GAUSSIAN COPULA FOR MIXED DATA WITH MISSING VALUES: MODEL ESTIMATION AND IMPUTATION

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Yuxuan Zhao

May 2022

GAUSSIAN COPULA FOR MIXED DATA WITH MISSING VALUES: MODEL ESTIMATION AND IMPUTATION

Yuxuan Zhao, Ph.D.
Cornell University 2022

Missing data imputation forms the first critical step of many data analysis pipelines. For practical applications, imputation algorithms should produce imputations that match the true data distribution and handle data of mixed types. This dissertation develops new imputation algorithms for data with many different variable types, including continuous, binary, ordinal, and truncated and categorical values, by modeling data as samples from a Gaussian copula model. This semiparametric model learns the marginal distribution of each variable to match the empirical distribution, yet describes the interactions between variables with a joint Gaussian that enables fast inference, imputation with confidence intervals, and multiple imputation. This dissertation also develops specialized extensions to handle large datasets (with complexity linear in the number of observations) and streaming datasets (with online imputation).

BIOGRAPHICAL SKETCH

Yuxuan Zhao was born and raised in Bozhou, Anhui, China. After graduating from junior high school in his hometown in 2010, he enrolled in Hefei No.1 high school in Hefei, the provincial capital city of Anhui. In 2013, he graduated from high school and began his undergraduate study in Nankai university, majoring in mathematics. During his four years of undergraduate study, he discovered that he was more interested in applying mathematics to real-world problems than theoretical mathematics. He graduated from Nankai University in 2017 and then went on to Cornell University for his Ph.D. in statistics. There he has been advised by Prof. Madeleine Udell. His five years of academical statistics research are dedicated to new methodology for heterogeneous data. After graduation, he will keep solving real-world problems using mathematics and statistics as an industry researcher.

Dedicated to my father Jin Zhao and my mother Weili Wang.

ACKNOWLEDGEMENTS

It has not been easy for me to complete this Ph.D., and it feels unreal when I finally reach the finish line. It's been a long journey filled with ignorance, confusion, disorientation, and exhaustion. I am truly grateful to everyone who helped me on this journey. Now I can confidently identify myself as a qualified doctor.

First of all, I am extremely grateful to my advisor, Madeleine Udell, for her guidance in research and constant support, encouragement and trust. She is passionate on research and always has new ideas. The majority of my research work arose as results of our discussion. She is one of the best academic writers and speakers I have ever met and has taught me a lot on how to write a good paper and give a good presentation. Those knowledge would benefit me for life.

I also want to thank my committee members Thorsten Joachims and Yang Ning. They graciously answered my research questions and provided numerous valuable research suggestions, which aided in the development of this dissertation. I am also grateful to David Matteson for his mentoring at my early Ph.D. stage. Additionally, I want to thank my collaborators Benjamin Risk and Alex Townsend for the effort we put together to have a good paper.

I also want to express my gratitude to my friends at Cornell: Xin Bing, Huijie Feng, Mo He, Yujia Ma, Xiangxiang Wang, Peter Wu, Yaosheng Xu, and many more. Their presence helps me adapt to the life in a new country quickly and brightens my Ph.D. life. Although they are now scattering all over the world, my time with them in Ithaca remains vivid in memory. It has been a great pleasure to get to know them in the gorgeous Ithaca.

My father, Jin Zhao, my mother, Weili Wang, and my partner, Xiaoyi Zhu, are the people I owe the most to. My parents always show me unconditional love. Many people in their generation who grew up in small towns of China did not approve of sending their children abroad for a five-year Ph.D. program, but my parents have always respected my decision and supported me in becoming who I want to be. My partner, Xiaoyi, has been with me through many of my darkest moments in my Ph.D. time. She is always there to raise me up. My life has became much more colorful since she has been in it. Together each of us becomes a better and happier person.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Missing data is ubiquitous in modern datasets, yet most machine learning algorithms and statistical models require complete data. Thus missing data imputation forms the first critical step of many data analysis pipelines. The difficulty is greatest for mixed datasets, including continuous, binary, ordinal, count, truncated and categorical variables. Mixed datasets may appear either as a single dataset recording different types of attributes or an integrated datasets from multiple sources. For example, social survey datasets are generally mixed since they often contain age (continuous) and Likert scales (ordinal) measuring how strongly a respondent agrees with certain stated opinions, such as the five category scale: strongly disagree, disagree, neither agree or disagree agree, strongly agree. The Cancer Genome Atlas Project is an example of integrated mixed dataset: it contains gene expression (continuous), mutation (binary) and microRNA (count) data. Imputation may be challenging even for datasets with only continuous variables if variables have very different scales and variability.

The Gaussian copula model nicely addresses the challenges of modeling mixed data by separating the multivariate interaction of the variables from their marginal distributions [60, 45, 30]. Specifically, this model posits that each data vector is generated by first drawing a latent Gaussian vector and then transforming it to match the observed marginal distribution of each variable. A copula correlation matrix fully specifies the multivariate interaction and is invariant to strictly monotonic marginal transformations of the variables.

1.1 Contribution

This dissertation develops methodology for estimating the Gaussian copula model from incomplete data, imputing the missing values using the estimated model and quantifying the imputation uncertainty. All proposed methodologies handle mixed datasets that may contain arbitrary continuous variables, ordinal variables (including binary as a special case), and truncated, and categorical variables naturally.

Chapter 2 introduces the definition of Gaussian copula model for mixed data, how to estimate it from incomplete data and how to impute missing values using an estimated model. The methodology developed in this chapter is mostly suited for skinny datasets, those have no more than 1000 features, since the model estimation algorithm has cubic time complexity in terms of the number of features. Chapter 3 instead aims at wide (or high dimensional) datasets with many features. In this chapter, we develop the low rank Gaussian copula model which imposes a low rank structure on the copula correlation matrix, and thus reduces the model estimation algorithm's cubic time complexity to linear, in terms of the number of features. We also provide theoretical guarantee on the imputation quality for incomplete data sampled from the low rank Gaussian copula model. Chapter 4 derives two types of measures to quantify the uncertainty of Gaussian copula model based imputation. For a specific missing entry, the variability of its imputation error is characterized. For example, imputation confidence intervals are provided. For multiple missing entries, a measure ranking the relative imputation quality is provided. One can use this measure to select the most reliable imputed entries, which may be useful for the top-k recommendation task in collaborative filtering. Chapter 5 develops an online missing data imputation algorithm, by incrementally updating the Gaussian copula model over sequentially arrived mini-batch of data. Concretely, we maintain an estimated Gaussian copula model, and as new data comes in, we immediately impute its missing entries using the maintained model the and then update the model using the new data. We also develop a much faster model estimation algorithm than that in Chapter 2 using the incremental update. By tracking the magnitude of the copula correlation update between two mini-batch of data, Chapter 6 provides a new method to detect change points in the multivariate dependence structure in online data. Before Chapter 7, all methods assume that every variable admits a total order, which excludes categorical variables. Chapter 7 shows how to extend a Gaussian copula model to handle categorical variables, and thus all aforementioned methods can deal with categorical and ordered variable mixed data. At last, Chapter 8 introduces our implemented software packages in Python and R and demonstrates their usage.

This dissertation is based on [103, 102, 105, 104] but reorganizes their contents for a coherent and concise presentation. Generally speaking, Chapter 2 comes from [103], Chapter 3 and Chapter 4 come from [102]. Chapter 5 and Chapter 6 come from [105]. Chapter 8 comes from [104]. Every paper, however, contributes to every chapter in some way. An exception is Chapter 7: it is our most recent work, and it has not yet been made public outside of this dissertation.

1.2 Background

1.2.1 Missing mechanism

Missing data mechanisms include: (1) missing completely at random (MCAR) meaning the missingness does not depend on the data value; (2) missing at random (MAR), meaning the missingness only depends on the observed data value; (3) missing not at random (MNAR), meaning the missingness depends on the missing data value [59]. As it will be stated in Section 2.2, the Gaussian copula model has two parts of parameters: a vector-valued function $\mathbf{f}$, which describes the marginal distribution, and a correlation matrix Σ, which describes the multivariate dependence structure. The MCAR assumption is needed to consistently estimate $\mathbf{f}$. If the true $\mathbf{f}$ is known, the MAR assumption suffices to consistently estimate Σ. We assume MCAR throughout this dissertation.

Extending our methods to MNAR setting is important future work. However, we find empirically our imputation methods developed under MCAR still performs reasonably well in the MNAR setting. Indeed, many different missing patterns may be called MNAR, and imputation methods designed for one MNAR mechanism do not necessarily outperform on other MNAR data due to this heterogeneity.

1.2.2 Notation and evaluation metric

Notation

Define $[p] = \{1, \ldots, p\}$ for $p \in \mathbb{N}^+$. Let $\mathbf{x} = (x_1, \ldots, x_p) \in \mathbb{R}^p$ be a random vector. Let $\mathbf{X} \in \mathbb{R}^{n\times p}$ be a matrix whose rows correspond to observations and columns to variables. We refer to the i-th row, j-th column, and (i, j)-th element as $\mathbf{x}^i, \mathbf{X}_j$ and x^i_j, respectively. We use f for a function mapped into $\mathbb{R}$ (scalar function) and $\mathbf{f} = (f_1, \ldots, f_p)$ for a function mapped into $\mathbb{R}^p$ (vector function).

For a subset $I \subset [p]$, we use $\mathbf{x}_I$ to denote the subvector of $\mathbf{x}$ with entries in I for $\mathbf{x} \in \mathbb{R}^p$, $\mathbf{W}_I$ to denote the submatrix of $\mathbf{W}$ with rows in I and $\mathbf{W}_{I,J}$ to denote the submatrix of $\mathbf{W}$ with rows in I and columns in J for matrix $\mathbf{W} \in \mathbb{R}^{p\times k}$, $\mathbf{f}_I$ to denote the subvector of $\mathbf{f}$ with entries in I for vector function $\mathbf{f} \in \mathbb{R}^p$.

Let $\mathcal{M}, \mathcal{O} \subset [p]$ denote missing and observed dimensions, respectively. Thus $\mathbf{x} = (\mathbf{x}_{\mathcal{O}}, \mathbf{x}_{\mathcal{M}})$ for a random vector $\mathbf{x}$, and $\mathbf{x}^i = (\mathbf{x}^i_{\mathcal{O}_i}, \mathbf{x}^i_{\mathcal{M}_i})$ for a realized sample $\mathbf{x}^i$.

We use ϕ and Φ for the probability density function (PDF) and cumulative distribution function (CDF) of the one-dimensional standard normal. We use $\phi(\cdot; \mu, \Sigma)$ for the PDF of a normal vector with mean μ and covariance matrix Σ. Denote the vector ℓ_2 norm as $\|\cdot\|_2$ and the matrix Frobenius norm as $\|\cdot\|_F$.

The elliptope $\mathcal{E} = \{Z \succeq 0 : \text{diag}(Z) = 1\}$ is the set of correlation matrices. For a positive definite covariance Σ, we use $P_{\mathcal{E}}(\Sigma)$ to denote its corresponding correlation matrix: for $D = \text{diag}(\Sigma)$, $P_{\mathcal{E}}(\Sigma) = D^{-1/2}\Sigma D^{-1/2}$. For a subset $\Omega \subset [n]\times[p]$ containing two dimensional coordinates and matrix $\mathbf{X} \in \mathbb{R}^{n\times p}$, we use $P_\Omega(\mathbf{X})$ to denote the matrix that agrees with $\mathbf{X}$ at coordinates in Ω but 0 otherwise.

We say random variables $x = y$ and random vectors $\mathbf{x} = \mathbf{y}$ if their CDF match.

Evaluation metric

For incomplete data observation $\mathbf{X} \in \mathbb{R}^{n\times p}$ observed at locations Ω and its imputation $\hat{\mathbf{X}}$ which agrees with $\mathbf{X}$ at Ω, we use metrics to the magnitude of their error. When all columns of $\mathbf{X}$ are approximately in the same scale, we mostly use the popular mean absolute error, mean squared error (MSE), root mean squared error (RMSE), and normalized root mean squared error (NRMSE), defined as $\|P_{\Omega^c}(\mathbf{X} - \hat{\mathbf{X}})\|_F / \|P_{\Omega^c}(\mathbf{X})\|_F$. When columns of $\mathbf{X}$ have very different scales or data types, to measure the imputation error on columns in I, we define and use a scaled mean absolute error (SMAE):

$$\text{SMAE} := \frac{1}{|I|}\sum_{j\in I}\frac{\|\hat{\mathbf{X}}_j - \mathbf{X}_j\|_1}{\|\mathbf{X}_j^{\text{med}} - \mathbf{X}_j\|_1},$$

where $\hat{\mathbf{X}}_j, \mathbf{X}_j^{\text{med}}$ are the imputed values and observed median for j-th column, respectively. The estimator's SMAE is smaller than 1 if it outperforms column median imputation. For each data type, the SMAE can be computed on corresponding columns.

CHAPTER 2

IMPUTATION VIA GAUSSIAN COPULA

This chapter introduces the general framework to fit a Gaussian copula model and impute missing values for continuous, ordinal (including binary and count as special cases) and truncated mixed data. It is mostly based on [103] except that the support for truncated variables is based on [104].

We introduce the Gaussian copula model for mixed data in Section 2.2, then show how to impute the missing values with given model estimates in Section 2.3. The model estimation algorithms come next in Section 2.4.

2.1 Introduction

Missing data is endemic and usually represents a large proportion in modern datasets. Missing value imputation generally precedes other analysis, since most machine learning algorithms and statistical models require complete observations. Imputation quality can strongly influence subsequent analysis. The difficulty in imputing missing data is greatest for mixed datasets: those that include real, Boolean, ordinal, count and truncated data — are a fixture of modern data analysis. Ordinal data is particularly common in survey datasets. For example, Netflix users rate movies on a scale of 1-5. Social surveys may roughly bin respondents' income or level of education as an ordinal variable, and ordinal Likert scales measure how strongly a respondent agrees with certain stated opinions. Binary variables may be considered a special case of an ordinal with two levels. Health data often contains ordinals that result from patient surveys or from coarse binning of continuous data into, e.g., cancer stages 0–IV or overweight vs obese patients.

To exploit the information in mixed data, imputation must account for the interaction among variables of different types. Unfortunately, the joint distribution of mixed data can be complex. Existing parametric models are either too restrictive [59] or require priori knowledge of the data distribution [93]. Nonparametric methods, such as MissForest [84], based on random forests, and imputeFAMD [4], based on principal components analysis, tend to perform better. However, these two methods treat ordinal data as categorical, losing valuable information about the order. Further, they can only afford a limited number of categories.

It is tempting, but dangerous, to treat ordinal data with many levels as continuous. For example, the ordinal variable "Weeks Worked Last Year" from the General Social Survey dataset takes 48 levels, but 74% of the population worked either 0 or 52 weeks. Imputation that treats this variable as continuous (e.g., imputing with the mean) works terribly! As another example, consider using low rank matrix completion [16, 76, 53, 64] to impute missing entries in a movie rating datasets using a quadratic loss. This loss implicitly treats ratings encoded as 1–5 as numerical values, so the difference between ratings 3 and 4 is the same as that between ratings 4 and 5. Is this true? How could we tell?

A more sensible (and powerful) model treats ordinal data as generated by thresholding continuous data, as in [77, 78]. Figure 2.1 illustrates how correlations can by garbled by treating such data as continuous. Our work builds on this intuition to model mixed data through the Gaussian copula model [45, 60, 30, 34], which assumes the observed vector is generated by transform-

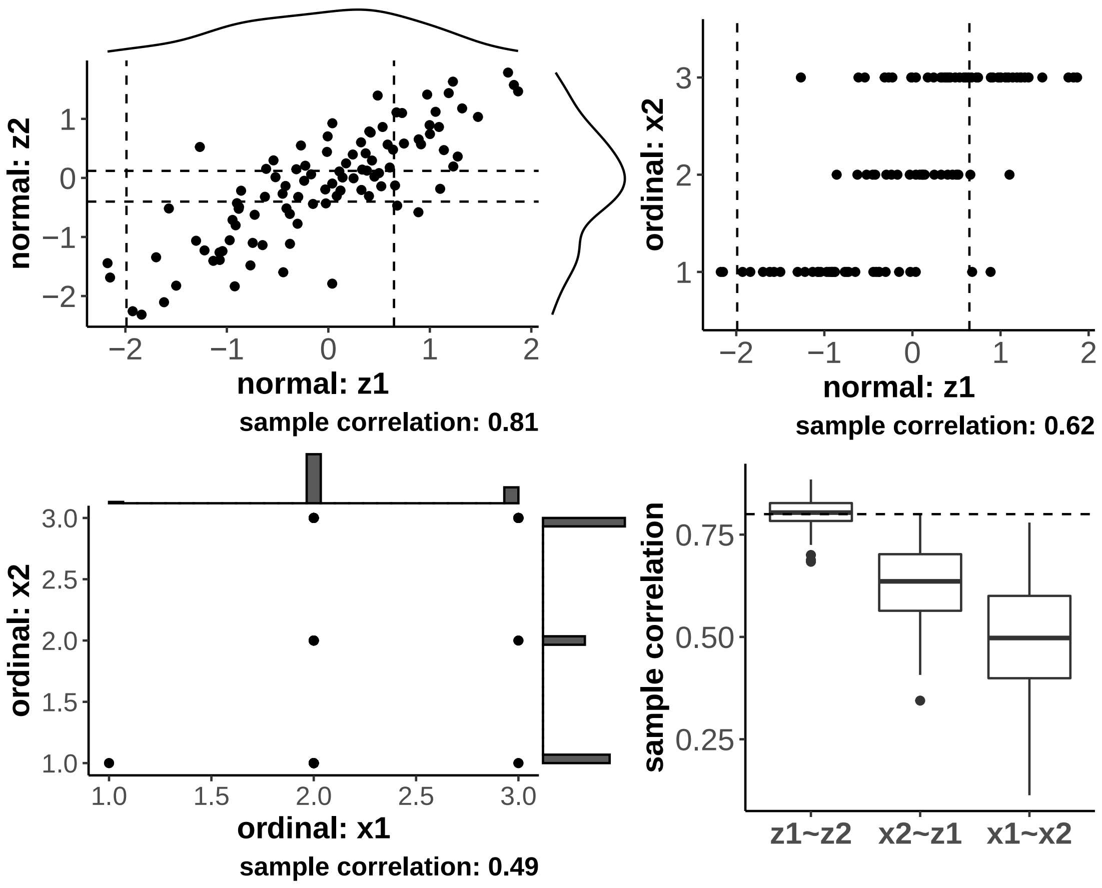


Figure 2.1: Draw (z_1, z_2) from a binormal with correlation 0.8. Discretize z_1 to x_1, z_2 to x_2 on random cutoffs. Top two and bottom left panels plot one repetition. Dashed lines mark the cutoffs. Bottom right panel plots the sample correlation over 100 repetitions. Dashed line marks the truth.

ing each marginal of a latent normal vector. Under this model, we associate each variable (both ordinal and continuous) with a latent normal variable. Each ordinal level corresponds to an interval of values of the corresponding latent normal variable.

Contribution

We propose an efficient EM algorithm to estimate a Gaussian copula model with incomplete mixed data and show how to use this model to impute missing values. Our method outperforms many state-of-the-art imputation algorithms for various real datasets including social survey data (whose columns have a varying number of ordinal levels), movie rating data (high missing ratio), music tagging data (binary data), etc. The proposed method has several advantages: the method has no hyper-parameters to tune and is invariant to coordinate-wise monotonic transformations in the data. Moreover, the fitted copula model is interpretable and can reveal statistical associations among variables, which is useful for social science applications. To our best knowledge, our proposed algorithm is the first frequentist approach to fit the Gaussian copula model with incomplete mixed data. Moreover, it is much faster than the existing Bayesian MCMC algorithm for the same model [45]; given the same time budget, our method produces substantially more accurate estimates.

Related work

Modeling mixed data with the Gaussian copula model has been studied using both frequentist approaches [30, 34] and Bayesian approaches [45, 68, 25]. In [68, 25], the authors further assume the latent normal vector is generated from a factor model. When all variables are ordinal, the Gaussian copula model is equivalent to the probit graphical model [39]. However, all these previous work focuses on model estimation and theoretical properties of the estimators, and has overlooked the potential of these models for missing value imputation.

In fact, the frequentist parameter estimation methods proposed [30, 34, 39] assume complete data; so these methods cannot perform imputation. Among Bayesian approaches, MCMC algorithms [45, 68, 25] can fit the copula model with incomplete data and impute missing values. However, to use these models, one must select the number of factors for the models in [68, 25]. The sensitivity of these models to this parameter makes it a poor choice in practice for missing value imputation.

The implementation of [45] is still the best method available to fit a Gaussian copula model for incomplete mixed data. An important case study of this method [46] applies the multiple imputation to sociological data analysis. However, the method is slow and sensitive: the burn-in and sampling period must be carefully chosen for MCMC to converge, and many iterations are often required, so the method does not scale to even moderate size data, which limits its use in practice. Our model matches that of [45], but our EM algorithm runs substantially faster.

The generalized low rank models (GLRM) framework [91] handles missing values imputation for mixed data using a low rank model with appropriately chosen loss functions to ensure proper treatment of each data type. However, choosing the right loss functions for mixed data is challenging.

A few papers share our motivation: for example, early papers by Rennie and Srebro [77, 78] proposed a thresholding model to generate ordinals from real low rank matrices. Monotonic transformations of a latent low rank matrix are estimated in [36], but the method performs poorly in practice. xPCA [3] posits that the mixed data are generated by marginally transforming the columns of the sum of a low rank matrix and isotropic Gaussian noise. While their marginal transformation coincides with the Gaussian copula model, their setup greatly differs in that it cannot identify the correlations between variables.

While low rank matrix completion methods scale well to large datasets, the low rank assumption is too weak to generalize well on long skinny datasets. Hence low rank methods tend to work well on "square-ish" datasets ($n \sim p$) [90], while the copula methods proposed here work better on long, skinny datasets.

2.2 Gaussian Copula model

The Gaussian copula models complex multivariate distributions through transformations of a latent Gaussian vector. We call a random variable $x \in \mathbb{R}$ continuous when it is supported on an interval. We can match the marginals of any continuous random vector $\mathbf{x}$ by applying a strictly monotone function to a random vector $\mathbf{z}$ with standard normal marginals. Further, the required function is unique, as stated in Lemma 1.

Lemma 1. Suppose $\mathbf{x} \in \mathbb{R}^p$ is a continuous random vector with CDF F_j for each coordinate $j \in [p]$, and $\mathbf{z} \in \mathbb{R}^p$ is a random vector with standard normal marginals. Then there exists a unique elementwise strictly monotone function $\mathbf{f}(\mathbf{z}) := (f_1(z_1), \ldots, f_p(z_p))$ such that

$$x_j = f_j(z_j) \quad \text{and} \quad f_j = F_j^{-1} \circ \Phi, \quad j \in [p], \tag{2.1}$$

where Φ is the standard normal CDF.

Notice the functions $\{f_j\}_{j=1}^p$ in Eq. (2.1) are strictly monotone, so their inverses exist. Define $\mathbf{f}^{-1} = (f_1^{-1}, \ldots, f_p^{-1})$. Then $\mathbf{z} = \mathbf{f}^{-1}(\mathbf{x})$ has standard normal marginals, but the joint distribution of $\mathbf{z}$ is not uniquely determined. The Gaussian copula model (or equivalently nonparanormal distribution [60]) further assumes $\mathbf{z}$ is jointly normal: $\mathbf{f}(\mathbf{z}) = \mathbf{x}$ and $\mathbf{z} \sim \mathcal{N}_p(\mathbf{0}, \Sigma)$ for a correlation matrix Σ.

This model is semiparametric: it comprises nonparametric functions $\mathbf{f}$ (denoted as the marginals) and parametric copula correlation matrix Σ. The monotone $\mathbf{f}$ establishes the mapping between observed $\mathbf{x}$ and latent normal $\mathbf{z}$, while Σ fully specifies the distribution of $\mathbf{z}$. Further, the correlation Σ is invariant to elementwise strictly monotone transformation of $\mathbf{x}$. Concretely, if $\mathbf{x}$ follows the Gaussian copula with correlation Σ and $\mathbf{y} = \mathbf{g}(\mathbf{x})$ with elementwise strictly monotone $\mathbf{g}$, then $\mathbf{y}$ also follows the Gaussian copula with correlation Σ, although with a different marginal $\mathbf{g} \circ \mathbf{f}$. Thus the Gaussian copula separates the multivariate interaction Σ from the marginal distribution.

