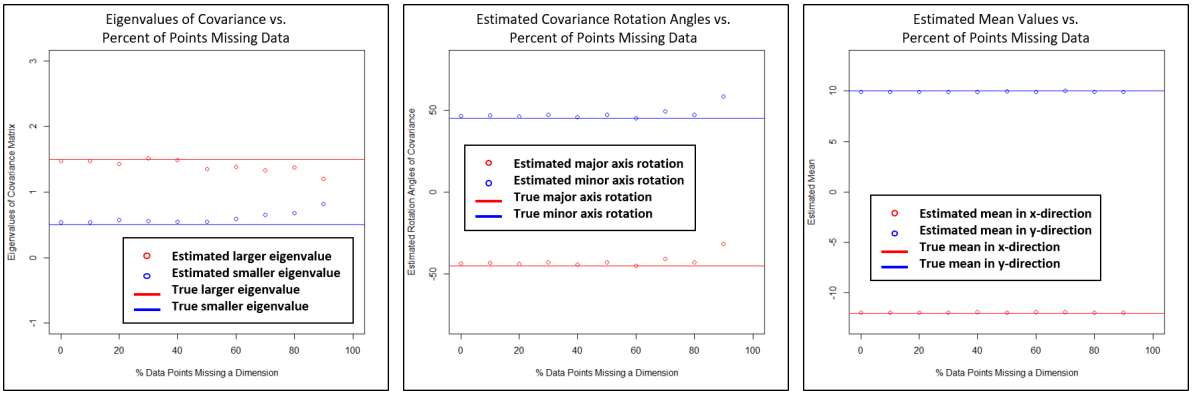

Figure 7. Results from estimating the eigenvalues, rotation angles of the covariance, and mean, while varying the number of points missing data in one dimension.

5. K-MEANS CLUSTERING ALGORITHM

5.1 INITIALIZATION

The k-means algorithm is certainly simple and fast, but it is widely known that finding the global optimal solution for clustering even low-dimensional 2-D data into k hard clusters is an NP-hard problem [14]. To get faster convergence to an optimal solution, the k-means++ algorithm was used to initialize the centers of each cluster. This initialization approach tends to select the k cluster means with large separation. The algorithm randomly selects a cluster center from the measurement data set and then continues to randomly choose additional cluster centers from the remaining measurements with a probability proportional to D^2, where D is the distance from a measurement to the nearest cluster's center.

For the simulations, the generation of the initial definition of precision or covariance matrices for the cluster distributions was kept relatively simple. Variances for all d dimensions were independently estimated from the complete data set while covariance terms were defined to be zero. The cluster precision matrix diagonal terms were then simply the multiplicative inverse of these variances while the initial off-diagonal terms were defined to be zero. The same initial precision matrix was used for all the clusters.

After selection and definition of the initial estimate of cluster means and precision matrices, the k-means algorithm alternated between the following two steps: assigning data points to the cluster with the highest probability of association and updating cluster distribution parameters. The algorithm would repeat the two steps until a convergence criterion was met (typically, when all measurements stop being reassigned) or the maximum number of iterations was reached. In this algorithm, the maximum number of iterations was set to 20 to ensure a timely completion of the algorithm.

5.2 ASSIGNMENT OF MEASUREMENTS TO CLUSTERS

The purpose of this step is to assign (or reassign) each data point to the appropriate cluster. Traditional k-means or fuzzy c-means algorithms utilize a distance metric (typically Euclidean), but this algorithm was written to utilize log-likelihood ratios. As every cluster distribution is characterized by a mean and precision matrix, a measurement is assigned to a cluster if and only if the probability density value for association between that measurement and that cluster is a maximum of all cluster association costs. Specifically, the hard assignment a_i for a data point z_i is updated upon each iteration with

$$a_i = \underset{j}{\operatorname{argmax}} \; p(z_i|\mu_j, \Lambda_{\mu_j}; \Lambda_{z_i}), \tag{14}$$

where j is the cluster index.

Equation 12 was used to generate the association log likelihood between each measurement and cluster. Note that Stan does not require the log probability to describe a normalized probability density function – being off by an additive constant term is fine. Stan drew samples to represent the posterior distribution, as specified by the model. Usually, more than one MCMC chain is recommended to determine convergence diagnostics. However, for the simulations associated with clustering, only one chain was drawn, due to time constraints. Experience gained from running these simulations suggests that MCMC should be run before clustering. This way, multiple chains can be run to verify convergence was achieved, as indicated via the Gelman-Rubin statistic [13]. Additionally, this can indicate an appropriate choice for the number of warm-up samples (burn-in period) and chain length. For this analysis, 1000 samples were typically drawn, with half of those being discarded as part of the burn-in period.

5.3 CLUSTER DISTRIBUTION PARAMETER UPDATES

To cluster the simple synthetic data, the NUTS algorithm was used to draw 1000 samples with no thinning. The generated sequence appeared to be roughly uncorrelated and had only one mode. The estimates $\hat{\mu}$ and $\hat{\Lambda}_\mu$ were calculated by taking the average of the valid posterior samples; essentially, this is the same as taking the expected value of the posterior distribution. These new estimates replaced the old ones in the k-means algorithm.

In the Stan programs that were written for this project, the variable value, NA, was used to denote "Not Available", or missing features, in a given measurement's state vector, z_i. The R programming language naturally uses NA to represent missing data and handles mathematical operations with this special assignment. Stan does not naturally support missing data fields. By default, in Stan, NA represents 'not a number' and the multiplication $\text{NA} \cdot 0$ returns NA, so to achieve the correct calculations, the programs simply converted all NAs to 0s before performing multiplications. This is a natural convention to adopt with the use of precision matrices to indicate missing data. In the principal coordinate frame for measurements, any real value of z_i multiplied by the zero row and column in the missing coordinate axis will return zero, as desired. The correct computation is naturally and correctly performed for products like $z_i^\intercal \Lambda_{z_i} z_i$. Some care is necessary for representing measurement from different principal axes. The zero terms need to be set for

precision matrices defined in the principal coordinate frames and then rotated into the common frame.

5.4 CLUSTERING ALGORITHM SUMMARY

The aforementioned steps for the k-means clustering algorithm are summarized in the pseudocode contained in Algorithm 1. Let d be the dimensionality of the measurements, z_i.

Algorithm 1 k-Means for Missing Data

procedure k-MEANS FOR MISSING DATA
  Initialize k means: $\mu_1, \mu_2, \mu_3, \ldots, \mu_k \in \mathbb{R}^d$ using the kmeans++ algorithm
  Initialize k $d \times d$ cluster precision matrices: $\Lambda_{\mu_1}, \Lambda_{\mu_2}, \Lambda_{\mu_3}, \ldots, \Lambda_{\mu_k}$
  Initialize the Normal-Wishart prior distribution on the cluster means and precision matrices
  repeat
    for $i \in \{1, 2, 3, \ldots, N\}$ **do**
      Update cluster assignment $a_i = \underset{j}{\operatorname{argmax}}\ p(z_i|\mu_j, \Lambda_{\mu_j}; \Lambda_{z_i})$
    end for
    for $j \in \{1, 2, 3, \ldots, k\}$ **do**
      Draw samples from the posterior distribution using the NUTS algorithm
      Estimate the cluster parameters μ_j and Λ_{μ_j} with sample means of the posterior
    end for
  until convergence
end procedure

In many clustering algorithms, the performance can be measured using an objective loss function. One was not explicitly used here, but performance can be quantified with a sum of the log-likelihood function for all the measurements and their assigned clusters, as with Equation 12. The constant term would be neglected.

5.5 R AND STAN CODE

The model was specified and described in Stan while the code to perform the iterative steps in the k-means algorithm was written in R. In Stan, the model was defined by the prior distribution and likelihood functions. R was used to generate simulated raw measurement data that was saved in an .Rda file for repeated executions of the algorithm. The raw measurement data were transferred from R to Stan. Stan was then used to generate and return posterior distribution samples to R, which the R code then used to determine the assignments of measurements to clusters or to calculate sample means, depending on the type of requested posterior distribution.

This page intentionally left blank.

6. EMPIRICAL CLUSTERING RESULTS

6.1 CLUSTERING WITH TWO-DIMENSIONAL GAUSSIAN DATA

The first simulation considered the case of two-dimensional, reasonably separated Gaussian clusters. Clean data were generated by initializing three Gaussian clusters via the distributions

$$\mathcal{N}\left(\boldsymbol{\mu_1} = \begin{bmatrix} -7 \\ 16 \end{bmatrix}, \boldsymbol{\Sigma_1} = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}\right), \tag{15}$$

$$\mathcal{N}\left(\boldsymbol{\mu_2} = \begin{bmatrix} -12 \\ 10 \end{bmatrix}, \boldsymbol{\Sigma_2} = \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}\right), \tag{16}$$

$$\mathcal{N}\left(\boldsymbol{\mu_3} = \begin{bmatrix} -5 \\ 8 \end{bmatrix}, \boldsymbol{\Sigma_3} = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}\right). \tag{17}$$

Two hundred simulated measurements were sampled from each distribution to create the cluster measurements. Then, six hundred 2×2 precision matrices were drawn from a Wishart distribution,

$$\mathcal{W}\left(\begin{bmatrix} 0.8 & 0 \\ 0 & 0.8 \end{bmatrix}, 10\right). \tag{18}$$

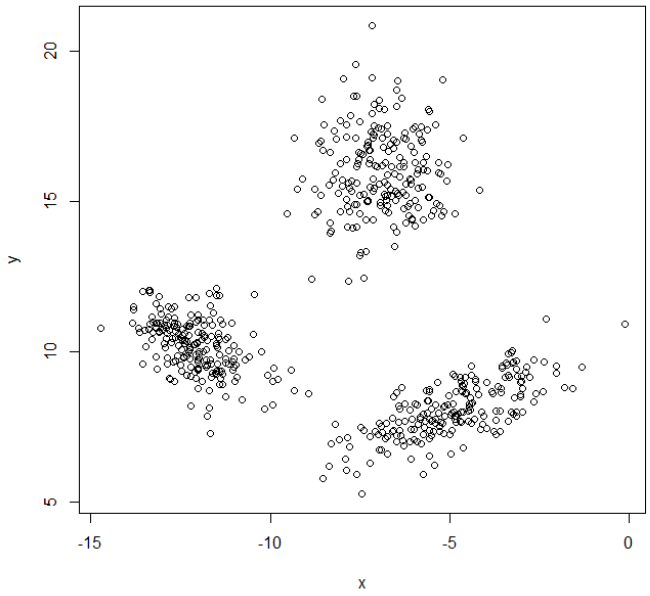

Figure 8. A plot of the simulated data for three clusters before the addition of noise.

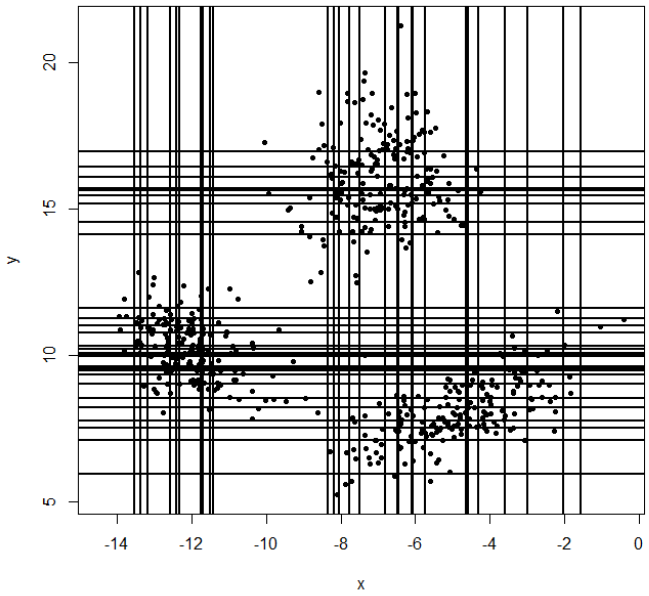

Figure 9. A plot of the simulated data for three clusters with the addition of noise and measurements with missing dimensions plotted as rectangles spanning the plot.