Modeling ordinal and truncated variables

When f_j is strictly monotone, $x_j = f_j(z_j)$ must be continuous. On the other hand, when f_j is monotone but not strictly monotone, x_j takes discrete values in the range of f_j and can model ordinals. Thus for ordinals, f_j will not be invertible. For convenience, we define a set-valued inverse $f_j^{-1}(x_j) := \{z_j : f_j(z_j) = x_j\}$. When the ordinal x_j has range $[k]$, Lemma 2 states that the only monotone function f_j mapping continuous z_j to x_j is a cutoff function, defined for some parameter $S \subset \mathbb{R}$ as

$$\text{cutoff}(z; \mathbf{S}) := 1 + \sum_{s \in \mathbf{S}} \mathbb{1}(z > s) \text{ for } z \in \mathbb{R}.$$

Lemma 2. Suppose $x \in \mathbb{R}$ is an ordinal random variable with range $[k]$ and probability mass function $\{p_l\}_{l=1}^k$ and $z \in \mathbb{R}$ is a continuous random variable with CDF F_z. Then $f = \text{cutoff}(z; \mathbf{S})$ is the unique monotone function f that satisfies $x = f(z)$, where $\mathbf{S} = \{s_l = F_z^{-1}\left(\sum_{t=1}^{l} p_t\right) : l \in [k-1]\}$.

For example, in recommendation system we can think of the discrete ratings as obtained by rounding some ideal real valued score matrix. The rounding procedure amounts to apply a cutoff function. See Figure 2.2 for an example of cutoff function.

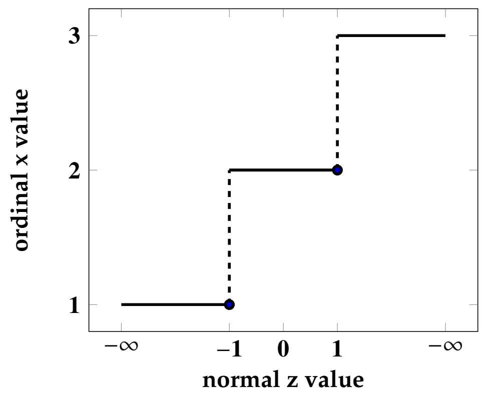


Figure 2.2: Cutoff function $f(\cdot)$ with cutoffs $\{-1, 1\}$ maps continuous z to ordinal $x \in \{1, 2, 3\}$.

We can even find a unique f_j for every one-sided or two-sided truncated variable. A variable x_j truncated below at $x = \alpha$ has a CDF:

$$F(x) = \mathbf{P}(x = \alpha)\mathbb{1}(x \geq \alpha) + (1 - \mathbf{P}(x = \alpha))\tilde{F}(x),$$

where $\tilde{F}(x)$ is the CDF of a random variable satisfying $\tilde{F}(\alpha) = 0$. An upper truncated variable and two sided truncated variable are defined similarly. The CDF of a truncated variables is a strictly monotonic function with a step either on the left (lower truncated) or the right (upper truncated) or both (two sided truncated).

The expression of f_j as well as their set inverse are summarized in Table 2.1. In short, f_j explains how the data is generated, while f_j^{-1} denotes available information for model inference given the observed data.

Type		
Continuous	Distribution	x has CDF $F(x)$.
	$f(z)$	$F^{-1}(\Phi(z))$
	$f^{-1}(x)$	$\Phi^{-1}(F(x))$
Ordinal	Distribution	x has PMF $\mathbf{P}(x=i)=p_i$, for $i=1,...,k$.
	$f(z)$	$\max\left\{i : \sum_{l=0}^{i-1} p_l \leq \Phi(z) < \sum_{l=0}^{i} p_l\right\}$, with $p_0 = 0$
	$f^{-1}(x)$	$\left\{z : \sum_{l=0}^{x-1} p_l \leq \Phi(z) < \sum_{l=0}^{x} p_l\right\}$, with $p_0 = 0$
Truncated to $x \in (\alpha,\beta)$	Distribution	$\mathbf{P}(x=\alpha)=p_\alpha$, $\mathbf{P}(x=\beta)=p_\beta$, and CDF $\tilde{F}(x)$ conditional on $x \in (\alpha,\beta)$, which satisfies $\tilde{F}(\alpha)=0$ and $\tilde{F}(\beta)=1$.
	$f(z)$	$\begin{cases} \alpha, & \Phi(z) \leq p_\alpha \\ \tilde{F}^{-1}\left(\frac{\Phi(z)-p_\alpha}{1-p_\alpha-p_\beta}\right), & \Phi(z) \in (p_\alpha, 1-p_\beta) \\ \beta, & \Phi(z) \geq 1-p_\beta \end{cases}$
	$f^{-1}(x)$	$\begin{cases} \{z : \Phi(z) \leq p_\alpha\}, & x=\alpha \\ \Phi^{-1}\left(p_\alpha + (1-p_\alpha-p_\beta)\tilde{F}(x)\right), & x \in (\alpha,\beta) \\ \{z : \Phi(z) \geq 1-p_\beta\}, & x=\beta \end{cases}$

Table 2.1: For any random variable x admitting a total order, there exists a unique monotonic transformation f such that $f(z) = x$ for a random standard Gaussian z. For each data type of x, this table includes its distribution specification, the marginal f, and the set inverse $f^{-1}(x) = \{z : f(z) = x\}$ of the marginal. Three different types of truncated variables are summarized together: (1) $\alpha = -\infty$ and $p_\alpha = 0$ corresponds to lower truncated x; (2) $\beta = \infty$ and $p_\beta = 0$ corresponds to upper truncated x; (3) finite α,β and positive p_α, p_β corresponds to two sided truncated x. $\Phi(\cdot)$ denotes the CDF of a standard normal variable.

Gaussian copula model for mixed data

To extend the Gaussian copula to mixed data, we simply specify a f_j corresponding to a desired distribution of x_j, as in Table 2.1. As before, the correlation Σ remains invariant to elementwise strictly monotone transformations. The main difference is that while $f_j^{-1}(x_j)$ is a single number when x_j is continuous, it is an interval when x_j is ordinal. Now we summarize the definition of Gaussian copula for general mixed data. The only requirement for the data $\mathbf{x}$ is that each marginal x_j admits a total order: for any two realized values of x_j, x_j^1 and x_j^2, either $x_j^1 > x_j^2$ or $x_j^1 \leq x_j^2$.

Definition 1. We say a random vector $\mathbf{x} \in \mathbb{R}^p$ follows the Gaussian copula $\mathbf{x} \sim$

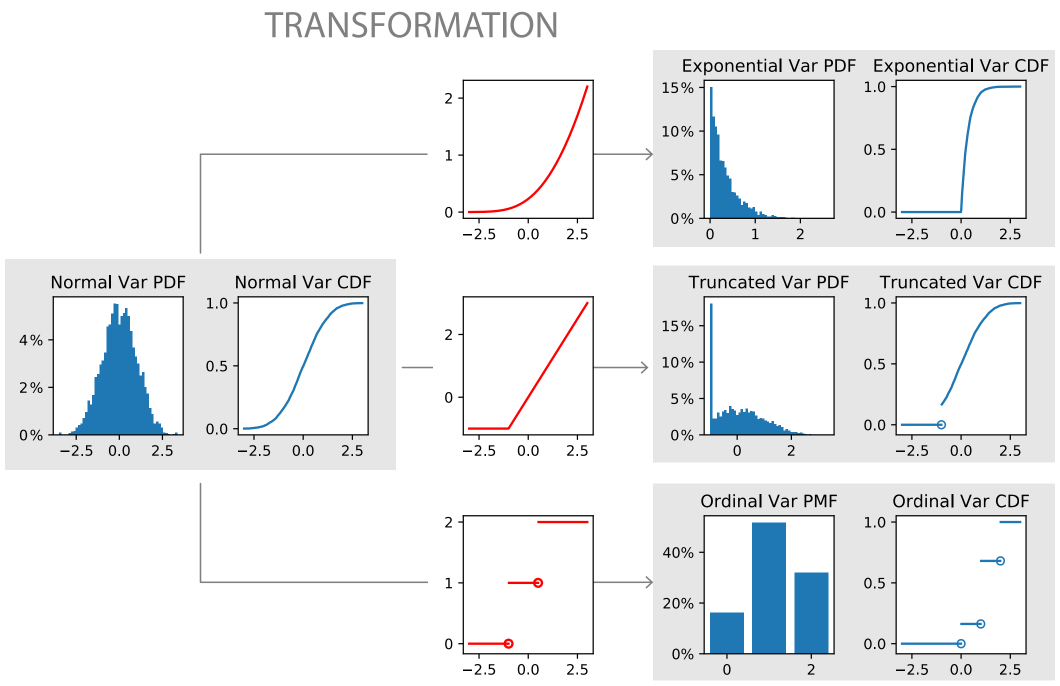


Figure 2.3: Three monotoic transformations of a Gaussian variable. The third column depicts the transformations that map the data distribution, visualized as both PDF (histogram approximation) and CDF (analytical form), in the left two columns to the data distribution in the right two columns.

GC($\Sigma, \mathbf{f}$) with parameters Σ and $\mathbf{f}$ if (1) each marginal of $\mathbf{x}$ admits a total order; (2) there exists a correlation matrix Σ and elementwise monotone function $\mathbf{f}$: $\mathbb{R}^p \to \mathbb{R}^p$ such that $\mathbf{f}(\mathbf{z}) = \mathbf{x}$ for $\mathbf{z} \sim \mathcal{N}_p(\mathbf{0}, \Sigma)$.

Fig. 2.3 depicts how a Gaussian variable is transformed into an exponential variable, a lower truncated variable, and an ordinal variable. Fig. 2.4 depicts the dependency structure induced by a Gaussian copula model: it plots randomly drawn samples from 2D Gaussian copula model with the same marginal distributions from Fig. 2.3. It shows that the Gaussian copula model is much more expressive than the multivariate normal distribution. Fig. 2.5 also shows the correspondence between the observed variables $\mathbf{x}$ and the latent normal variables $\mathbf{z}$.

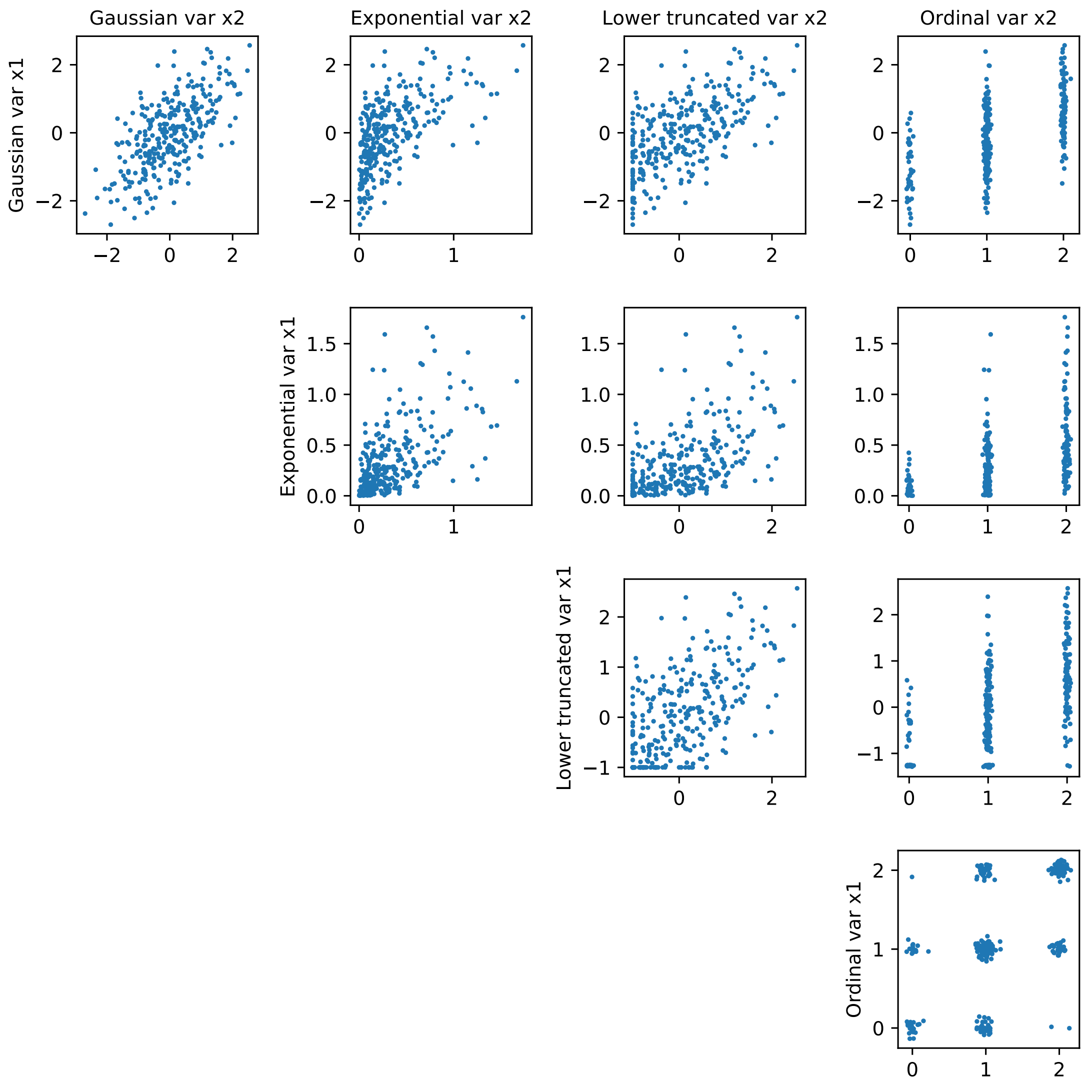


Figure 2.4: Scatterplot of samples from several 2D Gaussian copula models with different marginals. The data is generated by sampling (z_1, z_2) from a 2D Gaussian distribution with zero mean, unit variance and .65 correlation and computing $x_1 = f_1(z_1)$ and $x_2 = f_2(z_2)$, where f_1 and f_2 denote the transformations corresponding to the marginals for each model. For Gaussian marginals (1st row and 1st column), the transformation is the identity. For other marginals, the corresponding transformations are plotted as the third column of Fig. 2.3.

2.3 Imputation

So far we have introduced a very flexible model for mixed data. Our interest is to investigate missing value imputation under this model. Concretely, suppose the data matrix $\mathbf{X}$ has rows $\mathbf{x}^1, \ldots, \mathbf{x}^n \overset{i.i.d.}{\sim} \mathrm{GC}(\Sigma, \mathbf{f})$ and $\mathbf{x}^i = (\mathbf{x}^i_{O_i}, \mathbf{x}^i_{\mathcal{M}_i})$ for $i \in [n]$. we first estimate $\hat{\Sigma}$ and $\hat{\mathbf{f}}$ using observation $\{\mathbf{x}^i_{O_i}\}_{i=1}^n$ and then impute missing values $\{\mathbf{x}^i_{\mathcal{M}_i}\}_{i=1}^n$ using $\hat{\Sigma}$, $\hat{\mathbf{f}}$ and observation $\{\mathbf{x}^i_{O_i}\}_{i=1}^n$. In this section we first show how to impute the missing values with given estimates $\hat{\mathbf{f}}$ and $\hat{\Sigma}$. The estimation for $\mathbf{f}$ and Σ appear in Section 2.4.

For the latent normal vector $\mathbf{z}^i$ satisfying $\mathbf{x}^i = \mathbf{f}(\mathbf{z}^i)$, $\mathbf{z}^i$ follows truncated normal distribution. In an observed continuous dimension j, z^i_j reduces to the point $f_j^{-1}(x^i_j)$. In an observed ordinal dimension j, z^i_j lies in the interval $f_j^{-1}(x^i_j)$. In an observed truncated dimension j, z^i_j reduces to the point $f_j^{-1}(x^i_j)$ if x^i_j is not the truncated value, otherwise is a truncated normal in the interval $f_j^{-1}(x^i_j)$. Thus the constrained region $\mathbf{f}^{-1}_{O_i}(\mathbf{x}^i_{O_i})$ is a Cartesian product of intervals (including trivial single point intervals for continuous dimensions). There is no constraint in missing dimension $\mathcal{M}_i$. It is natural to impute $\mathbf{x}^i_{\mathcal{M}_i}$ by mapping the conditional mean of $\mathbf{z}^i_{\mathcal{M}_i}$ through the marginals $\mathbf{f}_{\mathcal{M}_i}$, summarized in Definition 2 and Algorithm 1, visualized in Fig. 2.5.

Definition 2 (GC Imputation)**.** Suppose $\mathbf{x} \sim \mathrm{GC}(\Sigma, \mathbf{f})$ with observations $\mathbf{x}_O$ and missing entries $\mathbf{x}_{\mathcal{M}}$. We impute the missing entries as:

$$\hat{\mathbf{x}}_{\mathcal{M}} = \mathbf{f}_{\mathcal{M}}(\mathrm{E}[\mathbf{z}_{\mathcal{M}}|\mathbf{x}_O]) = \mathbf{f}_{\mathcal{M}}(\Sigma_{\mathcal{M},O}\Sigma_{O,O}^{-1}\mathrm{E}[\mathbf{z}_O|\mathbf{x}_O]), \tag{2.2}$$

where $\mathrm{E}[\mathbf{z}_O|\mathbf{x}_O]$ is the expectation of a normal vector $\mathbf{z}_O \sim \mathcal{N}(\mathbf{0}, \Sigma_{O,O})$ truncated into the region $\{\mathbf{z}_O : \mathbf{f}_O(\mathbf{z}_O) = \mathbf{x}_O\}$ or $\mathbf{f}_O^{-1}(\mathbf{x}_O)$ for short. The evaluation of $\mathrm{E}[\mathbf{z}_{\mathcal{M}}|\mathbf{x}_O]$

is derived in Eq. (2.8).

Algorithm 1 Single imputation via Gaussian Copula

```
Input: observation $\{\mathbf{x}^i_{O_i}\}_{i=1}^n$, parameters estimate $\hat{\mathbf{f}}^{-1}$ and $\hat{\Sigma}$. For $i = 1, \dots, n$,
    • Compute the constraint $\mathbf{z}^i_{O_i} \in \hat{\mathbf{f}}^{-1}_{O_i}(\mathbf{x}^i_{O_i})$.
    • Impute $\hat{\mathbf{z}}^i_{\mathcal{M}_i} = \mathrm{E}[\mathbf{z}^i_{\mathcal{M}_i} | \mathbf{z}^i_{O_i} \in \hat{\mathbf{f}}^{-1}_{O_i}(\mathbf{x}^i_{O_i}), \hat{\Sigma}]$.
    • Impute $\hat{\mathbf{x}}^i_{\mathcal{M}_i} = \hat{\mathbf{f}}_{\mathcal{M}_i}(\hat{\mathbf{z}}^i_{\mathcal{M}_i})$.
Output: $\hat{\mathbf{x}}^i_{\mathcal{M}_i}$ for $i \in [n]$.
```

While most applications require just a single imputation, multiple imputations are useful to describe the uncertainty due to imputation. Our method also supports multiple imputation: in step (2) of Algorithm 1, replace the conditional mean imputation with conditional sampling and then impute $\hat{\mathbf{x}}^i_{\mathcal{M}_i}$ for each sample. The conditional sampling consists of two steps: (1) sample the truncated normal $\mathbf{z}^i_{O_i}$ conditional on $\mathbf{x}^i_{O_i}$ and $\hat{\Sigma}$; (2) sample the normal $\mathbf{z}^i_{\mathcal{M}_i}$ conditional on $\mathbf{z}^i_{O_i}$ and $\hat{\Sigma}$. Efficient sampling methods have been proposed [72] for multivariate truncated normal distribution.

Algorithm 2 Multiple imputation via Gaussian Copula

```
Input: # of imputations $m$, observation $\{\mathbf{x}^i_{O_i}\}_{i=1}^n$, parameters estimate $\hat{\mathbf{f}}^{-1}$ and $\hat{\Sigma}$.
For $i = 1, \dots, n$,
    • Compute the constraint $\mathbf{z}^i_{O_i} \in \hat{\mathbf{f}}^{-1}_{O_i}(\mathbf{x}^i_{O_i})$.
    • Sample $\hat{\mathbf{z}}^{i,s}_{O_i} \overset{i.i.d.}{\sim} \mathcal{N}(\mathbf{0}, \hat{\Sigma}_{O_i,O_i})$ truncated to $\mathbf{f}^{-1}_{O_i}(\mathbf{x}^i_{O_i})$ for $s = 1, ..., m$.
    • Sample $\hat{\mathbf{z}}^{i,s}_{\mathcal{M}_i} \overset{i.i.d.}{\sim} \mathbf{z}^i_{\mathcal{M}_i} | \hat{\mathbf{z}}^{i,s}_{O_i}, \hat{\Sigma}$ for $s = 1, ..., m$.
    • Compute $\hat{\mathbf{x}}^{i,s}_{\mathcal{M}_i} = \hat{\mathbf{f}}_{\mathcal{M}_i}(\hat{\mathbf{z}}^{i,s}_{\mathcal{M}_i})$ for $s = 1, .., m$.
Output: $\{\hat{\mathbf{x}}^{i,s}_{\mathcal{M}_i} | s \in [m]\}$ for $i \in [n]$.
```

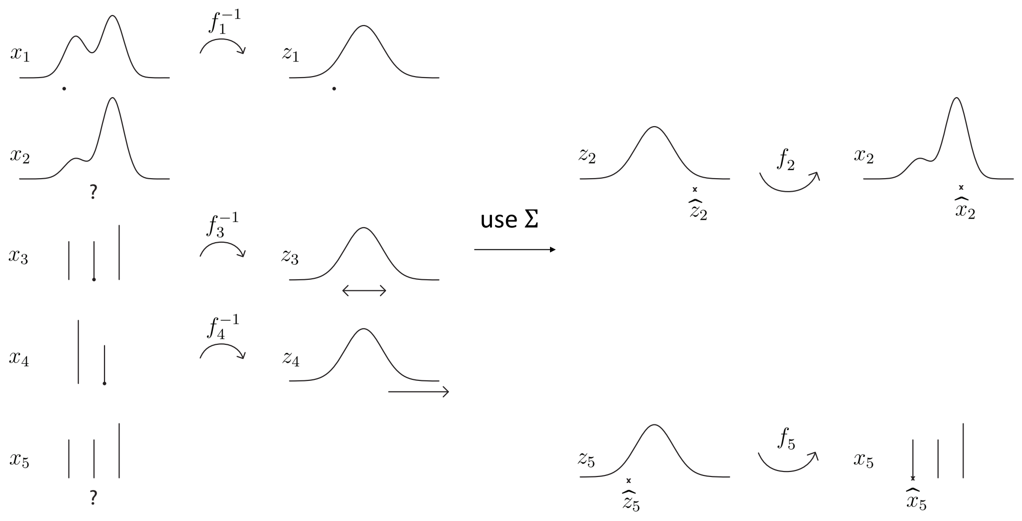


Figure 2.5: Gaussian copula imputation for a 5-dim partially observed mixed vector. Curves indicate the marginal probability density functions (for continuous) or probability mass function (for ordinal). First, compute the set of the latent normal vector which maps to the observation (x_1, x_3 and x_4) through $\mathbf{f}^{-1}$. Second, compute the conditional mean of the latent normal vector at missing locations ($\hat{z}_2$ and $\hat{z}_5$) given the copula correlation Σ and that z_1, z_3 and z_4 only take values from the computed inverse set. Lastly, map the conditional mean through $\mathbf{f}$ to obtain the imputations $\hat{x}_2$ and $\hat{x}_5$.

2.4 Parameter estimation

The model fitting algorithm has two steps. We first provide algorithms to estimate the marginal transformation function $\mathbf{f}$. Then we proceed to estaimte the copula correlation matrix Σ with a marginal estiamte fixed.

2.4.1 Marginal transformation estimation

To map between $\mathbf{x}$ and $\mathbf{z}$, we require both $\mathbf{f}^{-1}$ and $\mathbf{f}$. It is easier to directly estimate $\mathbf{f}^{-1}$. For $j \in C$, we have $f_j^{-1} = \Phi^{-1} \circ F_j$, as shown in Eq. (2.1). While the true CDF F_j is usually unavailable, it is natural to estimate it by the empirical CDF of $\mathbf{X}_j$

on the observed entries, denoted as $\hat{F}_j$. We use the following estimator:

$$\hat{f}_j^{-1}(x_j^i) = \Phi^{-1}\left(\frac{n}{n+1}\hat{F}_j(x_j^i)\right). \tag{2.3}$$

The scale constant $n/(n+1)$ ensures the output is finite. MCAR assumption guarantees the observed entries of $\mathbf{X}_j$ are from the distribution of F_j. Consider a case when MCAR is violated: an entry is observed if and only if it is smaller than a constant c, then the observed entries are actually from the distribution $\tilde{F}_j$:

$$\tilde{F}_j(x_j) = \begin{cases} F_j(x_j)/F_j(c), & \text{when } x \leq c \\ 1, & \text{when } x > c \end{cases}.$$

Thus we assume MCAR in this section. This assumption may be relaxed to MAR or even missing not at random by carefully modeling F_j or the missing mechanism. We leave that to our future work. Lemma 3 shows this estimator converges to f_j^{-1} in sup norm on the observed domain.