Based on the parameters, the noise was defined, on average, with the precision matrix $\begin{bmatrix} 8 & 0 \\ 0 & 8 \end{bmatrix}$. To simulate missing features, 60 measurements were selected at random: 30 measurements had the values in the first row and column replaced with 0 while the other 30 measurements had the values in the second row and column replaced with 0. This marked these dimensions as being associated with missing data. Each of these 600 precision matrices represents the measurement error distribution. Figure 8 shows the true x-y locations for the 600 data points. Figure 9 shows the measurements with added noise. Measurements with missing values for one of the two dimensions are plotted as rectangles that span the axis of the plot with the missing dimension.

The following series of figures show the progress of the clustering algorithm on the noisy data. These figures do not plot the missing data as horizontal and vertical rectangles to avoid obscuring the cluster assignments. The initial assignment of measurements to clusters and cluster covariance ellipses are shown in Figure 10.

The clustering algorithm iterated until there were no longer any reassignment of measurements to different clusters. The ellipses represent the 95% elliptical confidence regions for the three cluster distributions. Figures 11 through 17 show the cluster distributions marching to a good assignment after seven iterations. The colors (red, green, black) indicate the assignments that the algorithm made.

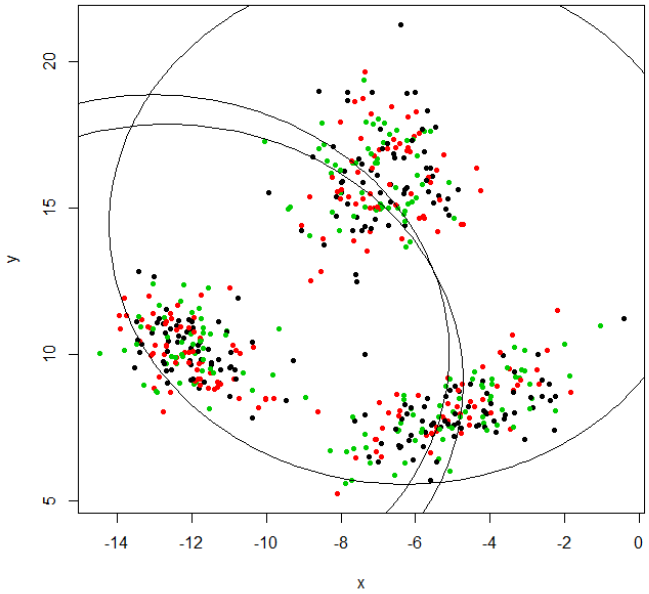

Figure 10. The initial clusters and assignments for a simulated data set.

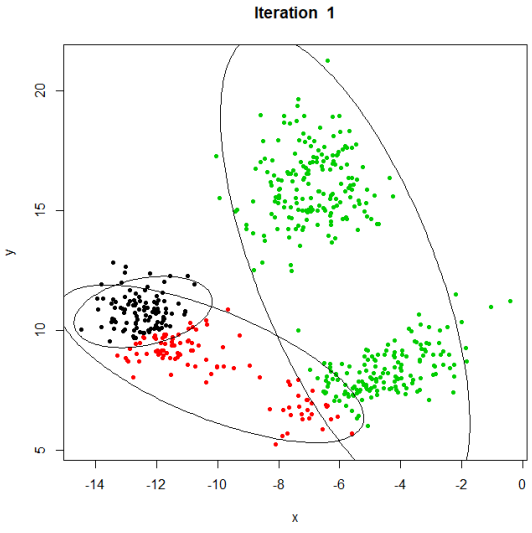

Figure 11. Cluster results after iteration one.

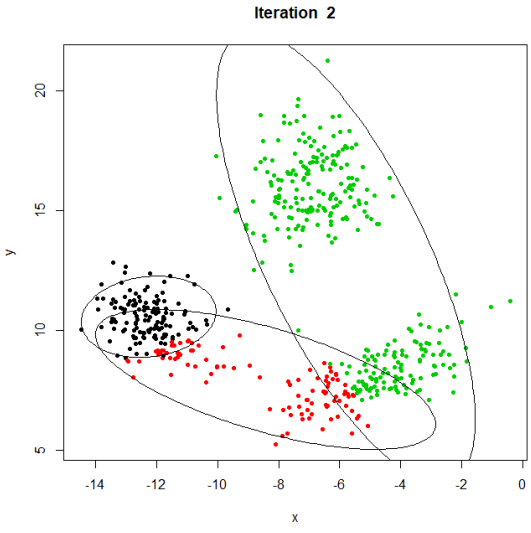

Figure 12. Cluster results after iteration two.

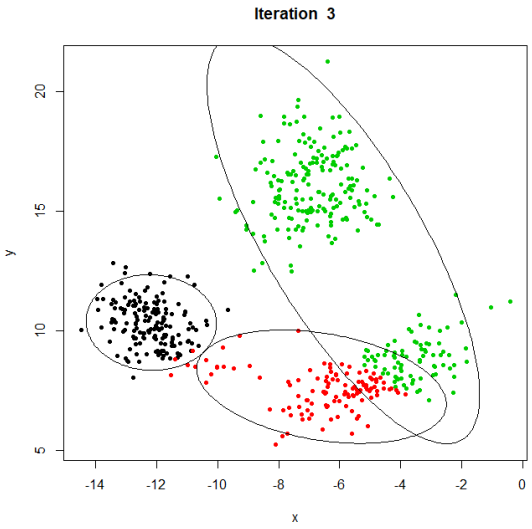

Figure 13. Cluster results after iteration three.