Lemma 3. Suppose the continuous random variable $x \in \mathbb{R}$ with CDF F_x and normal random variable $z \in \mathbb{R}$ satisfy $f(z)=x$ for a strictly monotone f. Given $x^1, \ldots, x^n \overset{i.i.d.}{\sim} F_x$, $m = \min_i x^i$, and $M = \max_i x^i$, the inverse $\hat{f}^{-1}$ defined in Eq. (2.3) satisfies

$$\mathrm{P}\left(\sup_{m \leq x \leq M} |\hat{f}^{-1}(x) - f^{-1}(x)| > \epsilon\right) \leq 2e^{-c_1 n\epsilon^2},$$

for any ϵ in $a_1 n^{-1} < \epsilon < b_1$, where $a_1, b_1, c_1 > 0$ are constants depending on $F_x(m)$ and $F_x(M)$.

For an ordinal variable j with k levels, $f_j(z_j) = \text{cutoff}(z_j; \mathbf{S}^j)$. Since $\mathbf{S}^j$ is determined by the probability mass function $\{p_l^j\}$ of x_j, we may estimate cutoffs $\hat{\mathbf{S}}^j$ as a special case of Eq. (2.3) by replacing p_l^j with its sample mean:

$$\mathbf{S}^j = \left\{ \Phi^{-1}\left(\frac{\sum_{i=1}^{n_j} \mathbb{1}(x_j^i \leq l)}{n_j + 1} \right),\ l \in [k-1] \right\}. \tag{2.4}$$

Lemma 4 shows that $\hat{\mathbf{S}}^j$ consistently estimates $\mathbf{S}^j$.

Lemma 4. Suppose the ordinal random variable $x \in [k]$ with probability mass function $\{p_l\}_{l=1}^k$ and normal random variable $z \in \mathbb{R}$ satisfy $f(z) = \text{cutoff}(z; \mathbf{S}) = x$. Given samples $x^1, \cdots, x^n \overset{i.i.d.}{\sim} \{p_l\}_{l=1}^k$, the cutoff estimate $\hat{\mathbf{S}}$ from Eq. (2.4) satisfies

$$\mathrm{P}\left(\|\hat{\mathbf{S}} - \mathbf{S}\|_1 > \epsilon \right) \leq 2^k e^{-c_2 n \epsilon^2/(k-1)^2},$$

for any ϵ in $(k-1)a_2 n^{-1} < \epsilon < (k-1)b_2$, where $a_2, b_2, c_2 > 0$ are constants depending on $\{p_1, p_k\}$.

For a truncated variable j, note f_j (see Table 2.1) is a strictly monotonic function with a step either on the left (lower truncated) or the right (upper truncated) or both (two sided truncated). Thus estimation of f_j^{-1} is a combination of Eq. (2.3) and Eq. (2.4).

2.4.2 Copula correlation estimation

We first consider maximum likelihood estimation (MLE) for Σ with complete continuous observation, then generalize the estimation method to incomplete mixed observation.

For a truncated variable $\mathbf{x}_j$, if a data point x_j^i is not the truncated value, then

z_j^i reduces to a point and it can be viewed as a continuous observation. If a data point x_j^i is the truncated value, then z_j^i is truncated normal and it can be viewed as an ordinal observation. Thus without loss of generality, we assume there are only ordinal variables and continuous variables. We use $\mathcal{C}, \mathcal{D} \subset [p]$ to denote the observed continuous and observed ordinal (discrete) dimensions, respectively. The observed dimensions are $\mathcal{O} = \mathcal{C} \cup \mathcal{D}$, so $\mathbf{x} = (\mathbf{x}_{\mathcal{C}}, \mathbf{x}_{\mathcal{D}}, \mathbf{x}_{\mathcal{M}}) = (\mathbf{x}_{\mathcal{O}}, \mathbf{x}_{\mathcal{M}})$.

Complete continuous observations

We begin by considering continuous, fully observed data. The density of the observed variable $\mathbf{x}$ is

$$p(\mathbf{x}; \Sigma, \mathbf{f})\ d\mathbf{x} = \phi(\mathbf{z}; \Sigma) d\mathbf{z},$$

where $\mathbf{z} = \mathbf{f}^{-1}(\mathbf{x}), d\mathbf{z} = \left|\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\right| d\mathbf{x}$, $\phi(\cdot; \Sigma)$ is the PDF of the normal vector with mean $\mathbf{0}$ and covariance Σ. The MLE of Σ maximizes the likelihood function defined as:

$$\ell(\Sigma; \mathbf{x}^i) = \frac{1}{n}\sum_{i=1}^{n} \log \phi(\mathbf{f}^{-1}(\mathbf{x}^i); \Sigma) = c - \frac{1}{2}\log\det\Sigma - \frac{1}{2}\mathrm{Tr}\left(\Sigma^{-1}\frac{1}{n}\sum_{i=1}^{n}\mathbf{z}^i(\mathbf{z}^i)^\top\right), \qquad (2.5)$$

over $\Sigma \in \mathcal{E}$, where $\mathbf{z}^i = \mathbf{f}^{-1}(\mathbf{x}^i)$ and c is a universal constant (We omit here and later the constant arising from $\left|\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\right|$ after the log transformation). Thus the MLE of Σ is the sample covariance of $\mathbf{Z} := \mathbf{f}(\mathbf{X}) = [f_1(\mathbf{X}_1), \ldots, f_p(\mathbf{X}_p)]$. When we substitute $\mathbf{f}$ by its empirical estimation in Eq. (2.3), the resulting covariance matrix $\tilde{\Sigma}$ of $\hat{\mathbf{Z}} := \hat{\mathbf{f}}(\mathbf{X})$ is still consistent and asymptotically normal under some regularity conditions [87], which justifies the use of our estimator $\hat{\mathbf{f}}$. To simplify notation, we assume $\mathbf{f}$ is known below.

For a Gaussian copula, notice Σ is a correlation matrix, thus we update $\hat{\Sigma} =$

$P_{\mathcal{E}}\tilde{\Sigma}$, where $P_{\mathcal{E}}$ scales its argument to output a correlation matrix. The obtained $\hat{\Sigma}$ is still consistent and asymptotically normal.

Incomplete mixed observations

When some columns are ordinal and some data is missing, the Gaussian latent vector $\mathbf{z}^i$ is no longer fully observed. We can compute the entries of $\mathbf{z}^i$ corresponding to continuous data: $\mathbf{z}^i_{C_i} = \mathbf{f}^{-1}_{C_i}(\mathbf{x}^i_{C_i})$. However, for ordinal data, $\mathbf{f}^{-1}_{\mathcal{D}_i}(\mathbf{x}^i_{\mathcal{D}_i})$ is a Cartesian product of intervals; we only know that $\mathbf{z}^i_{\mathcal{D}_i} \in \mathbf{f}^{-1}_{\mathcal{D}_i}(\mathbf{x}^i_{\mathcal{D}_i})$. The entries corresponding to missing observations, $\mathbf{z}^i_{\mathcal{M}_i}$, are entirely unconstrained. Hence the latent matrix $\hat{\mathbf{Z}}$ is only incompletely observed, and it is no longer possibly to simply compute its covariance.

We propose an expectation maximization (EM) algorithm to estimate Σ for incomplete mixed observation. Proceeding in an iterative fashion, we replace unknown $\mathbf{z}^i(\mathbf{z}^i)^\top$ with their expectation conditional on observations $\mathbf{x}^i_{O_i}$ and an estimate $\hat{\Sigma}$ in the E-step, then in the M-step we update the estimate of Σ as the conditional expectation of covaraince matrix:

$$G(\hat{\Sigma}, \mathbf{x}^i_{O_i}) = \frac{1}{n}\sum_{i=1}^{n} \mathrm{E}[\mathbf{z}^i(\mathbf{z}^i)^\top|\mathbf{x}^i_{O_i}, \hat{\Sigma}]. \tag{2.6}$$

Similar to the case of complete continuous data, we further scale the estimate to a correlation matrix. We first present the EM algorithm in Algorithm 3, then provide precise statements in Section 2.4.2. Computation details of Algorithm 3 appear in Section 2.4.2 and Section 2.4.3.

Algorithm 3 EM algorithm for Gaussian Copula

```
**Input:** observed entries $\{\mathbf{x}^i_{O_i}\}_{i=1}^n$.
**Initialize:** $t = 0, \Sigma^{(0)}$.
For $t = 0, 1, 2, \ldots$
    1. E-step: Compute $G^{(t)} = G(\Sigma^{(t)}, \mathbf{x}^i_{O_i})$.
    2. M-step: $\Sigma^{(t+1)} = G^{(t)}$.
    3. Scale to correlation matrix: $\Sigma^{(t+1)} = P_{\mathcal{E}}(\Sigma^{(t+1)})$
until convergence.
**Output:** $\hat{\Sigma} = \Sigma^{(t)}$.
```

EM algorithm

We first write down the marginal density of observed values by integrating out the missing data. Since $\mathbf{x}^i \sim \mathrm{GC}(\Sigma, \mathbf{f})$, there exist latent $\mathbf{z}^i$ satisfying $\mathbf{f}(\mathbf{z}^i) = \mathbf{x}^i$ and $\mathbf{z}^i \sim \mathcal{N}_p(\mathbf{0}, \Sigma)$. The likelihood of Σ given observation $\mathbf{x}^i_{O_i}$ is the integral over the latent Gaussian vector $\mathbf{z}^i_{O_i}$ that maps to $\mathbf{x}^i_{O_i}$ under the marginal $\mathbf{f}_{O_i}$. Hence the observed log likelihood we seek to maximize is:

$$\ell_{\mathrm{obs}}(\Sigma; \mathbf{x}^i_{O_i}) = \frac{1}{n} \sum_{i=1}^{n} \int_{\mathbf{z}^i_{O_i} \in \mathbf{f}^{-1}_{O_i}(\mathbf{x}^i_{O_i})} \phi(\mathbf{z}^i_{O_i}; \mathbf{0}, \Sigma_{O_i,O_i})\, d\mathbf{z}^i_{O_i}, \tag{2.7}$$

where Σ_{O_i,O_i} denote the submatrix of Σ with rows and columns in O_i. With known $\mathbf{f}$, MAR mechanism guarantees the maximizer of the likelihood in Eq. (7.8) shares the consistency and asymptotic normality of standard maximum likelihood estimate, according to the classical theory [59, Chapter 6.2].

However, the maximizer has no closed form expression. Even direct evaluation of $\ell_{\mathrm{obs}}(\Sigma; \mathbf{x}^i_{O_i})$ is challenging since it involves multivariate Gaussian integrals in a truncated region and the observed locations O_i varies for different observations i. Instead, the proposed EM algorithm is guaranteed to monotonically converge to a local maximizer according to classical EM theory [66, Chapter 3].

Now we derive the proposed EM algorithm in detail. Suppose we know the values of the unobserved $\mathbf{z}^i$. Then the joint likelihood function is the same as in Eq. (2.5). Since the values of $\mathbf{z}^i$ are unknown, we treat $\mathbf{z}^i$ as latent variables and $\mathbf{x}^i_{O_i}$ as observed variables. Substituting the joint likelihood function by its expected value given observations $\mathbf{x}^i_O$ and an estimate $\hat{\Sigma}$:

$$Q(\Sigma; \hat{\Sigma}, \mathbf{x}^i_{O_i}) := \frac{1}{n}\sum_{i=1}^{n} \mathrm{E}[\ell(\Sigma; \mathbf{x}^i_{O_i}, \mathbf{z}^i)|\mathbf{x}^i_{O_i}, \hat{\Sigma}] = c - \frac{1}{2}\left(\log\det(\Sigma) + \mathrm{Tr}\left(\Sigma^{-1} G(\hat{\Sigma}, \mathbf{x}^i_{O_i})\right)\right).$$

EM theory [66, Chapter 3] guarantees the updated $\tilde{\Sigma} = \mathrm{argmax}_{\Sigma\in\mathcal{E}}\, Q(\Sigma; \hat{\Sigma}, \mathbf{x}^i_{O_i})$ improves the likelihood with $\hat{\Sigma}$: $\ell_{\mathrm{obs}}(\tilde{\Sigma}; \mathbf{x}^i_{O_i}) \geq \ell_{\mathrm{obs}}(\hat{\Sigma}; \mathbf{x}^i_{O_i})$, and that by iterating this update, we produce a sequence $\{\Sigma^{(t)}\}$ that converges monotonically to a local maximizer of $\ell_{\mathrm{obs}}(\Sigma; \mathbf{x}^i_{O_i})$. At the t-th iteration, for the E step we compute $\mathrm{E}[\mathbf{z}^i(\mathbf{z}^i)^\top|\mathbf{x}^i_{O_i}, \Sigma^{(t)}]$ to express $Q(\Sigma; \Sigma^{(t)}, \mathbf{x}^i_{O_i})$ in terms of Σ. For the M step, we find $\Sigma^{(t+1)} = \mathrm{argmax}_{\Sigma}\, Q(\Sigma; \Sigma^{(t)}, \mathbf{x}^i_{O_i})$. In practice, we resort to an approximation, as in [39]. Notice that the unconstrained maximizer is $\tilde{\Sigma} = G(\Sigma^{(t)}, \mathbf{x}^i_{O_i})$. We update $\Sigma^{(t+1)} = P_{\mathcal{E}}\tilde{\Sigma}$.

Conditional expectation computation

Suppressing index i, we now show how to compute $\mathrm{E}[\mathbf{z}\mathbf{z}^\top|\mathbf{x}_O, \Sigma]$ in Eq. (2.4.3). With $\mathbf{z}_C = \mathbf{f}_C^{-1}(\mathbf{x}_C)$, it suffices to compute the following terms:

1. conditional mean and covariance of observed ordinal dimensions $\mathrm{E}[\mathbf{z}_{\mathcal{D}}|\mathbf{x}_O, \Sigma], \mathrm{Cov}[\mathbf{z}_{\mathcal{D}}|\mathbf{x}_O, \Sigma]$.
2. conditional mean and covariance of missing dimensions $\mathrm{E}[\mathbf{z}_{\mathcal{M}}|\mathbf{x}_O, \Sigma], \mathrm{Cov}[\mathbf{z}_{\mathcal{M}}|\mathbf{x}_O, \Sigma]$.

3. conditional covariance between missing and observed ordinal dimensions $\mathrm{Cov}[\mathbf{z}_{\mathcal{M}}, \mathbf{z}_{\mathcal{D}}|\mathbf{x}_{\mathcal{O}}, \Sigma]$.

We show that with the results from (1), we can compute (2) and (3). Computation for (1) is put in Sec 2.4.3.

Suppose we can know the ordinal values $\mathbf{z}_{\mathcal{D}}$ and thus $\mathbf{z}_{\mathcal{O}}$. Conditional on $\mathbf{z}_{\mathcal{O}}$, the missing dimensions $\mathbf{z}_{\mathcal{M}}$ follows normal distribution with mean $\mathrm{E}[\mathbf{z}_{\mathcal{M}}|\mathbf{z}_{\mathcal{O}}, \Sigma] = \Sigma_{\mathcal{M},\mathcal{O}}\Sigma_{\mathcal{O},\mathcal{O}}^{-1}\mathbf{z}_{\mathcal{O}}$. Further taking expectation of $\mathbf{z}_{\mathcal{O}}$ conditional on observation, we obtain

$$\mathrm{E}[\mathbf{z}_{\mathcal{M}}|\mathbf{x}_{\mathcal{O}}, \Sigma] = \mathrm{E}\left[\mathrm{E}[\mathbf{z}_{\mathcal{M}}|\mathbf{z}_{\mathcal{O}}, \Sigma]\middle|\mathbf{x}_{\mathcal{O}}, \Sigma\right] = \Sigma_{\mathcal{M},\mathcal{O}}\Sigma_{\mathcal{O},\mathcal{O}}^{-1}\mathrm{E}\left[\mathbf{z}_{\mathcal{O}}|\mathbf{x}_{\mathcal{O}}, \Sigma\right]. \tag{2.8}$$

One can compute $\mathrm{Cov}[\mathbf{z}_{\mathcal{M}}|\mathbf{x}_{\mathcal{O}}, \Sigma]$ and $\mathrm{Cov}[\mathbf{z}_{\mathcal{M}}, \mathbf{z}_{\mathcal{D}}|\mathbf{x}_{\mathcal{O}}, \Sigma]$ similarly: deferring details to the appendix, we find

$$\mathrm{Cov}[\mathbf{z}_{\mathcal{M}}, \mathbf{z}_{\mathcal{O}}|\mathbf{x}_{\mathcal{O}}, \Sigma] = \Sigma_{\mathcal{M},\mathcal{O}}\Sigma_{\mathcal{O},\mathcal{O}}^{-1}\mathrm{Cov}[\mathbf{z}_{\mathcal{O}}|\mathbf{x}_{\mathcal{O}}, \Sigma],$$

$$\mathrm{Cov}[\mathbf{z}_{\mathcal{M}}|\mathbf{x}_{\mathcal{O}}, \Sigma] = \Sigma_{\mathcal{M},\mathcal{M}} - \Sigma_{\mathcal{M},\mathcal{O}}\Sigma_{\mathcal{O},\mathcal{O}}^{-1}\Sigma_{\mathcal{O},\mathcal{M}} + \Sigma_{\mathcal{M},\mathcal{O}}\Sigma_{\mathcal{O},\mathcal{O}}^{-1}\mathrm{Cov}[\mathbf{z}_{\mathcal{O}}|\mathbf{x}_{\mathcal{O}}, \Sigma]\Sigma_{\mathcal{O},\mathcal{O}}^{-1}\Sigma_{\mathcal{O},\mathcal{M}}. \tag{2.9}$$

where $\mathrm{Cov}[\mathbf{z}_{\mathcal{O}}|\mathbf{x}_{\mathcal{O}}, \Sigma]$ has $\mathrm{Cov}[\mathbf{z}_{\mathcal{D}}|\mathbf{x}_{\mathcal{O}}, \Sigma]$ as its submatrix and 0 elsewhere, $\mathrm{Cov}[\mathbf{z}_{\mathcal{M}}, \mathbf{z}_{\mathcal{O}}|\mathbf{x}_{\mathcal{O}}, \Sigma]$ has $\mathrm{Cov}[\mathbf{z}_{\mathcal{M}}, \mathbf{z}_{\mathcal{D}}|\mathbf{x}_{\mathcal{O}}, \Sigma]$ as its submatrix and 0 elsewhere.

2.4.3 Approximating truncated normal mean and covariance

Now it remains to compute $\mathrm{E}[\mathbf{z}_{\mathcal{D}}|\mathbf{x}_{\mathcal{O}}, \Sigma]$ and $\mathrm{Cov}[\mathbf{z}_{\mathcal{D}}|\mathbf{x}_{\mathcal{O}}, \Sigma]$, which are the mean and covariance of a $|\mathcal{D}|$-dimensional normal truncated to $\mathbf{f}_{\mathcal{D}}^{-1}(\mathbf{x}_{\mathcal{D}})$, a Cartesian product of intervals. The computation involves multiple integrals of a nonlinear function and only admits a closed form expression when $|\mathcal{D}| = 1$. Direct computational methods [9] are very expensive and can be inaccurate even for moderate $|\mathcal{D}|$. Notice the computation needs to be done for each row $\mathbf{x}^i_{O_i}$ at each EM iteration separately, thus sampling truncated normal distribution to evaluate the empirical moments [72] is still expensive for large number of data points n. Instead, we use a fast iterative method that scales well to large datasets, following [39].

Suppose all but one element of $\mathbf{z}_{\mathcal{D}}$ is known. Then we can easily compute the resulting one dimensional truncated normal mean: for $j \in \mathcal{D}$, if $\mathbf{z}_j$ is unknown and $\mathbf{z}_{\mathcal{D}-j}$ is known, let $\mathrm{E}[z_j|\mathbf{z}_{\mathcal{D}-j}, \mathbf{x}_O, \Sigma] =: f_j(\mathbf{z}_{\mathcal{D}-j}; x_j, \Sigma)$ define the nonlinear function $f_j : \mathbb{R}^{|\mathcal{D}|-1} \rightarrow \mathbb{R}$, parameterized by x_j and Σ, detailed in the appendix. We may also use f_j to estimate $\mathrm{E}[z_j|\mathbf{x}_O, \Sigma]$ if $\mathrm{E}[\mathbf{z}_{\mathcal{D}-j}|\mathbf{x}_O, \Sigma]$ is known:

$$\begin{aligned}\mathrm{E}[z_j|\mathbf{x}_O, \Sigma] &= \mathrm{E}[\mathrm{E}[z_j|\mathbf{z}_{\mathcal{D}-j}, \mathbf{x}_O, \Sigma]|\mathbf{x}_O, \Sigma] \\ =&\mathrm{E}[f_j(\mathbf{z}_{\mathcal{D}-j}; x_j, \Sigma)|\mathbf{x}_O, \Sigma] \approx f_j(\mathrm{E}[\mathbf{z}_{\mathcal{D}-j}|\mathbf{x}_O, \Sigma]; x_j, \Sigma), \end{aligned} \tag{2.10}$$

if f_j is approximately linear. In other words, we can iteratively update the marginal mean of $\mathrm{E}[\mathbf{z}_{\mathcal{D}}|\mathbf{x}_O, \Sigma]$. At EM iteration $t + 1$, we conduct one iteration update with initial value from last EM iteration $\hat{\mathbf{z}}^{(t)}_{\mathcal{D}} \approx \mathrm{E}[\mathbf{z}_{\mathcal{D}}|\mathbf{x}_O, \Sigma^{(t)}]$:

$$\mathrm{E}[z_j|\mathbf{x}_O, \Sigma^{(t+1)}] \approx \hat{z}^{(t+1)}_j := f_j(\hat{\mathbf{z}}^{(t)}_{\mathcal{D}-j}; x_j, \Sigma^{(t+1)}). \tag{2.11}$$

Surprisingly, one iteration update works well and more iterations do not bring significant improvement.

We use a diagonal approximation for $\mathrm{Cov}\left[\mathbf{z}_{\mathcal{D}}|\mathbf{x}_{O}, \Sigma\right]$: we approximate $\mathrm{Cov}\left[z_j, z_k|\mathbf{x}_{O}, \Sigma\right]$ as 0 for $j \neq k \in \mathcal{D}$. This approximation performs well when z_j and z_k are nearly independent given all observed information. We approximate the diagonal entries $\mathrm{Var}\left[z_j|\mathbf{x}_{O}, \Sigma^{(t+1)}\right]$ for $j \in \mathcal{D}$ using a recursion similar to Eq. (2.11), detailed in the appendix.

We point out the estimated covariance matrix in Eq. (2.4.3) is the sum of the sample covariance matrix of the imputed $\mathbf{z}^i$ using its conditional mean and the expected covariance brought by the imputation. The diagonal approximation only applies to the second term, while the first term is dense. Consequently, the estimator in Eq. (2.4.3) is dense and can fit a large range of covariance matrices. Our experiments indicates that our approximation even outperforms the MCMC algorithm without such diagonal approximation [45].

Computation Cost The complexity of each EM iteration is $O(\alpha n p^3)$ with observed entry ratio α. The overall complexity is $O(T\alpha n p^3)$, where T is the number of EM steps required for convergence. We found $T \leq 50$ in most of our experiments. On a laptop with Intel-i5-3.1GHz Core and 8 GB RAM, it takes 1.2min for our algorithm to converge on a dataset with size 2000×60 and 25% missing entries (generated as in Section 2.5.1 when $p = 60$). Scaling our algorithm to large p is important future work. However, our algorithm is usually faster than many start-of-the-art imputation algorithms for large n small p. Speed comparison on a dataset with size 6039×207 is shown in Section 2.5.3.

Parallelization Noting the computation of expectation in is separable over the rows, we have developed a parallel algorithm to accelerate the EM algorithms.