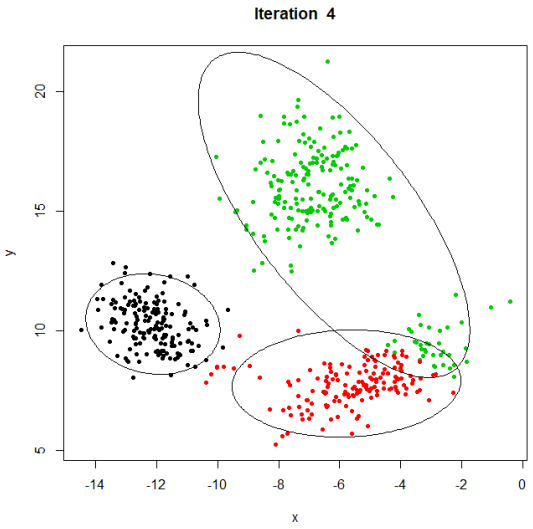

Figure 14. Cluster results after iteration four.

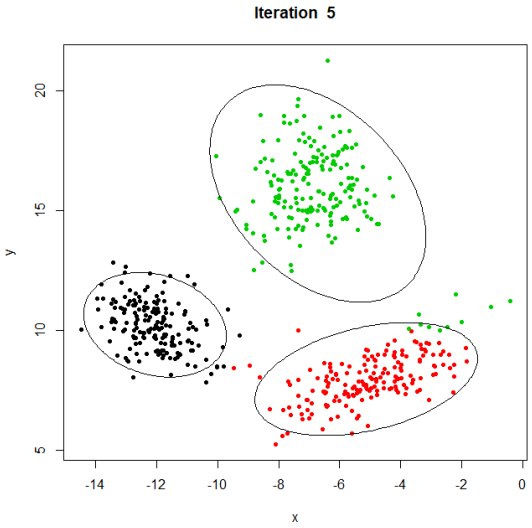

Figure 15. Cluster results after iteration five.

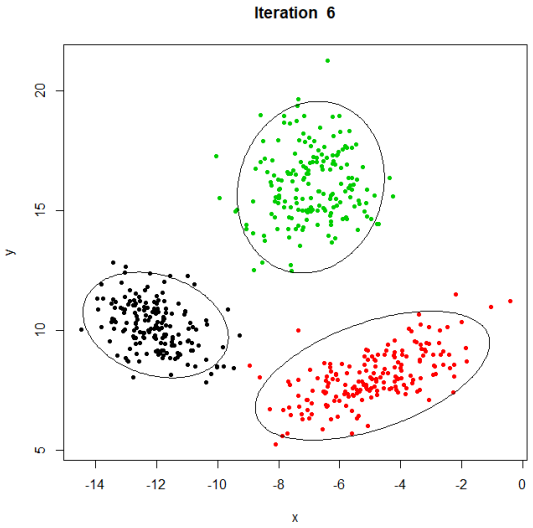

Figure 16. Cluster results after iteration six.

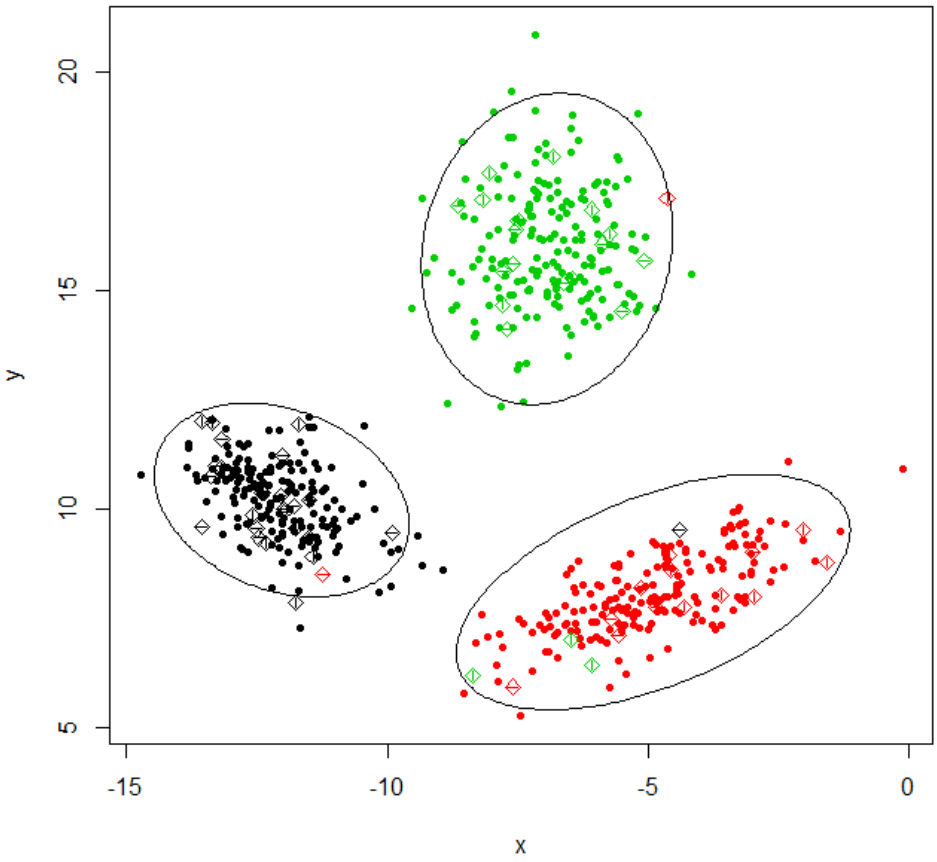

Figure 17. The final results from the clustering algorithm after iteration seven. Measurements are colored by their assignment. Measurements with missing data are plotted as diamonds with a vertical or horizontal bar, indicating the direction with missing data.

Figure 9 represents what the algorithm initially received, while Figure 17 is a plot of what the clustering algorithm returned. The missing data are depicted using diamonds with superimposed horizontal or vertical lines to indicate the dimension with no information. A horizontal line indicates no information about the first dimension while a vertical line indicates no information about the second dimension. Data with both dimensions intact is represented with a colored, filled-in circle with the color reflecting the cluster assignment for a measurement.