2.5 Experiments

Our first experiment demonstrates that our method, `Copula-EM`, is able to estimate a well-specified Gaussian copula model faster than the MCMC method `sbgcop` [45, 44]. Our other experiments compare the accuracy of imputations produced by `Copula-EM` with `missForest` [84], `xPCA` [3] and `imputeFAMD` [4], state-of-the-art nonparametric imputation algorithms for mixed data; and the low rank matrix completion algorithms `softImpute` [64] and `GLRM` [91], which scale to large datasets. `missForest` is implemented with recommended default settings: 10 maximum iterations and 100 trees [83]. All other methods require selecting either the rank or the penalization parameter. We select them through 5-fold cross validation (5CV), unless otherwise specified. See the appendix for implementation details. For real datasets, we report results from our `Copula-EM` but put that from `sbgcop` in the appendix, since `Copula-EM` outperforms on all evaluation metrics and converges substantially faster.

2.5.1 Synthetic data

The first experiment compares the speed of the two algorithms to estimate Gaussian copula models: `Copula-EM` and `sbgcop`. Note `Copula-EM` is implemented in pure R, while the computational core of `sbgcop` is implemented in C. Hence further acceleration of `Copula-EM` is possible.

We generate 100 synthetic datasets with $n = 2000$ observations and $p = 15$ variables from a well-specified Gaussian copula model with random Σ generated [73]. For each Σ, first generate rows of $\mathbf{Z} \in \mathbb{R}^{n\times p}$ as $\mathbf{z}^1, \cdots, \mathbf{z}^n \overset{i.i.d.}{\sim} \mathcal{N}(\mathbf{0}, \Sigma)$.

Then generate $\mathbf{X} = \mathbf{f}(\mathbf{Z})$ using monotone $\mathbf{f}$ such that $\mathbf{X}_1, \ldots, \mathbf{X}_5$ have exponential distributions, $\mathbf{X}_6, \ldots, \mathbf{X}_{10}$ are binary and $\mathbf{X}_{11}, \ldots, \mathbf{X}_{15}$ are 1-5 ordinal.

We randomly remove 30% of the entries of $\mathbf{X}$, train `Copula-EM` and `sbgcop`, and compute the imputation error on the held-out set. We plot the imputation accuracy and correlation estimation accuracy versus runtime of each algorithm in Figure 2.6. `Copula-EM` converges quickly, in about 25s, while `sbgcop` takes much longer and suffers high error at shorter times. `Copula-EM` estimates correlations and continuous imputations at convergence more accurately than `sbgcop` even when the latter algorithm is given 6 times more runtime. Interestingly, `Copula-EM` recovers the correlation matrix better than `sbgcop` even asymptotically. These results demonstrate the impact of the approximate EM algorithm 2.4.3 compared to the (fully accurate) MCMC model of `sbgcop`: the approximation allows faster convergence, to an estimate of nearly the same quality.

For ordinal data imputation, `Copula-EM` reaches the same performance as `sbgcop` 6 times faster. For binary data imputation, `sbgcop` is four times slower than `Copula-EM` at reaching the final performance of `Copula-EM`, but `sbgcop` outperforms `Copula-EM` given even more time. We conjecture that the drop in imputation accuracy of `Copula-EM` for binary data could be mitigated using multiple imputation [59, Chapter 5.4], as outlined in Section 2.3 by combining the imputations (using mean or median) into a single imputation to reduce the effect of approximating the truncated normal distribution.

The second experiment compares the imputation accuracy of `Copula-EM` and nonparametric algorithms. Using the same data generation mechanism, we randomly remove $10\% - 50\%$ of the entries of $\mathbf{X}$. The optimal rank selected using

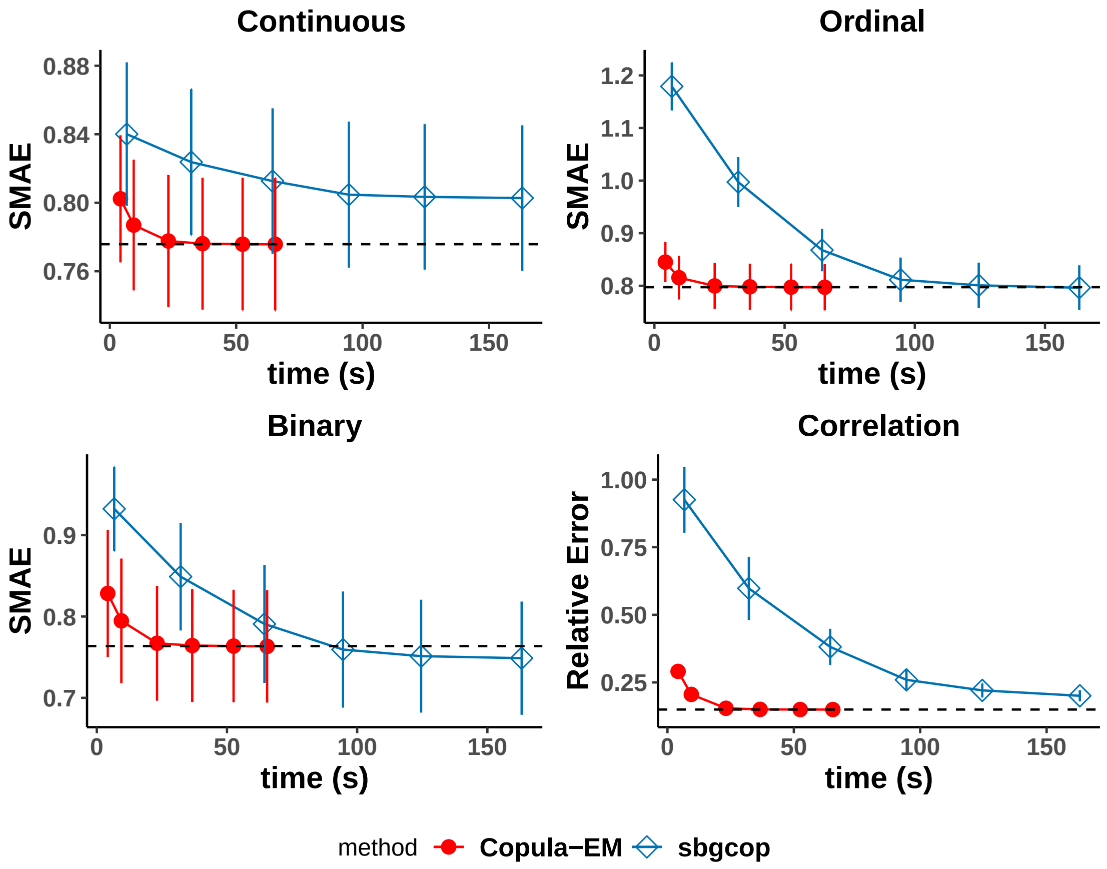


Figure 2.6: `Copula-EM` vs `sbgcop`: The imputation error for each data type and estimated correlation error over time cost. Dashed line indicates the final error of `Copula-EM`.

5CV is 3 for `xPCA` and 6 for `imputeFAMD`. Shown in Figure 2.7,`Copula-EM` substantially outperforms all nonparametric algorithms for all data types.

2.5.2 General Social Survey (GSS) data

We chose 18 variables with 2538 observations from GSS dataset in year 2014. 24.9% of the entries are missing. The dataset consists of 1 continuous (`AGE`) and 17 ordinal variables with 2 to 48 levels. We investigate the imputation accuracy

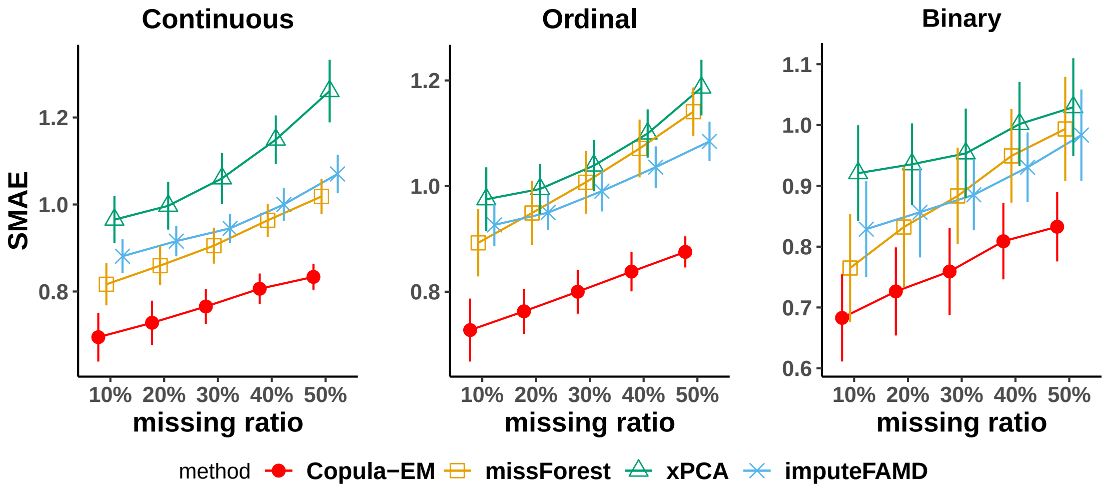


Figure 2.7: `Copula-EM` vs nonparametric algorithms: The imputation error for each data type on synthetic data.

Table 2.2: Imputation Error on Five GSS Variables

Variable	Copula-EM	missForest	xPCA	imputeFAMD
CLASS	**0.735(0.10)**	0.782(0.09)	0.795(0.08)	0.797(0.10)
LIFE	**0.759(0.12)**	0.828(0.17)	0.783(0.11)	0.821(0.11)
HEALTH	**0.877(0.09)**	1.143(0.18)	0.908(0.10)	0.947(0.04)
HAPPY	**0.896(0.08)**	1.079(0.15)	1.003(0.15)	1.001(0.10)
INCOME	**0.869(0.07)**	0.944(0.18)	1.090(0.15)	0.996(0.01)

on five selected variables: `INCOME`, `LIFE`, `HEALTH`, `CLASS`[1] and `HAPPY`. For each variable, we sample 1500 observation and divide them into 20 folds. We mask one fold of only one variable as test data in each experiment. The selected rank is 2 for both `xPCA` and `imputeFAMD`. We report the SMAE for each variable in Table 2.2. Our method performs the best for all variables. Further our method always performs better than median imputation. In contrast, the other three methods perform worse than median imputation for some variables. Our method also provides estimated variable correlation, which is usually desired in social survey study. We plot high correlations from the copula correlation matrix as a graph in Figure 2.8.

[1]Subjective class identification from lower to upper class

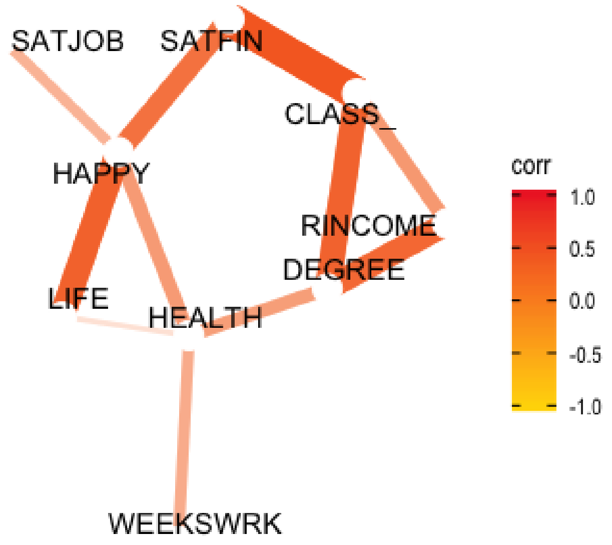


Figure 2.8: High Correlations ($|\cdot| > 0.3$) of 5 interesting variables from GSS data are plotted.

2.5.3 MovieLens 1M data

Recall our method scales cubicly in the number of variables. Hence for this experiment, we sample the subset of the MovieLens 1M data [41] consisting of the 207 movies with at least 1000 ratings and all users who rate at least one of those 207 movies. On this subset, 75.6% of entries are missing. Under the time limit 1 hour, we implement all algorithms but `imputeFAMD`. `Copula-EM` takes 9 mins and `missForest` takes 25 mins. These two methods have no parameters to tune. To select tuning parameters for other algorithms, we manually mask 10% of the data for the test set and use the remaining data to train the model, and repeat 20 times. The selected rank using 5CV is 99 for `softImpute`, 6 for `xPCA`, and 8 for `GLRM` with bigger-vs-smaller loss. With the selected tuning parameter, low rank matrix completion methods are substantially faster. For example, `softImpute` only takes 33s. However, counting the additional time to select tuning parameters using 5CV, `softImpute` takes 16mins to select the penaliza-

Table 2.3: Imputation Error on 207 Movies

Algorithm	MAE	RMSE
Column Median	0.702(0.004)	1.001(0.004)
Copula-EM	**0.579(0.004)**	**0.880(0.005)**
GLRM	0.595(0.004)	0.892(0.004)
softImpute	0.602(0.004)	0.883(0.004)
xPCA	0.613(0.004)	0.897(0.004)
missForest	0.669(0.004)	1.015(0.006)

tion parameter with regularization path length 50, which is already more expensive than `Copula-EM`. Interestingly, the ranks selected are quite different even when the models perform similarly: `GLRM` chooses rank 8 while `softImpute` chooses rank 99.

We report both mean absolute error (MAE) and RMSE in Table 2.3. Our method outperforms all others in both MAE and RMSE. This result is notable, because `Copula-EM` does not directly minimize MAE or RMSE, while `softImpute` directly minimizes RMSE. It also indicates `Copula-EM` does not overfit even with $O(p^2)$ free parameters.

2.5.4 Music Auto-tagging: CAL500exp data

The CAL500 expansion (CAL500exp) dataset [96] is an enriched version of the well-known CAL500 dataset [88]. This dataset consists of 67 binary tags (including genre, mood and instrument, labeled by experts) to 3223 music fragments from 500 songs. Music auto-tagging is a multi-label learning problem. A feature vector is usually computed first based on the music files and then a classifier is trained for each tag. This procedure is expensive and neglects the association among known labels. We treat this task as a missing data imputation problem and only use observed labels to impute unknown labels. This dataset is com-

Table 2.4: Imputation Error (SMAE) on CAL500exp.

Algorithm	40% missing	50% missing	60% missing
Copula-EM	**0.799(0.002)**	**0.822(0.003)**	**0.849(0.002)**
missForest	0.800(0.018)	0.984(0.026)	1.181(0.024)
imputeFAMD	0.823(0.013)	0.920(0.016)	1.114(0.020)
xPCA	0.911(0.018)	0.988(0.071)	1.108(0.145)

Table 2.5: Imputation Error on More Ordinal and Mixed Datasets.

Dataset	Size	Selected Rank
ESL	488×5, 4 features, 1 label	1 (xPCA), 5 (imputeFAMD)
LEV	1000×5, 4 features, 1 label	1 (xPCA), 5 (imputeFAMD)
GBSG	686×10, 6 continuous, 4 ordinal	2 (xPCA), 2 (imputeFAMD)
TIPS	244×7, 2 continuous, 5 ordinal	2 (xPCA), 6 (imputeFAMD)

pletely observed. We randomly remove some portions of the observed labels as a test set and repeat 20 times. The selected optimal rank is 4 for `xPCA` and 15 for `imputeFAMD`. Shown in Table 2.4, `Copula-EM` performs the best in terms of SMAE. The superiority of `Copula-EM` over other algorithms substantially grows as the missing ratio increases. Moreover, `Copula-EM` yields very stable imputations: the standard deviation of its SMAE is imperceptibly small.

2.5.5 More ordinal data and dixed data

We compare mixed data imputation algorithms on two more ordinal classification datasets[2], Lecturers Evaluation *(LEV)* and Employee Selection *(ESL)*, and two more mixed datasets, German Breast Cancer Study Group *(GBSG)*[3] and Restaurant Tips *(TIPS)*[4]. Dataset descriptions appear in Table 2.5, and more details appear in the appendix. All datasets are completely observed.

[2]Available at https://waikato.github.io/weka-wiki/datasets/
[3]Available at https://cran.r-project.org/web/packages/mfp/
[4]Available at http://ggobi.org/book/

Table 2.6: Imputation Error on More Ordinal and Mixed Datasets.

Dataset	Type	Copula-EM	missForest	xPCA	imputeFAMD
ESL	Label	**0.372(0.04)**	0.553(0.08)	0.404(0.04)	0.503(0.06)
	Feature	**0.584(0.03)**	0.873(0.06)	0.668(0.03)	0.687(0.03)
LEV	Label	**0.750(0.04)**	0.970(0.09)	0.860(0.06)	0.882(0.05)
	Feature	0.907(0.01)	**0.799(0.03)**	1.037(0.02)	1.085(0.04)
GBSG	Ordinal	**0.793(0.03)**	0.887(0.05)	0.876(0.04)	0.840(0.03)
	Continuous	**0.876(0.01)**	1.029(0.03)	1.100(0.04)	1.038(0.03)
TIPS	Ordinal	**0.786(0.05)**	0.928(0.09)	0.928(0.08)	0.891(0.09)
	Continuous	**0.755(0.04)**	0.837(0.05)	1.011(0.11)	0.892(0.13)

For each dataset, we randomly remove 30% entries as a test set and repeat 100 times. For ordinal classification datasets, we evaluate the SMAE for the label and for the features, respectively. For mixed datasets, we evaluate the SMAE for ordinal dimensions and for continuous dimensions, respectively. We report results in Table 2.6. Our method outperforms the others in all but one setting, often by a substantial margin.

2.6 Discussion

We end by noting a few contrasts between the present approach and typical low rank approximation methods for data imputation. Low rank approximation constructs a latent simple (low rank) object and posits that observations are noisy draws from that simple latent object. In contrast, our approach uses a parametric, but full-dimensional, model for the latent object; observations are given by a deterministic function of the latent object. In other words, in previous work the latent object is exact and the observations are noisy; in our work, the latent object is noisy and the observations are exact. Which more faithfully models real data? As evidence, we might consider whether low rank models agree on the best rank to fit a given dataset. For example, on the MovieLens dataset: (1) The low rank matrix completion methods `xPCA` and `GLRM`, implemented using alternating minimization, select small optimal ranks (6 and 8), while `softImpute`, implemented using nuclear norm minimization, selects the much larger optimal rank 99. (2) Our algorithm outperforms all the low rank matrix completion methods we tested. These observations suggest the low rank assumption commonly used to fit the MovieLens dataset may not be fundamental, but may arise as a mathematical artifact [90]. More supporting empirical results can be found in [6]: the performance of `softImpute` keeps improving as the rank increases (up to 10^3).

CHAPTER 3

IMPUTATION VIA LOW RANK GAUSSIAN COPULA

This chapter develops the low rank Gaussian copula model for high dimensional missing data imputation. This chapter is based on [102]. We introduce our model in Section 3.2, the estimation algorithm in Section 3.3, and analyze the MSE of our imputation estimator in Section 3.4.

3.1 Introduction

We have demonstrated in Section 2.5 our proposed Gaussian copula imputation Section 2.3 enjoys state-of-the-art performance on long skinny datasets. However, the proposed algorithm scales cubically in the number of columns, which is too expensive for applications to large-scale datasets such as collaborative filtering and medical informatics.

Our contribution

We propose a low rank Gaussian copula (LRGC) model for imputation with quantified uncertainty. The proposed model combines the advantages of probabilistic principal component analysis (PPCA) and Gaussian copula: the probabilistic description of missing entries allows for uncertainty quantification; the low rank structure allows for efficient estimation from large-scale data; and the copula framework provides the generality to accurately fit real-world data. The imputation proceeds in two steps: first we fit the LRGC model, and then we compute the distribution of the missing values separately for each row, conditional on the observed values in that row. We impute the missing values with the conditional mean and quantify their uncertainty with the conditional variance. Our contributions are as follows.

1. We propose a probabilistic imputation method based on the low rank Gaussian copula model to impute real-valued, ordinal and Boolean data. The rank of the model is the only tuning parameter.
2. We propose an algorithm to fit the proposed model that scales linearly in the number of rows and the number of columns. Empirical results show our imputations provide state-of-the-art accuracy across a wide range of data types, including those with high rank.
3. We characterize how the mean squared error (MSE) of our imputations depends on the SNR. In particular, we show the MSE converges exponentially to the noise level in the limit of high SNR.

Inheriting the advantages of the Gaussian copula model, LRGC naturally handles mixed data and has no model hyperparameters except for a rank.

Related work

Although our proposed model has a low rank structure, it greatly differs from LRMC in that the observations are assumed to be generated from, but not equal to, a real-valued low rank matrix. Many authors have considered generalizations of LRMC beyond real-valued low rank observations: to Boolean data [26], ordinal data [56, 10, 3], mixed data [91, 80], data from an exponential family distribution [38], and high rank matrices [36, 70, 31, 32]. However, none of these methods can quantify the uncertainty of the resulting imputations.

Researchers from a Bayesian tradition have also studied the LRGC model with missing data [68, 25]. However, the associated MCMC algorithms are expensive and do not scale to large-scale data.

3.2 Low rank Gaussian copula model

We propose a low rank Gaussian copula (LRGC) model that integrates the flexible marginals of the Gaussian copula model with the low rank structure of the PPCA model [86]. To define the model, first consider a p-dimensional Gaussian vector $\mathbf{z} \sim \mathrm{PPCA}(\mathbf{W}, \sigma^2)$ generated from the PPCA model:

$$\mathbf{z} = \mathbf{W}\mathbf{t} + \boldsymbol{\epsilon}, \text{ where } \mathbf{t} \sim \mathcal{N}_k(\mathbf{0}, \mathbf{I}_k), \boldsymbol{\epsilon} \sim \mathcal{N}_p(\mathbf{0}, \sigma^2\mathbf{I}_p), \mathbf{t} \text{ and } \boldsymbol{\epsilon} \text{ are independent,} \quad (3.1)$$

where $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_p]^\top \in \mathbb{R}^{p\times k}$ with $p > k$. We say $\mathbf{x}$ follows the *low rank Gaussian copula (LRGC) model* if $\mathbf{x} \sim \mathrm{GC}(\mathbf{W}\mathbf{W}^\top + \sigma^2\mathbf{I}_p, \mathbf{f})$ and $\mathbf{f}(\mathbf{z}) = \mathbf{x}$ for $\mathbf{z} \sim \mathrm{PPCA}(\mathbf{W}, \sigma^2)$.

To ensure that $\mathbf{x}$ follows the Gaussian copula, $\mathbf{z}$ must have zero mean and unit variance in all dimensions. Hence we require the covariance $\mathbf{W}\mathbf{W}^\top + \sigma^2\mathbf{I}_p$ to have unit diagonal: $\|\mathbf{w}_j\|_2^2 + \sigma^2 = 1$ for $j \in [p]$. We summarize the LRGC model in the following definition.

Definition 3. We say a random vector $\mathbf{x} \in \mathbb{R}^p$ follows the low rank Gaussian copula $\mathbf{x} \sim \mathrm{LRGC}(\mathbf{W}, \sigma^2, \mathbf{f})$ with parameters $\mathbf{W} \in \mathbb{R}^{p\times k}(p > k), \sigma^2$ and $\mathbf{f}$ if (1) $\mathbf{f}$

is an elementwise monotonic function; (2) $\mathbf{W}\mathbf{W}^\top + \sigma^2\mathbf{I}_p$ has unit diagonal; (3) $\mathbf{f}(\mathbf{z}) = \mathbf{x}$ for $\mathbf{z} \sim \text{PPCA}(\mathbf{W}, \sigma^2)$.

To see the generality of the LRGC model, suppose $\mathbf{X}$ has iid rows $\mathbf{x}^i \sim \text{LRGC}(\mathbf{W}, \sigma^2, \mathbf{f})$. Then

$$\begin{aligned}\mathbf{X} = \mathbf{f}(\mathbf{Z}) = \mathbf{f}(\mathbf{T}\mathbf{W}^\top + \mathbf{E}) &:= [f_1(\mathbf{Z}_1), \ldots, f_p(\mathbf{Z}_p)] \\ &= [f_1(\mathbf{T}\mathbf{w}_1 + \mathbf{E}_1), \ldots, f_p(\mathbf{T}\mathbf{w}_p + \mathbf{E}_p)] \end{aligned} \tag{3.2}$$

where $\mathbf{Z}, \mathbf{T}, \mathbf{E}$ have rows $\mathbf{z}^i, \mathbf{t}^i, \boldsymbol{\epsilon}^i$, respectively, satisfying $\mathbf{z}^i = \mathbf{W}\mathbf{t}^i + \boldsymbol{\epsilon}^i$ and $\mathbf{f}(\mathbf{z}^i) = \mathbf{x}^i$ for $i \in [n]$. While the latent normal matrix $\mathbf{Z}$ has low rank plus noise structure, the observation matrix $\mathbf{X}$ can have high rank or ordinal entries with an appropriate choice of the marginals $\mathbf{f}$. When all marginals of $\mathbf{f}$ are linear functions in $\mathbb{R}$, the LRGC model reduces to the PPCA model.

Our method differs from LRMC and multiple imputation (MI) [51, 67] in that we treat one factor $\mathbf{W}$ as model parameters, but the other factor $\mathbf{T}$ as unseen random samples. With estimated $\mathbf{W}$, we analytically integrate over all $\mathbf{T}$ to obtain the imputation and quantify uncertainty. In contrast, LRMC and its generalization aim to estimate both factors $\mathbf{W}$ and $\mathbf{T}$ as model parameters, which make it hard to quantify uncertainty. MI treats both factors $\mathbf{W}$ and $\mathbf{T}$ as unseen random samples, which make the computation, such as the posterior distribution, intractable and requires expensive sampling on large datasets.

Imputation The imputation under LRGC is a special case of imputation under general Gaussian copula model in Definition 2 and Algorithm 1. We provide the specific forms of $\mathrm{E}[\mathbf{z}_{\mathcal{M}}|\mathbf{x}_{\mathcal{O}}]$ and $\mathrm{Cov}[\mathbf{z}_{\mathcal{M}}|\mathbf{x}_{\mathcal{O}}]$, as stated in Lemma 5.

Lemma 5. Suppose $\mathbf{x} \sim \mathrm{LRGC}(\mathbf{W}, \sigma^2, \mathbf{f})$ with observations $\mathbf{x}_{\mathcal{O}}$ and missing entries $\mathbf{x}_{\mathcal{M}}$. Then for the latent normal vector $\mathbf{z}$ satisfying $\mathbf{f}(\mathbf{z}) = \mathbf{x}$, with corresponding latent subvectors $\mathbf{z}_{\mathcal{O}}$ and $\mathbf{z}_{\mathcal{M}}$,

$$\mathrm{E}[\mathbf{z}_{\mathcal{M}}|\mathbf{x}_{\mathcal{O}}] = \mathbf{W}_{\mathcal{M}}\mathbf{M}_{\mathcal{O}}^{-1}\mathbf{W}_{\mathcal{O}}^{\top}\mathrm{E}[\mathbf{z}_{\mathcal{O}}|\mathbf{x}_{\mathcal{O}}], \text{ where } \mathbf{M}_{\mathcal{O}} = \sigma^2\mathbf{I}_k + \mathbf{W}_{\mathcal{O}}^{\top}\mathbf{W}_{\mathcal{O}} \tag{3.3}$$

$$\mathrm{Cov}[\mathbf{z}_{\mathcal{M}}|\mathbf{x}_{\mathcal{O}}] = \sigma^2\mathbf{I}_{|\mathcal{M}|} + \sigma^2\mathbf{W}_{\mathcal{M}}\mathbf{M}_{\mathcal{O}}^{-1}\mathbf{W}_{\mathcal{M}}^{\top} + \mathbf{W}_{\mathcal{M}}\mathbf{M}_{\mathcal{O}}^{-1}\mathbf{W}_{\mathcal{O}}^{\top}\mathrm{Cov}[\mathbf{z}_{\mathcal{O}}|\mathbf{x}_{\mathcal{O}}]\mathbf{W}_{\mathcal{O}}\mathbf{M}_{\mathcal{O}}^{-1}\mathbf{W}_{\mathcal{M}}^{\top} \tag{3.4}$$

3.3 Parameter estimation

Suppose $\mathbf{X} \in \mathbb{R}^{n\times p}$ observed on Ω has iid rows $\mathbf{x}^i \sim \mathrm{LRGC}(\mathbf{W}, \sigma^2, \mathbf{f})$ and $\mathbf{x}^i$ has observations $\mathbf{x}^i_{\mathcal{O}_i}$ and missing entries $\mathbf{x}^i_{\mathcal{M}_i}$. We estimate the marginals $\mathbf{f}$ and copula correlation parameters $(\mathbf{W}, \sigma^2)$. The estimation of $\mathbf{f}$ is the same as that under the general Gaussian copula, see Section 2.4.1. To estimate $(\mathbf{W}, \sigma^2)$, we propose an EM algorithm that scales linearly in n and p, using ingredients from [39] for the E-step, and from [48] for the M-step. We present essentials here and summarize in Algorithm 4.

3.3.1 EM algorithm for W and σ^2

Ideally, we would compute the maximum likelihood estimates (MLE) for the copula parameters $(\mathbf{W}, \sigma^2)$ (under the likelihood in Eq. (3.5)), which are consistent under the MAR mechanism [59, Chapter 6.2] as $n \to \infty$. However, the likelihood involves a Gaussian integral that is hard to optimize. Instead, we estimate the MLE using an approximate EM algorithm.

The likelihood of $(\mathbf{W}, \sigma^2)$ given observation $\mathbf{x}^i_{O_i}$ is the integral over the latent Gaussian vector $\mathbf{z}^i_{O_i}$ that maps to $\mathbf{x}^i_{O_i}$ under the marginal $\mathbf{f}_{O_i}$. Hence the observed log likelihood we seek to maximize is:

$$\ell_{\text{obs}}(\mathbf{W}, \sigma^2; \{\mathbf{x}^i_{O_i}\}_{i=1}^n) = \sum_{i=1}^n \log \int_{\mathbf{z}^i_{O_i} \in \mathbf{f}^{-1}_{O_i}(\mathbf{x}^i_{O_i})} \phi(\mathbf{z}^i_{O_i}; \mathbf{0}, \mathbf{W}_{O_i}\mathbf{W}^\top_{O_i} + \sigma^2 \mathbf{I}_{|O_i|}) d\mathbf{z}^i_{O_i}, \tag{3.5}$$

where $\phi(\cdot; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ denotes the Gaussian vector density with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$. Recall the decomposition $\mathbf{Z} = \mathbf{T}\mathbf{W}^\top + \mathbf{E}$ as in Eq. (3.2). If $\mathbf{z}^i_{O_i}$ and $\mathbf{t}^i$ are known, the joint likelihood is simple:

$$\ell(\mathbf{W}, \sigma^2; \{\mathbf{x}^i_{O_i}, \mathbf{z}^i_{O_i}, \mathbf{t}^i\}_{i=1}^n) = \sum_{i=1}^n \log \left[\phi(\mathbf{z}^i_{O_i}; \mathbf{W}_{O_i}\mathbf{t}^i, \sigma^2 \mathbf{I}_p)\, \phi(\mathbf{t}^i; \mathbf{0}, \mathbf{I}_k)\, \mathbb{1}_{\mathbf{f}^{-1}_{O_i}(\mathbf{x}^i_{O_i})}(\mathbf{z}^i_{O_i}) \right]. \tag{3.6}$$

Here define $\mathbb{1}_A(x) = 1$ when $x \in A$ and 0 otherwise. The maximizers $(\hat{\mathbf{W}}, \hat{\sigma})$ of Eq. (3.6) are $\hat{\mathbf{W}} = \text{argmin}_{\mathbf{W}} \|P_\Omega(\mathbf{Z} - \mathbf{T}\mathbf{W}^\top)\|_F^2$ and $\hat{\sigma}^2 = \|P_\Omega(\mathbf{Z} - \mathbf{T}\hat{\mathbf{W}}^\top)\|_F^2 / |\Omega|$. Moreover, the problem is separable over the rows of $\hat{\mathbf{W}}$: to solve for the j-th row $\hat{\mathbf{w}}_j^\top$, we use only $\mathbf{z}^i_{O_i}, \mathbf{t}^i$ for $i \in \Omega_j = \{i : (i, j) \in \Omega\}$. Our EM algorithm treats the unknown $\mathbf{z}^i_{O_i}, \mathbf{t}^i$ as latent variables and $\mathbf{x}^i_{O_i}$ as the observed variable. Given an estimate $(\tilde{\mathbf{W}}, \tilde{\sigma}^2)$, the E-step computes the expectation $\mathbb{E}[\|P_\Omega(\mathbf{Z} - \mathbf{T}\mathbf{W}^\top)\|_F^2]$ with respect to $\mathbf{z}^i_{O_i}$ and $\mathbf{t}^i$ conditional on $\mathbf{x}^i_{O_i}$. Throughout the section, we use $\mathbb{E}$ to denote this conditional expectation. The M-step is similar to when $\mathbf{z}^i_{O_i}$ and $\mathbf{t}^i$ are known.

E step Calculate the expected likelihood $Q(\mathbf{W}, \sigma^2; \tilde{\mathbf{W}}, \tilde{\sigma}^2)$:

$$Q(\mathbf{W}, \sigma^2; \tilde{\mathbf{W}}, \tilde{\sigma}^2) = \mathbb{E}[\ell(\mathbf{W}, \sigma^2; \{\mathbf{x}^i_{O_i}, \mathbf{z}^i_{O_i}, \mathbf{t}^i\}_{i=1}^n)] = c - \frac{\sum_{i=1}^n |O_i| \log(\sigma^2)}{2} - \frac{\sum_{i=1}^n \left(\mathbb{E}[(\mathbf{z}^i_{O_i})^\top \mathbf{z}^i_{O_i}] - 2\mathrm{tr}(\mathbf{W}_{O_i}\mathbb{E}[\mathbf{t}^i(\mathbf{z}^i_{O_i})^\top]) + \mathrm{tr}(\mathbf{W}_{O_i}^\top \mathbf{W}_{O_i}\mathbb{E}[\mathbf{t}^i(\mathbf{t}^i)^\top])\right)}{2\sigma^2}. \quad (3.7)$$

where c is an absolute constant in terms the model parameters $\mathbf{W}$ and σ^2. The key fact we use to derive Eq. (3.7) is that conditional on known $\mathbf{z}_O$, $\mathbf{t}$ is normally distributed:

$$\mathbf{t}|\mathbf{z}_O \sim \mathcal{N}(\mathbf{M}_O^{-1}\mathbf{W}_O^\top \mathbf{z}_O, \sigma^2 \mathbf{M}_O^{-1}), \quad (3.8)$$

where $\mathbf{M}_O = \sigma^2 \mathbf{I}_k + \mathbf{W}_O^\top \mathbf{W}_O$. This result follows by applying the Bayes formula with $\mathbf{z}_O|\mathbf{t} \sim \mathcal{N}(\mathbf{W}_O\mathbf{t}, \sigma^2\mathbf{I}_p)$, $\mathbf{t} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_k)$ and $\mathbf{z}_O \sim \mathcal{N}(\mathbf{0}, \mathbf{W}_O\mathbf{W}_O^\top + \sigma^2\mathbf{I}_p)$. According to Eq. (3.7), it suffices to compute $\mathbb{E}[(\mathbf{z}^i_{O_i})^\top \mathbf{z}^i_{O_i}], \mathbb{E}[\mathbf{t}^i(\mathbf{z}^i_{O_i})^\top]$ and $\mathbb{E}[\mathbf{t}^i(\mathbf{t}^i)^\top]$, presented in Lemma 6.

Lemma 6. Suppose $\mathbf{x} \sim \mathrm{LRGC}(\mathbf{W}, \sigma^2, \mathbf{f})$ with observations $\mathbf{x}_O$ and missing entries $\mathbf{x}_\mathcal{M}$, with $\mathbf{W} \in \mathbb{R}^{p\times k}$. Then for the latent normal vector $\mathbf{z}$ and latent isotropic normal vector $\mathbf{t}$ satisfying $\mathbf{f}(\mathbf{z}) = \mathbf{x}$ and $\mathbf{z} = \mathbf{W}\mathbf{t} + \boldsymbol{\epsilon}$, with corresponding latent subvectors $\mathbf{z}_O$ and $\mathbf{z}_\mathcal{M}$,

$$\mathbb{E}[\mathbf{t}|\mathbf{x}_O] = \mathbf{M}_O^{-1}\mathbf{W}_O^\top \mathrm{E}[\mathbf{z}_O|\mathbf{x}_O], \quad \text{where } \mathbf{M}_O = \sigma^2\mathbf{I}_k + \mathbf{W}_O^\top \mathbf{W}_O. \quad (3.9)$$

$$\mathbb{E}[\mathbf{t}(\mathbf{z}_O)^\top|\mathbf{x}_O] = \mathbb{E}[\mathbf{t}|\mathbf{x}_O]\mathrm{E}[\mathbf{z}_O|\mathbf{x}_O]^\top + \mathbf{M}_O^{-1}\mathbf{W}_O^\top \mathrm{Cov}[\mathbf{z}_O|\mathbf{x}_O]. \quad (3.10)$$

$$\mathbb{E}[\mathbf{t}\mathbf{t}^\top|\mathbf{x}_O] = \sigma^2\mathbf{M}_O^{-1} + \mathbb{E}[\mathbf{t}|\mathbf{x}_O]\mathbb{E}[\mathbf{t}|\mathbf{x}_O]^\top + \mathbf{M}_O^{-1}\mathbf{W}_O^\top \mathrm{Cov}[\mathbf{z}_O|\mathbf{x}_O]\mathbf{W}_O\mathbf{M}_O^{-1}. \quad (3.11)$$

Recall that for continuous data, $\mathrm{E}[\mathbf{z}^i_{O_i}|\mathbf{x}^i_{O_i}] = \mathbf{f}^{-1}_{O_i}(\mathbf{x}^i_{O_i})$ and $\mathrm{Cov}[\mathbf{z}^i_{O_i}|\mathbf{x}^i_{O_i}] = \mathbf{0}$. For ordinal data, these quantities are the mean and covariance of a truncated normal vector, for each row i separately at each EM iteration. In Section 2.4.3, we introduced a fast iterative method to estimate those quantity, where we resort to a diagonal approximation of $\mathrm{Cov}[\mathbf{z}^i_{O_i}|\mathbf{x}^i_{O_i}]$. Note here the diagonal approximation also reduces the computation for Eq. (3.11)) from $O(|O|^2k)$ to $O(|O|k^2)$.

M step Let $\mathbf{e}_j \in \mathbb{R}^p$ be the jth standard basis vector. Take the derivative of the Q-function in Eq. (3.7) with respect to row $\mathbf{w}_j^\top$ and σ^2:

$$\begin{aligned}
\frac{\partial Q}{\partial \mathbf{w}_j^\top} &= \frac{-1}{|\Omega_j|\sigma^2}\sum_{i\in\Omega_j}(-\mathbf{e}_j^\top\mathbb{E}[\mathbf{z}^i_{O_i}\mathbf{t}_i^\top] + \mathbf{w}_j^\top\mathbb{E}[\mathbf{t}_i\mathbf{t}_i^\top]),\\
\frac{\partial Q}{\partial \sigma^2} &= \frac{1}{2\sigma^4}\sum_{i=1}^{n}\left(\mathbb{E}[(\mathbf{z}^i_{O_i})^\top\mathbf{z}^i_{O_i}] - 2\mathrm{tr}(\mathbf{W}_{O_i}\mathbb{E}[\mathbf{t}_i(\mathbf{z}^i_{O_i})^\top]) + \mathrm{tr}(\mathbf{W}_{O_i}^\top\mathbf{W}_{O_i}\mathbb{E}[\mathbf{t}_i\mathbf{t}_i^\top])\right) - \frac{\sum_{i=1}^n |O_i|}{2\sigma^2}.
\end{aligned}$$

Set both to zero to obtain the update for M-step:

$$\hat{\mathbf{w}}_j^\top = \left(\mathbf{e}_j^\top\sum_{i\in\Omega_j}\mathbb{E}[\mathbf{z}^i_{O_i}(\mathbf{t}^i)^\top]\right)\left(\sum_{i\in\Omega_j}\mathbb{E}[\mathbf{t}^i(\mathbf{t}^i)^\top]\right)^{-1},\ \hat{\sigma}^2 = \frac{\sum_{i=1}^n \mathbb{E}\left[\|\mathbf{z}^i_{O_i} - \hat{\mathbf{W}}_{O_i}\mathbf{t}^i)\|_2^2\right]}{\sum_{i=1}^n |O_i|}. \quad (3.12)$$

The maximizer $(\hat{\mathbf{W}}, \hat{\sigma}^2)$ increase the observed likelihood in Eq. (3.5) compared to the initial estimate $(\tilde{\mathbf{W}}, \tilde{\sigma}^2)$ [66, Chapter 3]. To satisfy the unit diagonal constraints $\|\mathbf{w}_j\|_2^2+\sigma^2 = 1$, we approximate the constrained maximizer by scaling the unconstrained maximizer shown in Eq. (3.13) as in [39, 103]:

$$\hat{\sigma}^2 \leftarrow \hat{\sigma}^2_{\text{new}} = \frac{1}{p}\sum_{j=1}^{p}\frac{\hat{\sigma}^2}{\|\hat{\mathbf{w}}_j\|_2^2 + \hat{\sigma}^2}, \quad \hat{\mathbf{w}}_j \leftarrow \frac{\hat{\mathbf{w}}_j}{\|\hat{\mathbf{w}}_j\|_2}\cdot\sqrt{1-\hat{\sigma}^2_{\text{new}}}. \quad (3.13)$$

We find this approximation works well in practice.

Stopping criteria We use the relative change of the parameter $\mathbf{W}$ as the stopping criterion. Concretely, with $\mathbf{W}_1$ from last iteration and $\mathbf{W}_2$ from current iteration, the algorithm stops if $\frac{\|\mathbf{W}_1-\mathbf{W}_2\|_F^2}{\|\mathbf{W}_1\|_F^2}$ is smaller than the tolerance level. We observe the algorithm converges in no more than 50 iterations in most cases.

Computation cost The computational complexity for each iteration is $O(|\Omega|k^2 + nk^3 + pk^3)$, upper bounded by $O(npk^2)$. We find the method usually converges in fewer than 50 iterations across our experiments. See the Movielens 1M experiment in Section 3.5 for a run time comparison with state-of-the-are methods.

Algorithm 4 Imputation via low rank Gaussian copula fitting

```
Input: $\mathbf{X} \in \mathbb{R}^{n\times p}$ observed on $\Omega$, rank $k$, $t_{\max}$.
1: Compute the empirical CDF $\hat{F}_j$ and empirical quantile function $\hat{F}_j^{-1}$ on observed $\mathbf{X}_j$, for $j \in [p]$.
2: Estimate $\hat{f}_j = \Phi^{-1} \circ \frac{n}{n+1}\hat{F}_j$ and $\hat{f}_j^{-1} = \hat{F}_j^{-1} \circ \Phi$, for $j \in [p]$.
3: Initialize: $\mathbf{W}^{(0)}, (\sigma^2)^{(0)}$
4: for $t = 1, 2, \ldots, t_{\max}$ do
5:     E-step: compute the required conditional expectation using Eq. (3.9-3.11).
6:     M-step: update $\mathbf{W}^{(t)}, (\sigma^2)^{(t)}$ using Eq. (3.12-3.13).
7: end for
8: Impute $\hat{\mathbf{x}}^i_{\mathcal{M}_i}$ using Definition 2 for $i \in [n]$ with $\mathbf{f} = \hat{\mathbf{f}}, \mathbf{W} = \mathbf{W}^{(t_{\max})}, \sigma^2 = (\sigma^2)^{(t_{\max})}$.
Output: $\hat{\mathbf{X}}$ with imputed $\hat{x}^i_j$ at $(i, j) \in \Omega^c$ and observed $x^i_j$ at $(i, j) \in \Omega$.
```

3.4 Imputation error bound

The imputation error consists of two parts: (1) the random variation of the error under the true LRGC model; (2) the estimation error of the LRGC model. Analyzing the estimation error (2) is challenging for output from EM algorithm; moreover, in our experiments we find that the imputation error can be attributed predominantly to (1), detailed in the appendix. Hence we leave (2) to future work. To analyze the random variation of the error under the true LRCG model, we examine the MSE of $\hat{\mathbf{x}}$ for a random row $\mathbf{x} \sim \mathrm{LRGC}(\mathbf{W}, \sigma^2, \mathbf{f})$ with fixed missing locations $\mathcal{M}$: $\mathrm{MSE}(\hat{\mathbf{x}}) = \|\mathbf{f}_{\mathcal{M}}(\hat{\mathbf{z}}_{\mathcal{M}}) - \mathbf{f}_{\mathcal{M}}(\mathbf{z}_{\mathcal{M}})\|_2^2/|\mathcal{M}|$. For continuous $\mathbf{x}$ with strictly monotone $\mathbf{f}$, we must assume that $\mathbf{f}$ is Lipschitz to obtain a finite bound on the error. With this assumption and assuming $\mathbf{W}, \sigma^2$ fixed and known, we can use the fact that $\mathbf{z}_{\mathcal{M}}|\mathbf{x}_{\mathcal{O}}$ is normal to bound large deviations of the MSE.

Theorem 1. Suppose subvector $\mathbf{x}_{\mathcal{O}}$ of $\mathbf{x} \sim \mathrm{LRGC}(\mathbf{W}, \sigma^2, \mathbf{f})$ is observed and that all marginals $\mathbf{f}$ are strictly monotone with Lipschitz constant L. Denote the largest and the smallest singular values of $\mathbf{W}'$ as $\lambda_1(\mathbf{W}')$ and $\lambda_k(\mathbf{W}')$. Then for any $t > 0$, the imputed values $\hat{\mathbf{x}}$ in Definition 2 satisfy

$$\Pr\left[\mathrm{MSE}(\hat{\mathbf{x}}) > L^2\sigma^2\left(\sqrt{1 + \frac{1-\sigma^2}{\sigma^2 + \lambda_k^2(\mathbf{W}_{\mathcal{O}})}} + \sqrt{2\left(1 + \frac{\lambda_1^2(\mathbf{W}_{\mathcal{M}})}{\sigma^2 + \lambda_k^2(\mathbf{W}_{\mathcal{O}})}\right)\frac{t}{|\mathcal{M}|}}\right)^2\right] \leq e^{-t}.$$

Theorem 1 indicates the imputation error concentrates at $\sigma^2 + \frac{\sigma^2(1-\sigma^2)}{\sigma^2 + \lambda_k^2(\mathbf{W}_{\mathcal{O}})/\sigma^2}$ with an expansion multiplier L^2 due to the marginals $\mathbf{f}$. The first term σ^2 represents the fraction of variance due to noise and the second term is small when the SNR is large. We also analyze the distribution of $\lambda_k^2(\mathbf{W}_{\mathcal{O}})/\sigma^2$ under a random design to provide insight into when the error is small: in Corollary 1, the second term vanishes with increasing observed length $|\mathcal{O}|$.

Corollary 1. Under the conditions of Theorem 1, further assume $\mathbf{W}$ has independent sub-Gaussian rows $\mathbf{w}_j$ with zero mean and covariance $\frac{1-\sigma^2}{k}\mathbf{I}_k$ for $j \in [p]$. Suppose $c_1 k < |\mathcal{O}| < c_2|\mathcal{M}|$ for some constant $c_1 > 0$ depending on the sub-Gaussian norm of the scaled rows $\sqrt{\frac{k}{1-\sigma^2}}\mathbf{w}_j$ and some absolute constant $c_2 > 0$.

Then for some constant $c_3 > 0$ depending on c_1, c_2,

$$\Pr\left[\mathrm{MSE}(\hat{\mathbf{x}}) > L^2\sigma^2\left(1 + K_{|\mathcal{O}|}\right)\right] \le c_3/|\mathcal{O}|, \quad \text{where } K_{|\mathcal{O}|} = O\left(\sqrt{\log(|\mathcal{O}|)/|\mathcal{O}|}\right). \quad (3.14)$$

See the appendix for definition of a sub-Gaussian vector. Analyzing the imputation error for ordinal $\mathbf{x}$ is much harder since $\mathbf{z}_{\mathcal{M}}|\mathbf{x}_{\mathcal{O}}$ is no longer Gaussian. We leave that for future work.

3.5 Experiments

Our experiments evaluate the imputation accuracy of `LRGC`. For comparison, we implement LRMC methods `softImpute` [64], `GLRM` [91] with ℓ_2, ℓ_1, bigger vs smaller (BvS, for ordinal data), hinge, and logistic loss. We also implement the high rank matrix completion method `MMC` [36], and `PPCA`, a special case of `LRGC` with Gaussian marginals.

3.5.1 Synthetic experiments

We consider three data types from LRGC: continuous, 1-5 ordinal and binary. We generate $\mathbf{W} \in \mathbb{R}^{p\times k}, \mathbf{T} \in \mathbb{R}^{n\times k}, \mathbf{E} \in \mathbb{R}^{n\times p}$ with independent standard normal entries, then scale each row of $\mathbf{W}$ such that $\|\mathbf{w}_j\|_2^2 + \sigma^2 = 1$. Then generate $\mathbf{X} = \mathbf{f}(\mathbf{Z}) = \mathbf{f}(\mathbf{T}\mathbf{W}^\top + \sigma\mathbf{E})$ using $\mathbf{f}$ described below. Missing entries of $\mathbf{X}$ are uniformly sampled. We set $n = 500$ and $p = 200$. For continuous data, we use $f_j(z) = z$ to generate a low rank $\mathbf{X} = \mathbf{Z}$ and $f_j(z) = z^3$ to generate a high rank $\mathbf{X}$. We set $k = 10, \sigma^2 = 0.1$ and the missing ratio as 40%. For 1-5 ordinal data and binary

Table 3.1: Imputation error (NRMSE for continuous and MAE for ordinal) reported over 20 repetitions, with rank r for available methods. GLRM methods are trained at rank 199.