The clustering results are reasonable. First, the algorithm captured reasonable estimates of the three underlying distributions' parameters. The estimated means and covariances, which can be compared to the true values above, are as follows:

$$\hat{\boldsymbol{\mu}}_{\mathbf{1}} = \begin{bmatrix} -6.941 \\ 15.964 \end{bmatrix}, \widehat{\boldsymbol{\Sigma}_{\mathbf{1}}} = \begin{bmatrix} 0.967 & 0.156 \\ 0.156 & 2.119 \end{bmatrix}, \tag{19}$$

$$\hat{\boldsymbol{\mu}}_{\mathbf{2}} = \begin{bmatrix} -12.024 \\ 10.212 \end{bmatrix}, \widehat{\boldsymbol{\Sigma}_{\mathbf{2}}} = \begin{bmatrix} 0.996 & -0.236 \\ -0.236 & 0.823 \end{bmatrix}, \tag{20}$$

$$\hat{\boldsymbol{\mu}}_{\mathbf{3}} = \begin{bmatrix} -4.904 \\ 8.106 \end{bmatrix}, \widehat{\boldsymbol{\Sigma}_{\mathbf{3}}} = \begin{bmatrix} 2.377 & 0.886 \\ 0.886 & 1.214 \end{bmatrix}. \tag{21}$$

The impact of missing features on the assignments can be seen in a number of cases. For instance, all of the data with a first component between -14 and -10, but with a missing second component, were assigned to the black cluster. This makes perfect sense because the other clusters have extremely low association probabilities in this interval. At the same time, some errors can be seen for the assignment of other measurements with missing features in other intervals because of cluster overlap in one dimension or the other. This is because the clustering algorithm performs a hard assignment based upon maximum log-likelihood. For example, one of the green diamonds towards the bottom of Figure 17 is misclassified as a member of the green cluster while in fact the data point was generated from the red cluster. This is because the algorithm only knows that the point lies somewhere close to the line $x = -8$. Membership is more likely to be with the green cluster given the likelihood values for the three clusters along this line. Because the clustering algorithm makes a hard decision, it follows that the algorithm incorrectly assigned this measurement to the green cluster.

With 60 missing data points, the algorithm achieved a clustering performance of $594/600 = 99\%$ probability of correct assignment. The missed classifications for this simulated data set are strictly due to missing features in the measurements. Data sets with higher overlap between clusters are likely to suffer from misclassifications because of the inability to fully separate the clusters.

6.1.1 Comparison with Other Techniques

This section compares the performance of the algorithm to that of several well-established techniques: (1) single imputation with mean substitution, (2) multiple imputation by chained equations (using the MICE package), and (3) nonparametric imputation (using the missForest package, which builds a random forest model for each variable and then uses these models to predict missing values). For (1) and (3), k-means++ is run after the dataset is "completed" by

imputing all missing values. For MICE, $m = 5$ completed datasets are formed, k-means++ is run on each completed dataset individually, and then the final cluster assignments are determined through voting. Table 1 summarizes the accuracy of each approach.

TABLE 1

Comparison of Missing Value Imputation Approaches

Approach	**Clustering Accuracy**
Mean substitution	$575/600 = 95.8\%$
MICE	$577/600 = 96.2\%$
missForest	$581/600 = 96.8\%$
Probabilistic programming	$594/600 = 99.0\%$

To bracket these results, recall that $540/600 = 90\%$ of the data points have no missing values. So a strategy of simply dropping all rows with missing data and clustering the rest can achieve at most 90% clustering accuracy. As another point of comparison, performing k-means++ on the noise-free data results in an accuracy of $599/600 = 99.8\%$.

The probabilistic programming approach outperforms the three techniques for the clustering simulation described above. Of course, a more extensive evaluation is needed to be able to declare that the probabilistic programming technique developed here is truly superior to other methods. The evaluation would need to be statistically significant, evaluate performance over different simulated and real data sets, and additional missing-data techniques. However, the results of this initial comparison suggest that the probabilistic programming algorithm may have superior performance to many other techniques. A more in-depth analysis will be pursued in the future.

Implementation Note: The randomForest package (required by missForest) appears to have a limitation in which there must be at least two independent variables in a model. This is a problem for the two-dimensional clustering setup, where missForest will create models to predict x given y and predict y given x. As a workaround, an R data frame was created with x replicated as $x1$ and $x2$, allowing missForest to run without errors. Using completed $x1$ or $x2$ values for x resulted in the same performance.

6.2 CLUSTERING WITH THREE-DIMENSIONAL GAUSSIAN DATA

This simulation was designed to generate measurements for four different 3-D Gaussian distributions. For visualization, the resulting measurements were rounded to the nearest integer in a range of [0,255]. The measurements can then be plotted as RGB colors for visualization. Each cluster has 100 samples, for a total count of 400 measurements. Like before, the measurement noise was generated randomly for each data point. In Figure 19, measurements that are missing one feature are depicted as a black square. The figure shows three measurements that were selected

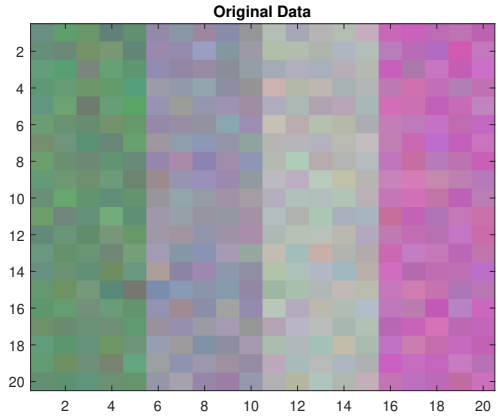

Figure 18. An RGB color representation of the simulated 3D data set, arranged in a grid pattern.