Continuous	LRGC	PPCA	softImpute	GLRM-ℓ_2	MMC
Low Rank	.347(.004), $r = 10$	**.338**(**.004**), $r = 10$	.371(.004), $r = 117$	.364(.003)	.633(.007), $r = 130$
High Rank	**.517**(**.011**), $r = 10$	.690(.010), $r = 10$	.703(.005), $r = 104$	.696(.006)	.824(.011), $r = 137$
1-5 ordinal	LRGC	PPCA	softImpute	GLRM-BvS	GLRM-ℓ_1
High SNR	**.358**(**.008**), $r = 5$	.501(.010), $r = 6$	.582(.011), $r = 83$	.407(.007)	.689(.010)
Low SNR	**.788**(**.013**), $r = 5$	.863(.013), $r = 5$	.951(.015), $r = 38$	.850(.011)	1.027(.020)
Binary	LRGC	PPCA	softImpute	GLRM-hinge	GLRM-logistic
High SNR	**.103**(**.003**), $r = 5$	.116(.002), $r = 6$	.136(.003), $r = 71$	.140(.002)	.117(.002)
Low SNR	**.205**(**.006**), $r = 5$	.208(.005), $r = 5$	.234(.007, $r = 61$	.226(.006)	.217(.005)

data, we use step functions f_j with random selected cut points. We generate one $\mathbf{X}$ with high SNR $\sigma^2 = 0.1$ and one $\mathbf{X}$ with low SNR $\sigma^2 = 0.5$. We set $k = 5$ and the missing ratio as 60%.

We examine the sensitivity of each method to its key tuning parameter. Both LRGC and PPCA do not overfit with large ranks. We report results using the best tuning parameter in Table 3.1. The complete results and implementation details appear in the appendix. All experiments are repeated 20 times.

Shown in Table 3.1, LRGC performs the best in all but one settings. The improvement is significant for high rank continuous data. For low rank continuous data, PPCA performs the best as expected since the model is correctly specified. The slightly larger error of LRGC is due to the error in estimating a nonparametric marginal $\mathbf{f}$. Notice both LRGC and PPCA admit much smaller rank as best parameter.

Table 3.2: Imputation error for MovieLens 1M over 5 repetition. Run time is measures in minutes.

Algorithm	MAE	RMSE	Run time
LRGC	**0.619(.002)**	0.910(.003)	38(1)
softImpute	0.629(.003)	**0.905(.003)**	93(2)
MMMF-BvS	0.633(.002)	0.921(.002)	25(1)

3.5.2 Movielens 1M

We sample the subset of the MovieLens1M data [40] consisting of 2514 movies with at least 50 ratings from 6040 users. We use 80% of observation as training set, 10% as validation set, and 10% as test set, repeated 5 times. The results are reported in Table 3.2. On a laptop with Intel-i5-3.1GHz Core and 8 GB RAM, `LRGC` (rank 10) takes 38 mins in R, `softImpute` (rank 201) takes 93 mins in R, and `GLRM-BvS` (rank 200) takes 25 mins in julia. We see that all the models perform quite similarly on this large dataset. In other words, the gain from carefully modeling the marginal distributions (using a LRGC) is insignificant. This phenomenon is perhaps unsurprising given that sufficiently large data matrices from a large class of generative models are approximately low rank [90].

CHAPTER 4
IMPUTATION UNCERTAINTY QUANTIFICATION

This chapter develops multiple measures to quantify uncertainty of the imputation derived from the Gaussian copula model, including the low rank Gaussian copula. This chapter is based on [102].

4.1 Introduction

Missing data imputation forms the first critical step of many data analysis pipelines; indeed, in the context of recommender systems, imputation itself is the task. The remarkable progress in low rank matrix completion (LRMC) [15, 53, 76] has led it to wide use in collaborative filtering [78], transductive learning [37], automated machine learning [98], and beyond. Nevertheless, reliable decision making requires one more step: assessing the uncertainty of the imputed entries. While multiple imputation [81, 59] is a classical tool to quantify uncertainty, its computation is often expensive and limits the use on large datasets. For single imputation methods such as LRMC, very little work has sought to quantify imputation uncertainty. The major difficulty in quantifying uncertainty lies in characterizing how the imputations depend on the observations through the solution to a nonsmooth optimization problem. In [21], this difficulty is avoided and confidence intervals for imputed real valued matrices are provided, by assuming isotropic Gaussian noise and a large signal-to-noise ratio (SNR). However, these assumptions are hardly satisfied for most noisy real data.

The probabilistic principal component analysis (PPCA) model [86] provides a different approach to quantify uncertainty. The PPCA model posits that the data in each row is sampled iid from a Gaussian factor model. In this framework, each missing entry has a closed form distribution conditional on the observations. The conditional mean, which is simply a linear transformation of the observations, is used for imputation [101, 48]. However, the Gaussian assumption is unrealistic for most real datasets.

Our contribution

The Gaussian copula model presents a compelling alternative that enjoys the analytical benefits of Gaussians and yet fits real datasets well. Here we further propose to quantify the uncertainty associated with a single Gaussian copula imputation. Concretely, we construct confidence intervals for imputed real values and provide lower bounds on the probability of correct prediction for imputed ordinal values. Empirical results show our confidence intervals are well-calibrated and our uncertainty measure predicts imputation error well: entries with lower estimated uncertainty do have lower imputation error (on average).

Related work

Multiple imputation (MI) requires repeating an imputation procedure many times to assess empirical uncertainty, often through bootstrap sampling [51, 5] or Bayesian posterior sampling [14] including probabilistic matrix factorization [67, 82]. The repeating procedure often leads to very expensive computation, especially for large datasets. While variational inference can accelerate the process in some cases [58], it may produce inaccurate results due to using overly simple approximation. Moreover, *proper* multiple imputation generally relies on strong distributional assumptions [67, 82]. In contrast, our quantified uncertainty estimates are useful for a much broader family of distributions and can be computed as fast as a single imputation. In addition, few MI papers explicitly explore the issue of *calibration*: does MI sample variance predict imputation accuracy? We find that the answer is usually no. In contrast, our uncertainty metric is clearly correlated with imputation accuracy.

Some interesting new approaches [19, 20] discuss constructing *honest* confidence regions, which depends on some (possibly huge) hidden constants. However, these unknown hidden constants prevent its use in practice. In contrast, our constructed confidence intervals are explicit.

4.2 Imputation uncertainty measure

Suppose we have observed a few entries $\mathbf{x}_{\mathcal{O}}$ of a vector $\mathbf{x} \sim \mathrm{GC}(\Sigma, \mathbf{f})$ with known Σ and $\mathbf{f}$. We can impute the missing entries $\mathbf{x}_{\mathcal{M}}$ as in Algorithm 1. Can we quantify the uncertainty in these imputations? Different from LRMC model which assumes a deterministic true value for missing locations, $\mathbf{x}_{\mathcal{M}}$ (as well as $\mathbf{z}_{\mathcal{M}}$ satisfying $\mathbf{f}_{\mathcal{M}}(\mathbf{z}_{\mathcal{M}}) = \mathbf{x}_{\mathcal{M}}$) is random under the Gaussian copula model. Consequently, the error $\hat{\mathbf{x}}_{\mathcal{M}} - \mathbf{x}_{\mathcal{M}}$ is random and hence uncertain even with deterministic imputation $\hat{\mathbf{x}}_{\mathcal{M}}$. The uncertainty depends on the concentration of $\mathbf{z}_{\mathcal{M}}$ around its mean $\mathrm{E}[\mathbf{z}_{\mathcal{M}}|\mathbf{x}_{\mathcal{O}}]$ and on the marginals $\mathbf{f}_{\mathcal{M}}$. If $\mathbf{f}_{\mathcal{M}}$ is constant or nearly constant over the likely values of $\mathbf{z}_{\mathcal{M}}$, then with high probability the imputation is accurate. Otherwise, the current observations cannot predict the missing entry well and we should not trust the imputation. Using this intuition, we may formally quantify the uncertainty in the imputations.

Imputation confidence interval

For continuous data, we construct confidence intervals using the normality of $\mathbf{z}_{\mathcal{M}}|\mathbf{z}_{\mathcal{O}} = \mathbf{f}_{\mathcal{O}}^{-1}(\mathbf{x}_{\mathcal{O}})$.

Theorem 2 (Uncertainty quantification for continuous data)**.** Suppose $\mathbf{x} \sim \mathrm{GC}(\Sigma, \mathbf{f})$ with observations $\mathbf{x}_{\mathcal{O}}$ and missing entries $\mathbf{x}_{\mathcal{M}}$ and that $\mathbf{f}$ is elementwise strictly monotone. For missing entry x_j, for any $\alpha \in (0,1)$, let $z^\star = \Phi^{-1}(1-\frac{\alpha}{2})$, the following holds with probability $1-\alpha$:

$$x_j \in \left[f_j(\mathrm{E}[z_j|\mathbf{x}_{\mathcal{O}}] - z^\star \mathrm{Var}[z_j|\mathbf{x}_{\mathcal{O}}]), f_j(\mathrm{E}[z_j|\mathbf{x}_{\mathcal{O}}] + z^\star \mathrm{Var}[z_j|\mathbf{x}_{\mathcal{O}}])\right] =: [x_j^-(\alpha), x_j^+(\alpha)] \quad (4.1)$$

where $\mathrm{E}[z_j|\mathbf{x}_{\mathcal{O}}]$ and $\mathrm{Var}[z_j|\mathbf{x}_{\mathcal{O}}]$ ares given in Eq. (2.8) and Eq. (2.9) with $\mathcal{M}$ replaced by j, for $j \in \mathcal{M}$.

If some observed variable in $\mathbf{x}_{\mathcal{O}}$ is not continuous, then Theorem 2 no longer holds. In those cases, we may compute an approximate confidence interval by assuming that $\mathbf{z}_{\mathcal{O}}$ has all probability mass at its conditional mean given $\mathbf{x}_{\mathcal{O}}$ and then compute the normal confidence interval of $\mathbf{z}_{\mathcal{M}}$ as it does for all continuous variables. The approximated confidence intervals are still reasonably well calibrated if there are not too many ordinal variables. However, a safer approach to build confidence intervals by performing multiple imputation as in Algorithm 8 and taking a confidence interval on the empirical percentiles of imputed values.

Lower bound for correct ordinal prediction

For ordinal data, we lower bound the probability of correct prediction $x_j = \hat{x}_j$ using a sufficient condition that z_j is sufficiently close to its mean $\mathrm{E}[z_j|\mathbf{x}_O]$. General results in bounding $\Pr(|\hat{x}_j - x_j| \leq d)$ for any $d \in \mathbb{N}$ appear in the appendix. Note a step function $f_j(z)$ with cut points set admits the form $\mathbf{S}$: $f_j(z) = 1 + \sum_{s\in\mathbf{S}} \mathbb{1}(z > s)$.

Theorem 3 (Uncertainty quantification for ordinal data)**.** Suppose $\mathbf{x} \sim \mathrm{GC}(\Sigma, \mathbf{f})$ with observations $\mathbf{x}_O$ and missing entries $\mathbf{x}_\mathcal{M}$ and that the marginal f_j is a step function with cut points $\mathbf{S}_j$, for $j \in [p]$. For missing entry x_j and its imputation $\hat{x}_j = f_j(\mathrm{E}[z_j|\mathbf{x}_O])$,

$$\Pr(\hat{x}_j = x_j) \geq 1 - \mathrm{Var}[z_j|\mathbf{x}_O]/d_j^2 \quad \text{where} \quad d_j = \min_{s\in\mathbf{S}_j} \left|s - \mathrm{E}[z_j|\mathbf{x}_O]\right|, \tag{4.2}$$

where $\mathrm{E}[z_j|\mathbf{x}_O]$ and $\mathrm{Var}[z_j|\mathbf{x}_O]$ ares given in Eq. (2.8) and Eq. (2.9) with $\mathcal{M}$ replaced by j, for $j \in \mathcal{M}$.

Imputation reliability

To predict the imputation accuracy using the quantified uncertainty, we develop a measure we call *reliability*. Entries with higher reliability are expected to have smaller imputation error. We first motivate our definition of reliability. For ordinal data, the reliability of an entry lower bounds the probability of correct prediction. For continuous data, our measure of reliability is designed so that reliable imputations have low normalized root mean squared error (NRMSE) under a certain confidence level α.

Our definition of reliability uses Theorems 2-3 to ensure that reliable imputations have low error.

Definition 4 (Imputation Reliability)**.** Suppose $\mathbf{X}$ has iid rows $\mathbf{x}^i \sim \mathrm{GC}(\Sigma, \mathbf{f})$ and is observed on $\Omega \subset [n] \times [p]$. Complete $\mathbf{X}$ to $\hat{\mathbf{X}}$ row-wise using Definition 2. For each missing entry $(i, j) \in \Omega^c$, define the *reliability* of the imputation $\hat{x}^i_j$ as

- (if $\mathbf{X}$ is an ordinal matrix) the lower bound provided in Eq. (4.2);
- (if $\mathbf{X}$ is a continuous matrix) $\|P_{\Omega^c \setminus (i,j)}(D_\alpha)\|_F / \|P_{\Omega^c \setminus (i,j)}(\hat{\mathbf{X}})\|_F$, where the (i', j')-th entry of matrix D_α is the length of the confidence interval $\hat{x}^{i',+}_{j'}(\alpha) - \hat{x}^{i',-}_{j'}(\alpha)$ defined in Eq. (4.1).

For continuous data, the interpretation is that if the error after removing (i_1, j_1) is larger than that after removing (i_2, j_2), then the imputation on (i_1, j_1) is more reliable than that on (i_2, j_2). If continuous entries in different columns are measured on very different scales, one can also modify the definition to compute reliability column-wise.

Our experiments show this reliability measure positively correlates with imputation accuracy as measured by mean absolute error (MAE) for ordinal data and NRMSE for continuous data. We also find for continuous data, the correlation is insensitive to α in a reasonable range; $\alpha = .05$ works well.

4.3 Experiments

The uncertainty measure proposed in this chapter applies to the general Gaussian copula as well as the low rank Gaussian copula (LRGC). Here we use LRGC as an example. Our experiments follow the settings in Section 3.5. We evaluate whether our reliability measure (denoted as `LRGC` reliability) can predict imputation accuracy well, and the empirical coverage of our proposed confidence intervals. For the second task, we evaluate the imputation on the *m%* entries with highest reliability for varying *m*. We say a measure predicts imputation accuracy if the imputation error on the *m%* entries is smaller for smaller *m*, i.e., it positively correlates with imputation accuracy. We introduce below competitors for each task. Implementation details appear in the appendix.

For reliability comparison, we compare with variance based reliability: the imputation for a given missing entry is more reliable if it has smaller variance. To obtain variance estiamte, we implement the PCA based MI method (denoted as `MI-PCA`) [51, 52], and construct MI style uncertainty quantification for general imputation algorithms: given an algorithm and incomplete $\mathbf{X}$, divide the observations into N parts. Then apply the algorithm N times, each time additionally masking one part of the observations. Compute the variance of the original missing entries across N estimates. We use $N = 10$ in this paper. We denote such methods as `MI+Algorithm` for applied algorithm.

For confidence interval (CI) comparison, we compare with CI based on the `softImpute` imputation (denoted as `LRMC`) [21] , CI based on `PPCA` and CI based on `MI-PCA`. All constructed intervals except `LRGC` are derived assuming normality: specifically, they all assume $\mathbf{X} = \mathbf{X}^\star + \mathbf{E}$ for some low rank $\mathbf{X}^\star$ and isotropic Gaussian error $\mathbf{E}$. In particular, their CIs are always symmetric around the imputed value, while `LRGC` can yield asymmetrical CIs. See the appendix for implementation details.

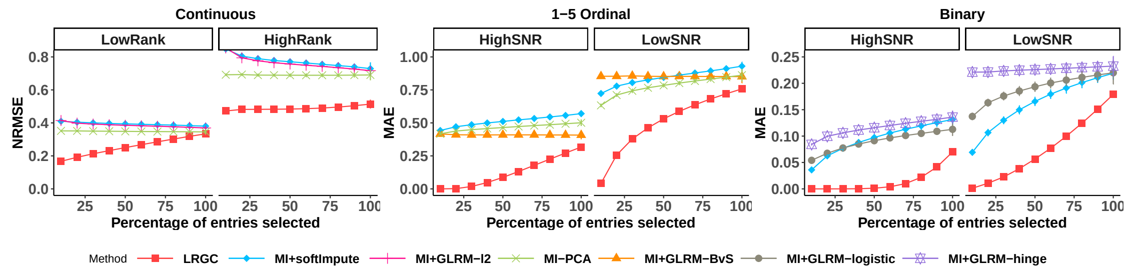


Figure 4.1: Imputation error on the subset of $m\%$ entries for which method's associated uncertainty metric indicates highest reliability, reported over 20 repetitions (error bars almost invisible).

4.3.1 Synthetic experiments

Shown in Figure 4.1, `LRGC` reliability predicts the imputation accuracy well: entries with higher reliability (smaller m) have higher accuracy. In contrast, entries with higher variance based reliability can have lower accuracy. Even when the variance based reliability predicts accuracy, `LRGC` reliability works better: the error over selected entries using variance based reliability is much larger than that of `LRGC` reliability when a small percentage of entries m are selected. `LRGC` reliability can even find entries with error near 0 from very noisy (low SNR 1-5 ordinal and binary) data. `LRGC` reliability better predicts imputation error for easier imputation tasks (lower rank and higher SNR). Predicting NRMSE is challenging, since imputing continuous data is in general harder than imputing ordinal data. In fact, we show in the appendix that as the number of levels of the ordinal variable increases, the shape of the error vs reliability curve matches that of continuous data.

The results on confidence intervals appear in Table 4.1. Notice constructing `MI−PCA` intervals is much more expensive than all other methods. For low rank Gaussian data, `PPCA` confidence intervals achieve the highest coverage rates with smallest length as expected, since the model is correctly specified. `LRGC`

Table 4.1: 95% Confidence intervals on synthetic continuous data over 20 repetitions.

Low Rank Data	LRGC	PPCA	LRMC	MI-PCA
Empirical coverage rate	0.927(.002)	0.940(.001)	0.878(.006)	0.933(.002)
Interval length	1.273(.004)	1.264(.004)	1.129(.015)	1.267(.004)
Run time (in seconds)	6.9(.5)	3.4(.7)	2.7(.4)	189.8(15.4)
High Rank Data	**LRGC**	**PPCA**	**LRMC**	**MI-PCA**
Empirical coverage rate	0.927(.002)	0.943(.002)	0.925(.004)	0.948(.002)
Interval length	3.614(.068)	9.086(.248)	6.546(.191)	9.307(.249)
Run time (in seconds)	7.2(1.2)	0.4(.1)	3.1(.6)	220.0(30.2)

confidence intervals have slightly smaller coverage rates due to the error in estimating a nonparametric marginal $\mathbf{f}$. For high rank data, the normality and the low rank assumption do not hold, so all other constructed confidence intervals but LRGC are no longer theoretically valid. Notably, LRGC confidence intervals for more challenging high rank data achieves the same empirical coverage rates as that for low rank data. The longer interval is due to the expanding marginal transformation $f_j(z) = z^3$. While all other confidence intervals have visually good coverage rates, their interval lengths are much larger than LRMC confidence intervals, which limits utility.

4.3.2 MovieLens 1M dataset

For the MovieLens 1M dataset, we exclude MI-PCA because it cannot finish even a single imputation in 3 hours in R. We plot the imputation error versus reliability in Figure 4.2. The value at $m = 100$ is the overall imputation error. The variance based reliability with GLRM-BvS cannot predict imputation accuracy. In practice, collaborative filtering methods usually recommends very few entries to users. In this setting, LRGC reliability predicts imputation accuracy

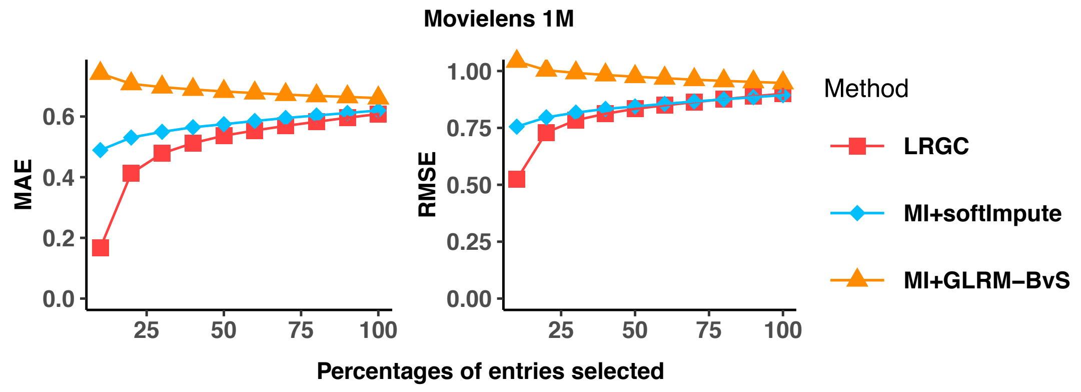


Figure 4.2: Imputation error on the subset of $m\%$ entries for which method's associated uncertainty metric indicates highest reliability, reported over 5 repetitions (error bars almost invisible).

much better than variance based reliability with `softImpute`.

CHAPTER 5

ONLINE IMPUTATION

This chapter develops a new online missing value imputation algorithm using a mini-batch Gaussian copula fitting algorithm. It is based on [105].

5.1 Introduction

Missing values also appear in online data, generated by sensor networks, or ongoing surveys, as sensors fail or survey respondents fail to respond. In this setting, online (immediate) imputation for new data points is important to facilitate online decision-making processes. However, most missing value imputation methods, including missForest [85] and MICE [14], cannot easily update model parameters with new observation in the online setting. Re-applying offline methods after seeing every new observation consumes too much time and space. Online methods, which incrementally update the model parameters every time new data is observed, enjoy lower space and time costs and can adapt to changes in the data, and hence are sometimes preferred even in the offline setting.

Our contribution

Here we propose an online algorithm to impute missing values for long skinny mixed data, including real-valued data and ordinal data as special cases. Our online imputation method builds on the offline Gaussian copula imputation model introduced in Chapter 2. We make two major contributions here:

- We propose an online algorithm for missing value imputation using the Gaussian copula model, which incrementally updates the model and thus can adapt to a changing data distribution. It does not need to store historical data.
- We develop a mini-batch Gaussian copula fitting algorithm to accelerate the training in the offline setting and a parallel implementation. Compared to the offline algorithm (Algorithm 3), our methods achieve nearly the same imputation accuracy but being an order of magnitude faster, which allows the Gaussian copula model to scale to larger datasets.

Inheriting the advantages of the Gaussian copula model, all our proposed methods naturally handle long skinny mixed data with missing values, and have no model hyperparameters except for common online learning rate parameters. This property is crucial in the online setting, where the best model hyperparameters may evolve.

Related work

The Gaussian copula has been used to impute incomplete mixed data in the offline setting using an EM algorithm (Algorithm 3). Here, we develop an online EM algorithm to incrementally update the copula correlation matrix, following [18], and an online method to estimates the marginals, so that there is no need to store historical data except for the previous model estimate.

Existing online imputation methods mostly rely on matrix factorization (MF). Online LRMC methods [7, 28] assume a low rank data structure. Consequently, they work poorly for long skinny data, as the low rank assumption generally fails [89]. Online KFMC [31] first maps the data to a high dimensional space and assumes the mapped data has a low rank structure. It learns a non-linear structure and outperforms online LRMC for long skinny data. However, its performance is sensitive to a selected rank r, which should be several times larger than the data dimension p and thus needs to be carefully tuned in a wide range. As p increases, it also requires increasing r to outperform online LRMC methods; for moderate p, the $O(r^3)$ computation time of online KFMC becomes prohibitive. For all aforementioned MF methods, their underlying continuity assumptions can lead to poor performance on mixed data. Moreover, the sensitivity to the rank poses a difficulty in the online setting, as the best rank may vary over time, and the rank chosen by cross-validation early on can lead to poor performance or even divergence later.

While recent deep generative imputation methods [100, 62] look like online methods (due to the SGD update), they actually require lots of data, and are slow to adapt to changes in the data stream, which are unsatisfying for real-time tasks. Deep time series imputation methods [17, 35] use the future to impute the past, and thus do not suit the considered online imputation task.

5.2 Parameter estimation from online data

We can still use the general Gaussian copula imputation algorithm (Algorithm 1):

$$\hat{\mathbf{x}}_{\mathcal{M}} = \mathbf{f}_{\mathcal{M}}(\mathbb{E}[\mathbf{z}_{\mathcal{M}}|\mathbf{x}_{\mathcal{O}}, \Sigma, \mathbf{f}]) = \mathbf{f}_{\mathcal{M}}(\Sigma_{\mathcal{M},\mathcal{O}}\Sigma_{\mathcal{O},\mathcal{O}}^{-1}\mathbb{E}[\mathbf{z}_{\mathcal{O}}|\mathbf{x}_{\mathcal{O}}, \Sigma, \mathbf{f}]). \tag{5.1}$$

The adaptation we need to make is to estimate parameters in an online way. We first show how to estimate the transformation online and then how to estimate the copula correlation online in with a given marginal estimate.

5.2.1 Online marginal transformation estimation

In the offline setting, we estimate the transformation $\mathbf{f}$ based on the observed empirical distribution as in Section 2.4.1 and [60]: for $j \in [p]$, using observations in $\mathbf{X}_j$, we construct the estimates as:

$$\hat{f}_j = \hat{F}_j^{-1} \circ \Phi, \quad \hat{f}_j^{-1} = \Phi^{-1} \circ \hat{F}_j. \tag{5.2}$$

where F_j and F_j^{-1} are the empirical CDF and quantile function on the observed entries of the j-th variable. In the online setting, we simply update the observation set as new data comes in for each column $\mathbf{X}_j$. Specifically, we store a running window matrix $\tilde{\mathbf{X}} \in \mathbb{R}^{k\times p}$ which records the k most recent observations for each column, and update $\tilde{\mathbf{X}}$ as new data comes in. The window size is an online learning rate hyperparameter that should be tuned to improve accuracy. A longer window works better when the data distribution is mostly stable but has a few abrupt changes. If the data distribution changes rapidly, a shorter window is needed. Domain knowledge should also inform the choice of window length.

Online datasets may have high autocorrelation, which can improve online imputation. Thus it may be beneficial to allocate different weights to different stored observations and imputing missing entries by empirical weighted quantiles. For example, we can allocate decaying weights for the m stored observations: d^t with $d \in (0, 1]$ for each time lag $t = 1, ..., m$. The decay rate d should be tuned for best performance. This approach interpolates between imputing the last observed value (as $d \to 0$) and the standard Gaussian copula imputation (when $d = 1$).

5.2.2 Online copula correlation estimation

We estimate copula correlation matrix Σ through maximum likelihood estimation (MLE). The offline method Section 2.4.2 applies EM algorithm to find the Σ that maximizes the likelihood value. The key idea of our online estimation is to replace each offline EM iteration with an online EM variant, which incrementally updates the likelihood objective as new data comes in. This online approach does not need to retain all data to perform updates. We first present the offline likelihood objective to be maximized and then show how to update it in the online setting.

Recall from Section 2.4.2, at EM iteration $l + 1$ with estimate Σ^l from iteration l, we need to maximize the objective function as below:

$$Q(\Sigma; \Sigma^l, \{\mathbf{x}^i_{O_i}\}_{i=1}^n) = \frac{1}{n}\sum_{i=1}^n \mathbb{E}[\ell(\Sigma; \mathbf{z}^i, \mathbf{x}^i_{O_i})|\mathbf{x}^i_{O_i}, \Sigma^l, \hat{\mathbf{f}}]. \tag{5.3}$$

The maximizer for Eq. (5.3) is simply the expected "empirical covariance matrix" of the latent variables $\mathbf{z}^i$:

$$\Sigma^{l+1} = \sum_{i=1}^n \frac{1}{n}\mathbb{E}[\mathbf{z}^i(\mathbf{z}^i)^\top|\mathbf{x}^i_{O_i}, \Sigma^l, \hat{\mathbf{f}}]. \tag{5.4}$$

The expectation weights these $\mathbf{z}^i$ by their conditional likelihood value. At last the the obtained estimate is scaled to have unit diagonal to satisfy the copula model constraints: $\Sigma^{l+1} \leftarrow P_{\mathcal{E}}\left(\Sigma^{l+1}\right)$.

Now we show how to adjust and maximize the objective Q in the online setting. When data points come in different batches, i.e. rows S_{t+1} observed at time $t+1$, it is proposed in [18] to update the objective function Q with new rows as:

$$Q_{t+1}(\Sigma) = (1-\gamma_t)Q_t(\Sigma) + \gamma_t Q(\Sigma; \Sigma^t, \{\mathbf{x}^i_{O_i}\}_{i\in S_{t+1}}), \tag{5.5}$$

with $Q_1(\Sigma) = Q(\Sigma; \Sigma^0, \{\mathbf{x}^i_{O_i}\}_{i\in S_1})$ given initial estimate Σ^0 and a monotonically decreasing stepsize $\gamma_t \in (0,1)$. Using Eq. (5.5), we derive a very natural update rule, stated as Lemma 7: in each step we simply take a weighted average of the previous covariance estimate and the estimate we get with a single EM step on the next batch of data. We require the batch size to be larger than the data dimension p to obtain a valid update. One can still make an immediate prediction for each new data point, but to update the model we must wait to collect enough data or use overlapping data batches. If it is crucial to update the model at each new data point, we can use use the most recent n_b data points including the new data point to conduct the model update for some $n_b \geq p$.

Lemma 7. For data batches $\{\mathbf{x}^i\}_{i\in S_1}, ..., \{\mathbf{x}^i\}_{i\in S_t}$ with $\mathbf{x}^i \in \mathbb{R}^p$ and $\min_{l\in[t]} |S_l| > p$, and objective $Q_t(\Sigma)$ as in Eq. (5.5) for $\gamma_t \in (0,1)$. Given a marginal estimate $\hat{\mathbf{f}}$, for $l = 1, \ldots, t$, $\Sigma^l := \mathrm{argmax}_\Sigma\, Q_l(\Sigma)$ satisfies

$$\Sigma^{t+1} = (1-\gamma_t)\Sigma^t + \frac{\gamma_t}{|S_{t+1}|}\sum_{i\in S_{t+1}} \mathbb{E}[\mathbf{z}^i(\mathbf{z}^i)^\top | \mathbf{x}^i_{O_i}, \Sigma^t, \hat{\mathbf{f}}]. \tag{5.6}$$

We also project the resulting matrix to a correlation matrix as in the offline setting. The update takes $O(\alpha p^3|S_t|)$ time with missing fraction α and $|S_t|$ rows. The proof shows that online EM formally requires a weighted update to the expectation computed in the E-step. But for our problem, the parameter Σ, computed as the maximizer (in the M-step), is a *linear* function of the computed expectation (from the E-step). Hence the maximizer also evolves according to the same simple weighted update. A weighted update rule for the parameter fails — leading to divergence — for more general models, when the maximizer is not linear in the expectation, such as for the low-rank-plus-diagonal copula correlation model in Section 3.2.

It is proved [18] an online EM algorithm converges to the stationary points of the KL divergence between the true distribution of the observation π (not necessarily the assumed model) and the learned model distribution, under some regularity conditions. We adapt their result to Theorem 4.

Theorem 4. Let $\pi(\mathbf{x}_O)$ be the distribution function of the true data-generating distribution of the observations and $g_\Sigma(\mathbf{x}_O)$ be the distribution function of the observed data from $\mathrm{GC}(\Sigma, \mathbf{f})$, assuming data is missing uniformly at random (MCAR). Suppose the step-sizes $\gamma_t \in (0, 1)$ satisfy $\sum_{t=1}^{\infty} \gamma_t^2 < \sum_{t=1}^{\infty} \gamma_t = \infty$. Let $\mathcal{L} = \{\Sigma \in S_{++}^p : \nabla_\Sigma \mathrm{KL}(\pi \| g_\Sigma) = 0\}$ be the set of stationary points of $\mathrm{KL}(\pi \| g_\Sigma)$ for a fixed $\mathbf{f}$. Under two regularity conditions on π (see the supplement), the iterates Σ^t produced by online EM (Eq. (5.6)) converge to $\mathcal{L}$ with probability 1 as $t \to \infty$.

The conditions on stepsize γ_t are standard for stochastic approximation methods. If the true correlation Σ generating the data evolves over time, a constant stepsize $\gamma_t \in (0, 1)$ should be used to adapt the estimate to the changing correlation structure. We find using $\gamma_i = c/(i + c)$ with $c = 5$ for the offline setting and $\gamma_i = 0.5$ for the online setting gives good results throughout our experiments. Further tuning over different step size may bring additional gain.

Algorithm 5 Online Imputation with the Gaussian Copula

```
Input: Window size $k$, step size $\gamma_t$ for $t \in [T]$.
 1: Initialize $\Sigma^0$ and running window matrix $\tilde{\mathbf{X}} \in \mathbb{R}^{k\times p}$.
 2: for $t = 1, 2, \ldots, T$ do
 3:     Obtain new data batch $\{\mathbf{x}^i\}_{i\in S_t}$, with $\mathbf{x}^i$ partially observed at $O_i$ and missing
        at $\mathcal{M}_i$.
 4:     Replace the oldest point in $\tilde{\mathbf{X}}_j$ with $x^i_j$ for $j \in O_i, i \in S_t$.
 5:     Estimate marginals $\hat{\mathbf{f}}, \hat{\mathbf{f}}^{-1}$ using $\tilde{\mathbf{X}}$ as in Eq. (5.2) .
 6:     EM step update: obtain $\Sigma^{t+1}$ as in Eq. (5.6).
 7:     Scale to a correlation matrix: $\Sigma^{t+1} = P_{\mathcal{E}}\left(\Sigma^{t+1}\right)$.
 8:     Impute $\hat{\mathbf{x}}^i_{\mathcal{M}_i}$ using $\Sigma^{t+1}$ and $\hat{\mathbf{f}}$ as in Eq. (5.1) for $i \in S_t$.
 9: end for
10: return Imputation $\{\hat{\mathbf{x}}^i_{\mathcal{M}_i}\}_{i\in S_t}$ and $\Sigma^t$ for $t \in [T]$.
```

Online versus offline implementation We may estimate $\hat{\mathbf{f}}$ in Eq. (5.6) either online or offline. The decision entails some tradeoffs. When the storage limit is the main concern, as in the streaming data setting, we can employ the online marginal estimate, storing only a running window and a correlation matrix estimate. We call such an implementation fully *online EM*. When the data marginal distribution evolves over time, it is also important to use *online EM* to forget the old data. On the other hand, when training time is the main concern but the whole dataset is available, the online EM algorithm can be implemented as an offline mini-batch EM algorithm to accelerate convergence. In that setting, the offline marginals are used to provide more accurate and stable estimates as well as to reduce the time for estimating the marginals. We call this implementation *(offline) mini-batch EM*. We present the fully online algorithm in Algorithm 5 with data batches observed sequentially.

5.3 Experiments

The experiments are divided into two parts: online datasets (rows obtained sequentially) and offline datasets (rows obtained simultaneously). The online setting examine the ability of our methods to learn the changing distribution of the steaming data. The offline setting evaluate the speedups and the potential accuracy lost due to minibatch training and online marginal estimation compared to offline EM. See the appendix for more experiments under different data dimension, missing ratio and missing mechanisms.

Algorithm implementation we implement the *offline EM* algorithm Chapter 2, the minibatch EM with online marginal estimate denoted by *online EM*, and the minibatch EM with offline marginal estimate denoted by *minibatch EM*. For imputation comparison, we implement GROUSE [7] and online KFMC [31]. For fair comparison, we use 1 core for all methods, but report the acceleration brought by parallelism for all Gaussian copula methods in the appendix. All experiments use a laptop with a 3.1 GHz Intel Core i5 processor and 8 GB RAM. All EM methods are implemented using Python. We implement GROUSE and online KFMC using the authors' provided Matlab codes at `https://web.eecs.umich.edu/~girasole/grouse/` and `https://github.com/jicongfan/Online-high-rank-matrix-completion`.

Tuning parameters selection

we do not use tuning parameter for offline EM and minibatch EM. We use 1 tuning parameter for online EM: the window size m for online marginal estimates, 2 tuning parameters for GROUSE, the rank and the step size, and 2 tuning parameters for online KFMC, the rank in a latent space and the regularization parameter.

For all methods, we use grid search to choose the tuning parameters. The window size of online EM is selected from $\{50, 100, 200\}$ for online experiments and fixed as 200 for offline experiments. The constant c in step size c/t is selected from $\{0.1, 1, 10\}$ for GROUSE on offline experiments. The constant step size c is selected from $\{10^{-8}, 10^{-4}, 10^{-2}\}$ for GROUSE on online experiments. The rank is selected from $\{1, 5, 10\}$ for GROUSE on all experiments. The rank is selected from $\{200, 300, 400\}$ and the regularization parameter is selected from $\{0.1, 0.01, 0.001\}$ for online KFMC on all experiments. Online KFMC also requires a momentum update, for which we take the author's suggested value .5.

We note one other issue. For online algorithms, it is typical to choose hyperparameters during an initial "burn-in" period. For example, in GROUSE, choosing the step-size from initial data can result in divergence later on as the data distribution changes. As a result, the maximum number of optimization iterations is also difficult to choose: the authors' default settings are often insufficient to give good performance, while allowing too many iterations may lead to (worse) divergence. We will report and discuss an example of divergence in our online real data experiment.

Computational complexity For a new data vector in $\mathbb{R}^p$ with k observed entries, GROUSE has the smallest computation time $O(pr_g + kr_g^2)$ with rank $r_g < p$; online EM comes second with computation time $O(k^3 + k(p-k)p)$; online KFMC has the largest computation time $O(r_k^3)$ with $r_k > p$.

5.3.1 Offline synthetic experiment

We construct a dataset consisting of 6000 i.i.d. data points drawn from a 15-dimensional Gaussian Copula, with 5 continuous (exponential distribution with parameter 1/3), 5 ordinal with 5 levels, and 5 binary entries, as in Section 2.5.1. The cut points for generating the ordinal and binary entries are randomly selected. We randomly mask 40% entries as missing: approximately 2 out of 5 entries for each data type are masked. We generate independent two identical and independent datasets: one for choosing the tuning parameters, the other for training and evaluating the performance. For GROUSE, the selected rank is 1 and the selected constant c in decaying stepsize c/t is 1. For online KFMC, the selected rank is 400 and the selected regularization is .1. The used batch size is 100 for all methods.

Shown in Table 5.1, the minibatch and online variants of the EM algorithm converge substantially faster than offline EM and provide similar imputation accuracy. The results are especially remarkable for online EM, which estimates the marginals using only 200 points. The minibatch variant is three times faster than offline EM with the same accuracy. All EM methods outperform online KFMC and GROUSE, and even median imputation outperforms GROUSE. Interestingly, the best rank for GROUSE is 1. The results here show LRMC methods fit poorly for long skinny datasets, although the selected best rank, 1, misleadingly indicates the existence of low rank structure.

Table 5.1: Mean(sd) for runtime, imputation error of each data type for synthetic offline data over 10 trials.

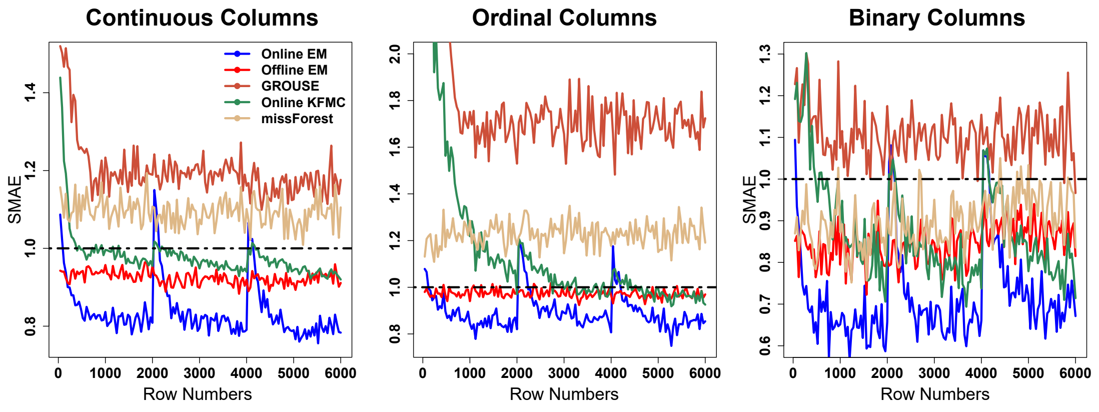
Method	Runtime (s)	Continuous	Ordinal	Binary
Offline EM	187.7(0.8)	0.79(.04)	0.84(.03)	0.63(.07)
Minibatch EM	48.2(0.5)	0.79(.04)	0.83(.03)	0.63(.07)
Online EM	54.5(3.4)	0.80(.04)	0.84(.02)	0.63(.07)
Online KFMC	79.6(1.6)	0.92(.03)	0.92(.02)	0.67(.08)
GROUSE	7.7(.3)	1.17(.03)	1.67(.05)	1.10(.07)

5.3.2 Online synthetic experiment

Now we consider streaming data from a changing distribution. To do this, we generate and mask the dataset similar to Section 5.3.1, but set two change points at which a new correlation matrix is chosen: $\mathbf{x}^1, \ldots, \mathbf{x}^t \sim \mathrm{GC}(\Sigma_1, \mathbf{f})$, $\mathbf{x}^{t+1}, \ldots, \mathbf{x}^{2t} \sim \mathrm{GC}(\Sigma_2, \mathbf{f})$ and $\mathbf{x}^{2t+1}, \ldots, \mathbf{x}^{3t} \sim \mathrm{GC}(\Sigma_3, \mathbf{f})$, with $t = 2000$. We repeat the experiment 10 times. For each repetition, we randomly sample Σ_1, Σ_2 and Σ_3 and fix $\mathbf{f}$, as in the offline experiments. We also generate independent two identical and independent datasets: one for choosing the tuning parameters, the other for training and evaluating the performance. We implement all online algorithms from a cold start and make only one pass through the data, to mimic the streaming data setting. For comparison, we also implement offline EM and missForest [85] (also offline) and allow them to make multiple passes. For online EM, the selected window size is 200. For GROUSE, the selected rank is 1 and the selected constant stepsize is 10^{-6}. For online KFMC, the selected rank is 200 and the selected regularization is .1. The used batch size is 40 for all methods.

Shown in Fig. 5.1, online EM clearly outperforms the offline EM on average, by learning the changing correlation. Online EM has a sharp spike in error as the correlation abruptly shifts, but the error rapidly declines as it learns

Figure 5.1: Mean imputation error and change point tracking statistics over 10 trials for online synthetic datasets. Each point stands for an evaluation over a data batch of 40 points.



the new correlation. Both online EM and online KFMC outperform missForest. Surprisingly, online KFMC cannot even outperform offline EM, which is only able to impute using a single correlation estimate for all data points. GROUSE performs even worse in that it cannot outperform median imputation as in the offline setting. The results indicate online imputation methods can fail to learn the changing distribution when their underlying model does not fit the data well.

5.3.3 Offline real data experiment

To further show the speedup of the minibatch algorithms, we evaluate on a subset of the MovieLens 1M dataset [40] that consists of all movies with more than 1000 ratings, with 1-5 ordinal ratings of size 6939×207 with over 75% entries missing. We divide all available entries into training (80%), validation (10%) and testing (10%). For GROUSE, the selected rank is 5 and the selected constant c in decaying step size c/t is 1. For online KFMC, the selected rank is 200 and the selected regularization parameter is .1. The used batch size is 121

Table 5.2: Mean(sd) for runtime and imputation error on a subset of MovieLens1M data over 10 trials.

Method	Runtime (s)	MAE	RMSE
Offline EM	1690(9)	0.583(.002)	0.883(.004)
Minibatch EM	252(2)	0.585(.003)	0.886(.003)
Online EM	269(3)	0.590(.002)	0.890(.003)
Online KFMC	176(21)	0.631(.005)	0.905(.006)
GROUSE	27(2)	0.634(.003)	0.933(.004)

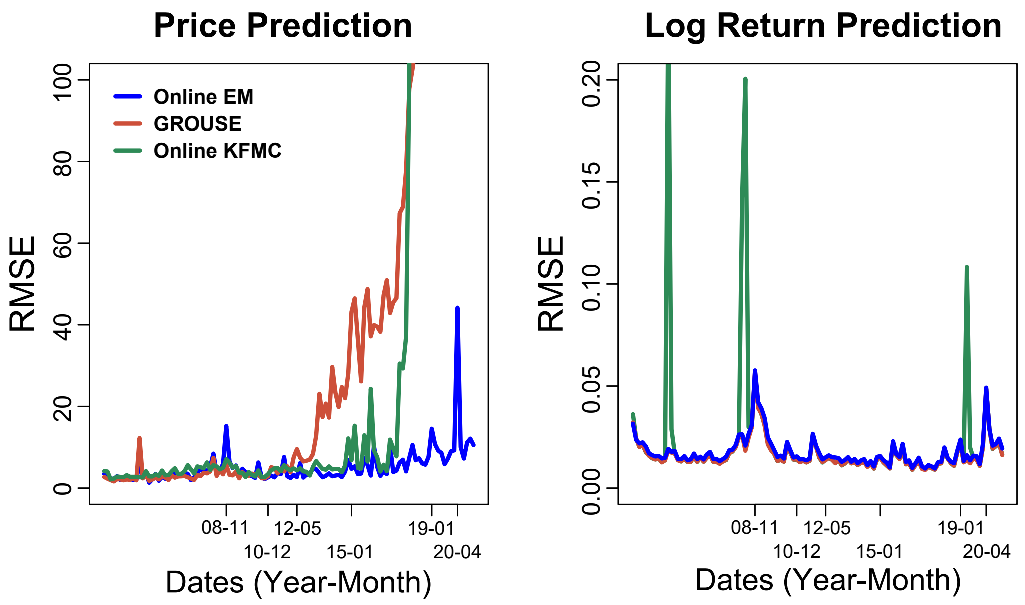
for all methods.

Table 5.2 shows that the minibatch and online EM still obtain comparable accuracy to the offline EM. The minibatch EM is around 7 times faster than the offline EM. All EM methods significantly outperform online KFMC and GROUSE. Interestingly, as the dataset gets wider, online KFMC loses its advantage over GROUSE. The results here indicate the nonlinear structure learned by online KFMC fails to provide better imputation than the linear structure learned by GROUSE. In contrast, the structural assumptions of our algorithm retain their advantage over GROUSE even on wider data.

5.3.4 Online real data experiment

We now evaluate online missing data imputation on the daily prices and returns of 30 stocks currently in the Dow Jones Industrial Average (DJIA) across 5030 trading days. Three stocks have missing entries (90.6%, 16.9% and 35.6%) corresponding to dates before the stock was publicly traded. We construct a dataset of size 5029×60, where the first 30 columns store yesterday's price (or log return) and last 30 columns store today's. We scan through the rows, making online imputations. Upon reaching the t-th row, the first $t-1$ rows and

Figure 5.2: The imputation error for the DJIA daily price (left) and the DJIA daily log-returns (right), averaged over all stocks. Each point stands for an evaluation over a time interval of 40 points.



the first 30 columns of the t-th row are completely revealed, while the last 30 columns of the t-th row are to be predicted and thus masked. Once the prediction is made and evaluated, the masked entries are revealed to update the model parameters. such dataset allows the imputations methods can learn both the dependence among different stocks and the auto-dependence of each stock. We use first 400 days' data to train the model and the next 400 days' data as a validation set to choose the tuning parameters, and all remaining data to evaluate the model performance. For online EM, the selected window size is 50 for price prediction and 200 for log return prediction. For GROUSE, the rank and the constant step size are selected as $\{10, 10^{-6}\}$ for price prediction, and $\{1, 10^{-6}\}$ for log return prediction. For online KFMC, the rank and the regularization are selected as $\{300, 0.1\}$ for both the price prediction and the log return prediction. The used batch size is 40 for all methods.

In Fig. 5.2, the left plot shows that all methods predict prices well early on, but GROUSE and online KFMC both diverge eventually. GROUSE starting from around 2013 and online KFMC starting from around 2017. In contrast, online EM has robust performance throughout. Although the imputation error peaks around the start of 2020, online EM is able to quickly adjust to the changing distribution: the imputation error quickly falls back. Thus online EM stands out in that it obviates the need of online hyperparameter selection to have stable performance. The right plot shows that online EM and GROUSE perform similarly on log returns: their error curves almost overlap each other. Online KFMC underperforms: it makes large errors more often. We conjecture GROUSE and online KFMC perform better on the log returns than on the price data because the scale of the data is stable, so that hyperparameters chosen early on still exhibit good performance later. The good performance of GROUSE indicates the asset log returns are approximately low rank, Still, online EM is robust to different (even changing) marginal data distributions and performs well on approximately low rank data.