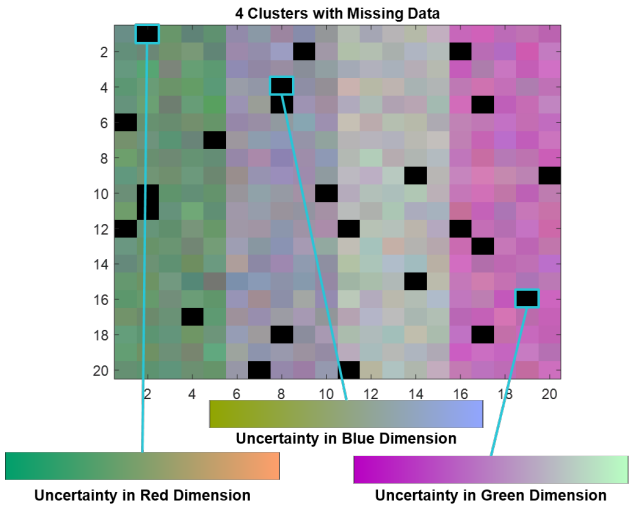

Figure 19. A plot of simulated 3D data with missing features.

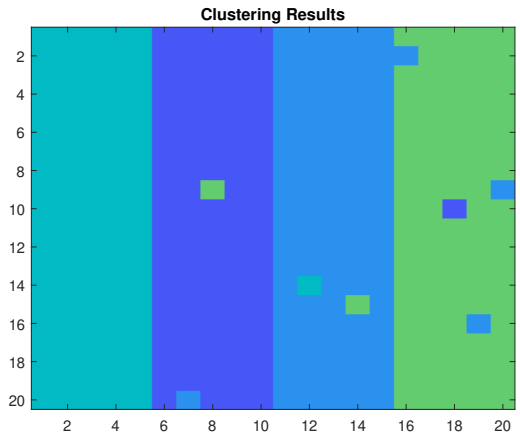

Figure 20. The resulting cluster assignments for the simulated 3D data set with missing features.

to illustrate missing dimensions. The displayed color gradients that associate with black squares indicate a range of possible colors for the squares, given the constraints from the measured color values. The color gradients are based upon the means of the two known RGB values with the missing value spanning a range from 0 to 255.

The cluster distributions for the 3D data were all spherical distributions with the covariance matrices defined as $\begin{bmatrix} 80 & 0 \\ 0 & 80 \end{bmatrix}$. The measurement noise precision matrices were all sampled from a Wishart distribution, $\mathcal{W}\left(\begin{bmatrix} 1/5 & 0 \\ 0 & 1/5 \end{bmatrix}, 15\right)$.

Figure 20 shows the results of the clustering. The mean cluster color is shown for the measurements. There were 8 misclassifications, giving an overall accuracy of $392/400 = 98\%$. It can be noted that not all of the misclassifications are due to the 24 measurements with missing features. Two are misclassifications of measurements with no missing information. Likely, these misclassifications were due to the overlap between clusters that was enhanced by measurement uncertainty.

This page intentionally left blank.

7. SUMMARY AND FUTURE WORK

The primary goal of this project was to determine if a probabilistic programming language could be used to process multivariate data with missing features. On that front, the primary task was to develop programs that would estimate Gaussian cluster parameters and thereby, also cluster data sets with missing fields. By adopting precision matrices to represent measurement noise, it was not only possible to account for sensors with different levels of precision, but also account for different coordinate systems and for missing features.

A possible perceived weakness of the algorithm described here is that the parameter estimates become biased towards the prior parameters with decreasing sample sizes or increasing measurement noise. In actuality, this is the intended behavior for Bayesian algorithms; estimates are influenced by the selection of prior probability distributions and only weakly influence the resulting estimates when sample sizes are large and measurement noise low. The advice is then that the parameters for prior distribution should be chosen carefully so that results are a reflection of the available information.

There are a number of possible avenues for future work. This technical report only presents an initial analysis of the effectiveness of probabilistic programs for processing incomplete data. Further analyses can be performed to identify subtleties with the method.

The clustering algorithm developed here makes hard assignments of measurements to clusters within its iterative loop. Clustering algorithms that make soft assignments could be developed and studied. This class of algorithms would provide more information on possible cluster assignments for measurements with missing information so that an analyst could recognize conditions where cluster assignments might be ambiguous. The effects of missing features on a soft clustering algorithm remain to be studied. It has been suggested by colleagues that the maximization step in the data assignment step can be replaced with another MCMC algorithm for soft clustering. During this MCMC run, the logistic normal distribution may be applicable for estimating partial memberships.

A serious issue with the algorithms that were studied here is that the MCMC clustering algorithms take increasingly significant time to run as the number of dimensions increase. Major research efforts are required to significantly improve the computational performance of MCMC algorithms, even beyond the applications presented in this report.

Estimating the appropriate value of the number of clusters k remains a big question in clustering research. A review of the clustering literature and implementation of algorithms that attempt to estimate the number of clusters could be performed. There are algorithms that run a set of k-means algorithms with different values for k and then compare a global cost measure to determine the optimal fit. Another possible study is to determine if the same precision matrix technique can be ported to nonparametric Bayesian methods, where k dynamically fluctuates as the algorithm runs, with eventual convergence to an optimal value for k.

As a final suggestion in an incomplete list of possible future studies, additional studies could be conducted to investigate how other multivariate distributions, such as the multivariate gamma distribution, might support a representation for missing data. Currently, this study has only

considered the representation of missing features for multivariate Gaussian distributions. Other data sets might be better described by other classes of multivariate probability distributions.

A INTEGRALS REQUIRED FOR ESTIMATES OF MEANS, COVARIANCE MATRICES, AND POSTERIOR PROBABILITIES

Although the integrals involved in the estimation of posterior parameters contain well behaved distribution functions such as Gaussian and Wishart distributions, integration of the combinations of functions has been difficult to perform via analytical methods. Expanding on Section 1.3, the typical integral for expectation values and posterior probabilities is of the form

$$\langle \mathcal{F} \rangle = \int \mathcal{F} P\left(\mu, \Lambda_\mu | \left\{z_i, x_i; \Lambda_{z_i}\right\}\right) d\mu \; d\Lambda_\mu, \tag{A.22}$$

where $\mathcal{F}$ is a function or variable. The probability $P\left(\mu, \Lambda_\mu | \left\{z_i, x_i; \Lambda_{z_i}\right\}\right)$ is expanded in Equation 1 with the curly braces indicating the product of probability functions, as with Equation 4.

The full integral expansion is then of the form of

$$\langle \mathcal{F} \rangle = \frac{\int \mathcal{F} \prod_i \mathcal{N}\left(z_i, x_i, \Lambda_{z_i}\right) \mathcal{N}\left(x_i, \mu, \Lambda_\mu\right) \mathcal{N}\left(\mu, \mu_0, \lambda\Lambda_\mu\right) \mathcal{W}\left(\mathbf{W}, \nu\right) \; dx_i \; d\mu \; d\Lambda_\mu}{\int \prod_i \mathcal{N}\left(z_i, x_i, \Lambda_{z_i}\right) \mathcal{N}\left(x_i, \mu, \Lambda_\mu\right) \mathcal{N}\left(\mu, \mu_0, \lambda\Lambda_\mu\right) \mathcal{W}\left(\mathbf{W}, \nu\right) \; dx_i \; d\mu \; d\Lambda_\mu}. \tag{A.23}$$

The Gaussian distribution is defined for precision matrices, as with Equation 9. The Wishart distribution is defined as

$$\mathcal{W}\left(\mathbf{W}, \nu\right) = \frac{1}{2^{\nu p/2} \left|\mathbf{W}\right|^{\nu/2} \Gamma_p\left(\frac{\nu}{2}\right)} \left|\Lambda_\mu\right|^{(\nu-p-1)/2} e^{-\mathrm{tr}\left(\mathbf{W}^{-1}\Lambda_\mu\right)/2} \tag{A.24}$$

where

$$\Gamma_p\left(\frac{\nu}{2}\right) = \pi^{p(p-1)/4} \prod_{j=1}^{p} \Gamma\left(\frac{n}{2} - \frac{j-1}{2}\right) \tag{A.25}$$

The variable p is the dimension of the precision matrices.

The integrals over the product of probabilities involving z_i and x_i is straightforward and are given in Equations 10 and 11. The solutions for these integrals when $\mathcal{F}$ is independent of z_i and x_i is

$$\begin{aligned} p(z_i|\mu, \Lambda_\mu; \Lambda_{z_i}) &= \left|\Lambda_{z_i}\right|^{1/2} \left|\Lambda_\mu\right|^{1/2} \left|2\pi\left(\Lambda_{z_i} + \Lambda_\mu\right)\right|^{-1/2} \\ &\quad e^{-\frac{1}{2}\left(z_i^\intercal \Lambda_{z_i} z_i + \mu^\intercal \Lambda_\mu \mu - (\Lambda_{z_i} z_i + \Lambda_\mu \mu)^\intercal (\Lambda_{z_i} + \Lambda_\mu)^{-1} (\Lambda_{z_i} z_i + \Lambda_\mu \mu)\right)} \end{aligned} \tag{A.26}$$

The full integral for the numerator of Equation A.23 is now

$$\begin{aligned} \langle \mathcal{F}_\mathcal{N} \rangle &= \int \mathcal{F} \prod_i \left|\Lambda_{z_i} + \Lambda_\mu\right|^{-1/2} e^{-\frac{1}{2}\left(\mu^\intercal \Lambda_\mu \mu - (\Lambda_{z_i} z_i + \Lambda_\mu \mu)^\intercal (\Lambda_{z_i} + \Lambda_\mu)^{-1} (\Lambda_{z_i} z_i + \Lambda_\mu \mu)\right)} \times \\ &\quad \mathcal{N}\left(\mu, \mu_0, \lambda\Lambda_\mu\right) \mathcal{W}\left(\mathbf{W}, \nu\right) \left|\Lambda_\mu\right|^{1/2} \; d\mu \; d\Lambda_\mu \end{aligned} \tag{A.27}$$

and the denominator is

$$\begin{aligned} \langle \mathcal{F}_\mathcal{D} \rangle &= \int \prod_i \left|\Lambda_{z_i} + \Lambda_\mu\right|^{-1/2} e^{-\frac{1}{2}\left(\mu^\intercal \Lambda_\mu \mu - (\Lambda_{z_i} z_i + \Lambda_\mu \mu)^\intercal (\Lambda_{z_i} + \Lambda_\mu)^{-1} (\Lambda_{z_i} z_i + \Lambda_\mu \mu)\right)} \times \\ &\quad \mathcal{N}\left(\mu, \mu_0, \lambda\Lambda_\mu\right) \mathcal{W}\left(\mathbf{W}, \nu\right) \left|\Lambda_\mu\right|^{1/2} \; d\mu \; d\Lambda_\mu. \end{aligned} \tag{A.28}$$

The $\mathcal{F}$ variables or functions of interest in this report contain terms in μ and, for the predictive posterior, terms in Λ_μ. It may be noted that common factors not containing integration terms have been canceled.

While the integrals in Equations A.27 and A.28 may have analytical solutions, the authors have been unable to derive them nor discover solutions from literature searches. DeGroot provides a terse solution to similar equations involving normal distributions and a normal Wishart prior [15] (as referenced without proof by Murphy [16]) but the integrals under study here are more complex because of the product of determinants of Λ_μ summed with the various Λ_{z_i}. Faced with this problem, it was determined to use probabilistic programming to obtain a numerical solution for the remaining integrals.

B STAN PROBABILISTIC PROGRAM FOR 2D CLUSTERING

The Stan source code file used in the 2D clustering algorithm contained the following lines:

```
functions {
  real likelihood_log(row_vector z, row_vector mu,
                      matrix Yz, matrix Yu) {
    real logprob;
    logprob = 1/2.0 * (-log_determinant(Yz+Yu) + log_determinant(Yu));
    logprob = logprob - 1/2.0 * (z*Yz*z' + mu*Yu*mu' -
              (Yz*z'+Yu*mu')'*inverse(Yz+Yu)*(Yz*z'+Yu*mu'));

    return logprob;
  }
}

data {
  int<lower=2> dim; // dimensionality
  int<lower=1> N; // number of data points
  row_vector[dim] y[N]; // data
  matrix[dim, dim] prec_noise[N]; // all precision matrices
}

parameters {
  cov_matrix[dim] Lambda_data; // precision matrix
  row_vector[dim] mu; // mean vector
}

transformed parameters {
}

model {
  matrix[dim,dim] W = [[1,0],[0,1]]; // initialize prior parameters
  real nu = 3;
  real lambda = 2;
  row_vector[dim] mu0 = [-8,12];

  Lambda_data ~ wishart(nu, W); // normal wishart prior
  mu ~ multi_normal_prec(mu0, lambda*Lambda_data);

  for (i in 1:N) {
    y[i] ~ likelihood(mu, prec_noise[i], Lambda_data);
  }
}
```

This page intentionally left blank.

GLOSSARY

FIML	Full Information Maximum Likelihood
HMC	Hamiltonian Monte Carlo
LL	Lincoln Laboratory
MAP	Maximum A Posteriori
MCMC	Markov Chain Monte Carlo
MI	Multiple Imputation
MIT	Massachusetts Institute of Technology
MVN	Multi-Variate Normal
NaN	Not a Number
NA	Not Available
NP	Nondeterministic Polynomial
NUTS	No-U-Turn Sampler
NW	Normal-Wishart
PDF	Probability Density Function or Probability Distribution Function
ROC	Receiver Operating Characteristics

This page intentionally left blank.

REFERENCES

[1] D.B. Rubin, "Inference and missing data," *Biometrika* 63(3), 581–592 (1976).

[2] D.B. Rubin and N. Schenker, "Multiple imputation for interval estimation from simple random samples with ignorable nonresponse," *Journal of the American Statistical Association* 81(394), 366–374 (1986).

[3] C.K. Enders, "A primer on maximum likelihood algorithms available for use with missing data," *Structural Equation Modeling* 8(1), 128–141 (2001).

[4] C.K. Enders and D.L. Bandalos, "The relative performance of full information maximum likelihood estimation for missing data in structural equation models," *Structural Equation Modeling* 8(3), 430–457 (2001).

[5] S. Lloyd, "Least square quantization in pcm. bell telephone laboratories paper. published in journal much later: Lloyd, sp: Least squares quantization in pcm," *IEEE Trans. Inform. Theor.(1957/1982) Google Scholar* (1957).

[6] J.A. Hartigan and M.A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 28(1), 100–108 (1979).

[7] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, Society for Industrial and Applied Mathematics (2007), pp. 1027–1035.

[8] M.D. Hoffman and A. Gelman, "The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo," *arXiv:1111.4246* (2011), URL http://arxiv.org/abs/1111.4246.

[9] A.W.v.d. Vaart, *Asymptotic Statistics*, Cambridge Series in Statistical and Probabilistic Mathematics, Cambridge University Press (1998).

[10] J.L. Doob, "Application of the theory of martingales," *Le calcul des probabilites et ses applications* pp. 23–27 (1949).

[11] Stan Development Team, "Stan modeling language users guide and reference manual, version 2.17.0," The Stan Project (2017), URL https://github.com/stan-dev/stan/releases/download/v2.17.0/stan-reference-2.17.0.pdf.

[12] J. Annis, B.J. Miller, and T.J. Palmeri, "Bayesian inference with stan: A tutorial on adding custom distributions," *Behavior Research Methods* 49(3), 863–886 (2017), URL https://doi.org/10.3758/s13428-016-0746-9.

[13] A. Gelman and D.B. Rubin, "Inference from Iterative Simulation Using Multiple Sequences," *Statist. Sci.* 7, 457–472 (1992).

[14] M. Mahajan, P. Nimbhorkar, and K. Varadarajan, "The planar k-means problem is np-hard," in *Proceedings of the 3rd International Workshop on Algorithms and Computation*, Berlin,

Heidelberg: Springer-Verlag (2009), WALCOM '09, pp. 274–285, URL http://dx.doi.org/10.1007/978-3-642-00202-1_24.

[15] M.H. DeGroot, *Optimal statistical decisions*, McGraw-Hill, Inc (1970).

[16] K.P. Murphy, "Conjugate bayesian analysis of the gaussian distribution," University of British Columbia (2007), URL http://www.cs.ubc.ca/{~}murphyk/Papers/bayesGauss.pdf.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
26-02-2019	Technical Report	

4. TITLE AND SUBTITLE

Probabilistic Programming with Missing Data

5a. CONTRACT NUMBER
FA8721-05-C-0002 & FA8702-15-D-0001

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

D. Suen, K.L. Nahabedian, M.J.Yee, and M.B. Hurley

5d. PROJECT NUMBER
2236

5e. TASK NUMBER
3301

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

MIT Lincoln Laboratory
244 Wood Street
Lexington, MA 02421-6426

8. PERFORMING ORGANIZATION REPORT NUMBER

TR-1238

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
ASD R&E
3030 Defense Pentagon
Washington, DC 20301-3030

10. SPONSOR/MONITOR'S ACRONYM(S)
ASD R&E

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release: distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This report summarizes work performed to develop a computer algorithm that is capable of handling missing data fields in multivariate data sets. The results presented here are based upon prior work which examined the applicability of inverse covariance matrices, or precision matrices, to representing missing data as zero eigenvalues in the precision matrices. The prior work used maximum a posteriori (MAP) estimates for a combination of normally distributed multivariate data with normally distributed multivariate measurement errors and assumed that the prior probability distributions for means and precision matrices were uniform.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER *(include area code)*
Unclassified	Unclassified	Unclassified	Same as report	52	