5.4 Discussion

This chapter develops an algorithm to incrementally update on the Gaussian copula model using mini batch of data. One important future direction would extend the online Gaussian copula estimation to wide datasets, using low rank Gaussian copula Chapter 3, to ensure the computational complexity scales linearly in the number of observations.

CHAPTER 6

ONLINE DEPENDENCE CHANGE POINT DETECTION

This chapter develops a new method to detect change points in the multivariate dependence structure in online data with missing values, by fitting a Gaussian copula model and tracking the magnitude of the copula correlation update. It is based on Zhao et al. [105].

6.1 Introduction

A common interest for online data (or time series) is change point detection: does the data distribution change abruptly, and can we pinpoint when the change occurs? While there are many different types of temporal changes, we focus on changes in the dependence structure of the data, a crucial issue for many real world applications. For example, classic Markowitz portfolio design uses the dynamic correlation structure of exchange rates and market indexes to design a portfolio of assets that balances risk and reward [61].

Our contribution

Here we detect changes in the dependency structure of online long skinny mixed data, based on the online estimation of Gaussian copula model introduced in Chapter 5. Concretely, we propose a Monte Carlo test for dependence structure change detection *at any time*. The method tracks the magnitude of the copula correlation update and reports a change point when the magnitude exceeds a threshold. Inheriting the advantages of the Gaussian copula model, our proposed online change point detection algorithm naturally handles long skinny mixed data with missing values.

Related work

Change point detection (CPD) is an important topic with a long history. See [2] for an expansive review. Online CPD seeks to identify change points in real-time, before seeing all the data. Missing data is also a key challenge for CPD: most CPD algorithms require complete data. The simplest fix for this problem, imputation followed by a complete-data CPD method, can hallucinate change points due to the changing missingness structure or imputation method used. Our proposed method avoids these difficulties. Another workaround, Bayesian online CPD methods [1, 33], can fill out the missing entries by sampling from its posterior distribution given all observed entries.

6.2 Monte Carlo test for change point detection

We first outline the CPD problem in the context of the Gaussian copula model. Consider a sequence of incomplete mixed data observations $\mathbf{x}^1, \ldots, \mathbf{x}^T \sim \mathrm{GC}(\Sigma, \mathbf{f})$, where $\mathbf{x}^i$ is observed at locations O_i for $i \in [T]$. We wish to identify whether there is a change point t_0 — a time when the copula correlation Σ changes substantially — and if so, when this change occurs. We formulate the single CPD problem as the following hypothesis test, for fixed t_0:

$\mathbf{x}^1, \ldots, \mathbf{x}^{t_0} \sim \mathrm{GC}(\Sigma, \mathbf{f})$, and $\mathbf{x}^{t_0+1}, \ldots, \mathbf{x}^T \sim \mathrm{GC}(\tilde{\Sigma}, \mathbf{f})$,

$$\mathrm{H}_0 : \tilde{\Sigma} = \Sigma \text{ versus } \mathrm{H}_1 : \tilde{\Sigma} \neq \Sigma. \tag{6.1}$$

We assume time-invariant marginal $\mathbf{f}$. In practice, it suffices for $\mathbf{f}$ to be stable in a small local window. The latent correlation matrix changes, reflecting the changing dependence structure. To detect a change-point, a test statistic is computed for each point to measure the deviation of new points from old distribution. A change is detected if the test statistic exceeds a certain threshold. We consider the online detection problem instead of a retrospective analysis with all data available. Specifically, to test whether a change occurs at time t_0, we may use only the data $\mathbf{x}^{t_0+1}, \ldots, \mathbf{x}^T$ for a small window length $T - t_0$ and the fitted model at time t_0.

To derive a test statistic, notice that $\Sigma^{-1/2}\tilde{\Sigma}\Sigma^{-1/2} = \mathrm{I}_p$ under H_0. Thus for some matrix norm h, we use the matrix distance $d(\Sigma, \tilde{\Sigma}; h) = h(\Sigma^{-1/2}\tilde{\Sigma}\Sigma^{-1/2} - \mathrm{I}_p)$ to measure the deviation of new points from old distribution. While Σ and $\tilde{\Sigma}$ are unknown, we replace them with the estimates Σ^{t_0} and Σ^T, generated by the EM iteration up to time t_0 and time T, respectively. Thus we construct our test statistic as $d(\Sigma^{t_0}, \Sigma^T; h)$: large values indicate high probability of a change point. Experimentally, we find that different choices of h give very similar trends. Hence below we report results using the Frobenius norm as h, to reduce computation.

The change point is detected when $d(\Sigma^{t_0}, \Sigma^T; h)$ exceeds some threshold b_α, which is chosen to control the false-alarm rate α. Calculating b_α analytically requires the asymptotic behaviour of the statistic under the null distribution, which is generally intractable including our case. We use Monte Carlo (MC) methods to simulate the null distribution of our test statistic and select the threshold. This method is similar to the permutation test for CPD [63]. We present our test for the hypothesis in Eq. (6.1) as Algorithm 6 . Notice comparing $d(\Sigma^{t_0}, \Sigma^T; h)$ to b_α is equivalent to comparing the returned empirical p-value with the desired false-alarm rate α. See [27, 69] for the use of empirical p-values. In practice, α can be regarded as a hyperparameter to tune the false positive/negative rate.

Algorithm 6 Monte Carlo test for Gaussian copula correlation change point detection

```
Input: New data $\{\mathbf{x}^i\}_{i=t_0+1}^T$, the number of samples $B$, estimated model $\Sigma^{t_0}, \Sigma^T$ and
    $\mathbf{f}^{t_0}$.
1: Compute the test statistic $s = d(\Sigma^{t_0}, \Sigma^T)$.
2: for $j = 1, 2, \ldots, B$ do
3:     Sample $\mathbf{y}^i \sim \mathrm{GC}(\Sigma^{t_0}, \mathbf{f}^{t_0})$ and mask $\mathbf{y}^i$ at where $\mathbf{x}^{i+t_0}$ is missing for $i = 1, ..., T-t_0$.
4:     Update the model at $t_0$ with new points $\{\mathbf{y}^i\}_{i=1}^{T-t_0}$ using Eq. (5.6).
5:     Compute $s_j = d(\Sigma^{t_0}, \Sigma^{T,j})$ with the updated correlation $\Sigma^{T,j}$.
6: end for
Output: The p-value $(|\{s_j : s \le s_j\}| + 1)/(B + 1)$.
```

6.3 Sequential multiple change points detection

So far we have introduced a method to determine whether a change point occurs at a time t_0 using the new data $\{\mathbf{x}^i\}_{i=t_0+1}^T$. In the online setting, we will seek to detect whether a change point occurred at the start of any of the time intervals $S_1, \ldots, S_T$ using the corresponding data batches $\{\mathbf{x}^i\}_{i\in S_1}, \ldots, \{\mathbf{x}^i\}_{i\in S_T}$. (Notice that we can detect a change point at any time by using overlapping time windows.) To do so, we can apply our MC test to every new data batch $\{\mathbf{x}^i\}_{i\in S_t}$ for $t \in [T]$. Unfortunately, controlling the significance level for each test still yields too many false positives due to the number of tests T. Instead, we use online

FDR control methods [74, 50, 75] to control the FDR to a specified level across the whole process. At each time point, the decision as to whether a change point occurs is made by comparing the obtained p-values from the MC test with the allocated time-specific significance level, which only depends on previous decisions. We summarize the sequential algorithm in Algorithm 7.

Algorithm 7 Online change point detection via Gaussian copula

```
Input: Online FDR control algorithm $\mathcal{A}$, time windows $S_1, \ldots, S_T$, the FDR level
    $\alpha$, and an initial model estimate $\Sigma^0$ and $\mathbf{f}^0$.
1: for $t = 1, 2, \ldots, T$ do
2:     Obtain new data batch $\{\mathbf{x}^i\}_{i \in S_t}$
3:     Update the model estimate as $\Sigma^t$ and $\mathbf{f}^t$ as in Algorithm 6.
4:     Obtain the MC p-value $p_t$ with new data $\{\mathbf{x}^i\}_{i \in S_t}$, $\Sigma^{t-1}, \Sigma^t$ and $\mathbf{f}^{t-1}$ as input.
5:     Obtain the significance $\alpha_t = \mathcal{A}(\{R_1, \ldots, R_{t-1}\})$.
6:     Set the binary decision $R_t = 1$ if $p_t < \alpha_t$ and 0 otherwise.
7: end for
Output: Decisions $\{R_t\}_{t \in [T]}$ and p-values $\{p_t\}_{t \in [T]}$.
```

If the time intervals are very short, it can be useful to set a burn-in period to estimate the new correlation matrix before attempting to detect more change points in order to prevent false positives. Our method as stated is designed to detect change points quickly by using the model at the end of the time interval as an estimate for the new model $\tilde{\Sigma}$. Instead, one could wait longer to compute a better estimate of $\tilde{\Sigma}$ to assess whether a change point had occurred in an earlier interval. This approach detects change points slower and requires more memory than our approach, but could deliver higher precision.

We point out an important practical concern: online FDR control methods often allocate very small significance levels ($< 10^{-4}$) in practice [79], while the smallest p-value that the MC test with B samples can output is $1/(B+1)$. Under the null that no change point happens, the probability that the test statistics s is larger than all $\{s_j\}_{j \in [B]}$ computed from MC samples is $1/(B+1)$. Thus only when B is very large ($> 10^4$) can a MC test possibly detect a change point. Setting B this large is usually computationally prohibitive. One ad-hoc remedy is to use the biased empirical p-value $\#\{: s \leq s_j\}/(B+1)$, which can output a p-value of 0; however, this approach is equivalent to choose all significance levels $\alpha_t \in (\frac{1}{B+1}, \frac{2}{B+1})$. Developing a principled approach to online FDR control that can target less conservative significance levels (higher power) in the context of online CPD constitutes important future work.

6.4 Experiments

Here we examine the ability of our method to detect the changing distribution in online data, using the same data generating setting as in Section 5.3.2.

Implementation details

We also implement the online Bayesian change point detection (BOCP) algorithm [1], one of the best performing CPD method according to a recent evaluation paper [94]. The norm of subspace fitting residuals for GROUSE [7] can also serve for CPD: a sudden peak of large residual norm indicates abrupt changes. Although it lacks a formal criterion to conduct CPD on the residuals, it provides an intuitive visual illustration of the changes of the online model. We compare our test statistic, defined in Algorithm 6, with the residual norms from GROUSE, to see which identifies change points more accurately.

We select tuning parameters for online EM and GROUSE in the same way as described in Section 5.3. BOCP requires 4 hyperparameters for its priors and its hazard function [1]. We choose them with a uniform random search [8]. BOCP is implemented using the R package `ocp` [71].

Results

Shown in Fig. 6.1, our correlation deviation statistic provides accurate prediction for change points, while the residual norms from GROUSE remains stable after the burn-in period for model training, which verifies GROUSE cannot adapt to the changing dependence here (stated in Section 5.3.2).

Show in Fig. 6.2, online EM successfully detects both change points in all repetitions. In fact, the algorithm detects a change point (of decreasing magnitude) during several batches after each true change, showing how long it takes to finally learn the new dependence structure. To avoid the repeated false alarms, one could set a burn-in period following each detected change point. In contrast, BOCP only reports one false discovery, showing its inability to detect the changing dependence structure.

6.5 Discussion

We presented an online change point detection method using Gaussian copula for long skinny mixed datasets. Developing a method to identify changes in the marginal distribution and a less conservative online FDR control method that supports online change point detection by utilizing the dependence structure of sequential tests are important future work.

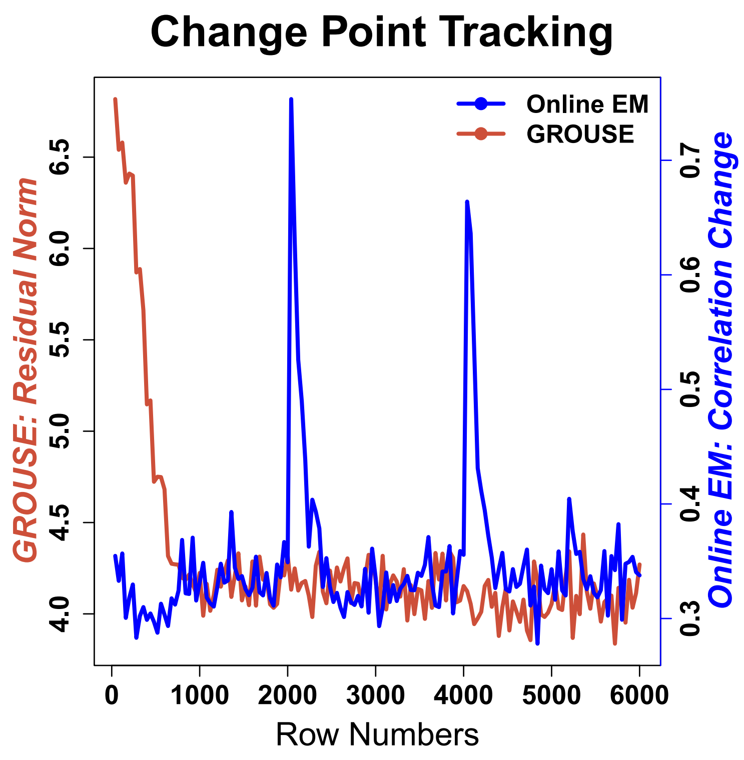


Figure 6.1: Mean change point tracking statistics over 10 trials for online synthetic datasets. Each point stands for an evaluation over a data batch of 40 points.

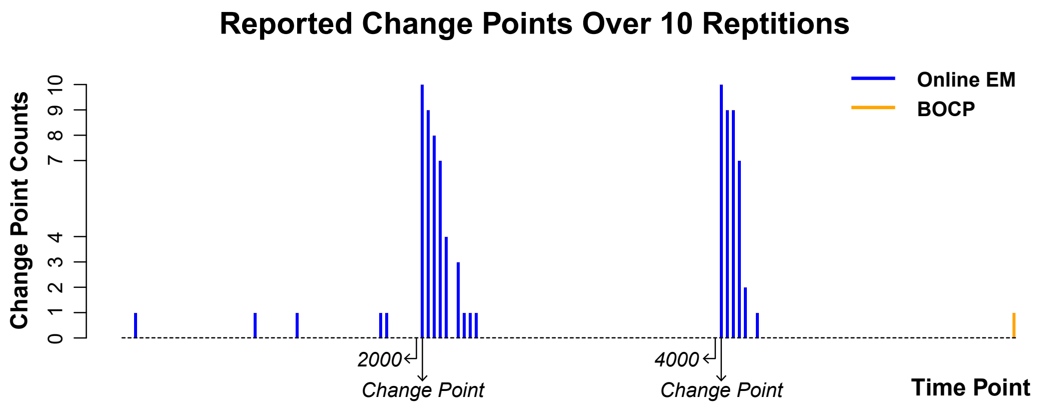


Figure 6.2: Change points from online EM detection (ours) and BOCP over 10 trials in online synthetic experiments. Each bar stands for a decision over a data batch of 40 points.

CHAPTER 7

EXTENDING GAUSSIAN COPULA TO HANDLE CATEGORICAL DATA

In this chapter, we show how to extend the Gaussian copula model, previously defined in Section 2.2, to further handle categorical variables. This chapter is our most recent work, and has not yet been made public outside of this dissertation.

7.1 Introduction

Missing data imputation is the first critical step of many data analysis pipelines, since missing entries are abundant in modern datasets and most machine learning algorithms require complete data input. The imputation task is challenging with categorical variables, which take values from unordered categories. Categorical variables frequently appear in real world dataset, often alongside ordered variables including continuous and ordered categorical variables. It is crucial for an imputation algorithm to be able to impute both categorical and ordered variables using all observation accurately and efficiently. While there has been many missing data imputation algorithms, not many of them take categorical variables into consideration. The choices for imputing categorical and ordered variables are even fewer.

A sensible model treats categorical data as generated by transforming a latent continuous vector, using the softmax operator or argmax operator. There are several advantages for this continuous representation. First, the number of possible observation grows exponentially with vector length for a categorical vector, and many of them may never be observed in practice. Using a latent continuous vector, the number of free parameters we need is often just quadratic with vector length, and thus addresses the sparsity problem. Second, we can model the interaction between categorical and continuous variables now using methods for continuous vectors, which are rich. Here our work models a categorical variable as the argmax of a latent Gaussian vector. Our work further models a categorical vector by concatenating the latent Gaussian vector for each variable. The choice of Gaussianity is inspired by the Gaussian copula model [103], which models a mixed data vector with all ordered variables as a elementwisely transformed Gaussian vector. Combining our proposition and the Gaussian copula, we propose the *extended Gaussian copula* for categorical and ordered variables mixed data vector. The model associates each categorical variable with a latent normal vector and each ordered variable with a latent normal scalar. Each variable is obtained by transforming its associated latent Gaussian. The data dependence structure is captured by latent Gaussian correlations.

Contribution

We propose a joint distribution model for categorical and ordered variables mixed data as transformed Gaussian, the *extended Gaussian copula*. The ordered variables may be continuous, binary, ordinal or truncated. As a special case, our proposition models multivariate categorical vectors. Same as the Gaussian copula, the extended Gaussian copula does not require any model hyperparameter and makes no assumption for the data marginal distribution.

We propose fitting algorithms for our *extended Gaussian copula*, which consists of two steps. The first step solves for the latent Gaussian mean vector so that its argmax has exactly the same distribution as the desired categorical variable. The second step reduces the dependence structure estimation to that of the Gaussian copula for ordered variables only which is already known [103].

We develop imputation methods for both missing categorical and missing ordered variables using all available observation within our extended Gaussian copula. The developed imputation methods include both single imputation and multiple imputation and can be used for online imputation.

Related work

Modeling a categorical variable as the argmax of a latent continuous vector is a classical technique appearing in the multinomial probit model [13, 65]. Our proposition greatly differs from the multinomial probit model in that it does not require explanatory variables. Instead, our work is more closely related to [23], which firstly proposed to extend the Gaussian copula to accommodate categorical variables, inspires While our models are very similar, the model in [23] has redundant parameters and thus is unidentifiable. We instead have an identifiable model after removing redundant parameters. Moreover, we firstly show that the extended Gaussian copula is capable of modeling arbitrary categorical variable and provide a very accurate marginal fitting algorithm.

Notation For a function $f(x)$ from $\mathbb{R}^K$ to $\mathbb{R}$ where $K \in \mathbb{N}$, we define $f^{-1}(y)$ as the set $\{x : f(x) = y\}$. Here is one exception: for 1D cumulative distribution function (CDF) $F(x)$, we define $F^{-1}(y) = \inf\{x \in \mathbb{R} : F(x) \geq y\}$. We use $\mathrm{argmax}(\mathbf{z})$ to denote the index of the maximum in vector $\mathbf{z} = (z_1, \ldots, z_p)$: $\mathrm{argmax}_{l=1,\ldots,p}(z_1, \ldots, z_p)$.

7.2 Categorical variables as transformed Gaussian

In this section, we first show how to model a categorical variable by transforming a latent Gaussian vector. Then we extend it to model the joint distribution of a categorical vector and further the joint distribution of a mixed data vector with both categorical and ordered variables. To ease the notation, we assume that all categorical variables take value from K categories encoded as $\{\text{“}1\text{”}, \ldots, \text{“}K\text{”}\}$. It is straightforward to allow categorical variables have different number of categories.

7.2.1 Univariate categorical variable

We model a univariate categorical variable x with K categories as the argmax of a K-dim latent Gaussian $\mathbf{z} = (z_1, \ldots, z_K)$ with some mean $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_K)$ and identity covariance. That is,

$$x := \operatorname{argmax}(\mathbf{z} + \boldsymbol{\mu}), \qquad \text{where } \mu_1 = 0, \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_K). \tag{7.1}$$

We restrict $\mu_1 = 0$, because only the difference among entries of $\mathbf{z}$ matters for the resulting argmax's distribution. The first dimension, z_1, serves as a *base dimension*. While a dense covariance matrix for $\mathbf{z}$ is sometimes used [23], we show in Theorem 5 that it is unnecessary, because the model in Eq. (7.1) is capable of modeling arbitrary categorical variable and identifiable.

Theorem 5 (Existence and Uniqueness)**.** For a categorical distribution with probability mass function $\mathbf{P}(x = \text{“}k\text{”}) = p_k > 0$ for $k = 1, \ldots, K$ and $\sum_{k=1}^{K} p_k = 1$, there exists a unique $\boldsymbol{\mu} \in \mathbb{R}^K$ such that:

$$\mathbf{P}(\text{argmax}(\mathbf{z} + \boldsymbol{\mu}) = k) = p_k, \quad \text{where } \mu_1 = 0, \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_K), \tag{7.2}$$

for $k = 1, \dots, K$.

Given a categorical variable, algorithm to estimate $\boldsymbol{\mu}$ is described later in Section 7.4.1.

7.2.2 Multivariate categorical vector

We now extend Eq. (7.1) to model a categorical vector $\mathbf{x} = (x_1, \dots, x_p)$ is straightforward. We simply concatenate the latent normal vectors corresponding to different categorical variables into one vector. While we impose identity covariance assumption on the latent normal vector for each categorical variable, we allow the latent normal variables corresponding to different categorical variables to be correlated. However, the base dimension is still restricted to not correlate with any other dimension. That is, we use the following model:

$$\begin{aligned} &\mathbf{z} = (\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(p)}) \sim \mathcal{N}(\mathbf{0}, \Sigma), \\ &x_j = f_j(\mathbf{z}^{(j)}; \boldsymbol{\mu}^{(j)}) := \text{argmax}(\mathbf{z}^{(j)} + \boldsymbol{\mu}^{(j)}) \text{ with } \mu_1^{(j)} = 0, \end{aligned} \tag{7.3}$$

and $\Sigma \in \mathbb{R}^{pK \times pK}$ satisfies that

$$\Sigma_{[j],[j]} = \mathbf{I}_K \text{ and } \Sigma_{[j]_1, -[j]_1} = \mathbf{0}, \text{ where } [j] = \{(j-1)K + l | l = 1, \dots, K\}. \tag{7.4}$$

for $j = 1, \ldots, p$. We use $\Sigma_{I,J}$ to denote the submatrix of Σ with rows in I and columns in J. A negative set $-I$ means all the index excluding those in I. Note (1) $[j]$ denotes the index of $\mathbf{z}^{(j)}$ in $\mathbf{z}$, and we restrict $\Sigma_{[j],[j]} = \mathbf{I}_K$ according to Eq. (7.1); (2) $[j]_1$ denotes the index of $z_1^{(j)}$, the base dimension of $\mathbf{z}^{(j)}$, in $\mathbf{z}$, and we restrict $\Sigma_{[j]_1,-[j]_1} = \mathbf{0}$ to enforce zero correlation between a base dimension and any other dimension.

While Σ introduces dependence structure among $\mathbf{x}$, the value of $\boldsymbol{\mu}^{(j)}$ solely determines the marginal distribution of x_j, because the marginal distribution of $\mathbf{z}^{(j)}$ is $\mathcal{N}(\mathbf{0}, \mathbf{I}_K)$ and independent of Σ. The ability to separate marginal distribution from dependence structure is very similar to that in the Gaussian copula for ordered variables [103].

7.2.3 Mixed data with categorical and ordered variables

In this section, we extend the Gaussian copula model for ordered variables to model mixed data containing both categorical and ordered variables. Note our proposition for categorical variables in Eq. (7.3) shares the same form as $\mathbf{x} = \mathbf{f}(\mathbf{z})$ in the Gaussian copula, except that a categorical variable x_j is generated using a normal vector $\mathbf{z}^{(j)}$ instead of a normal scalar z_j. Combining the two latent models together, we propose the following extended Gaussian copula model for mixed data with both categorical and ordered variables.

Definition 5 (Extended Gaussian copula)**.** For a mixed data vector $\mathbf{x} = (\mathbf{x}_{\text{cat}}, \mathbf{x}_{\text{ord}})$ where $\mathbf{x}_{\text{cat}}$ is a p_{cat}-dim categorical vector and $\mathbf{x}_{\text{ord}}$ is a p_{ord}-dim vector admitting marginal CDF, we say $\mathbf{x}$ follows the extended Gaussian copula $\mathbf{x} \sim \text{EGC}(\Sigma, \mathbf{f}^{\text{ord}}, \boldsymbol{\mu})$, if there exists a correlation matrix Σ, an elementwise monotone

$\mathbf{f}^{\text{ord}}$ and $\boldsymbol{\mu}$ such that for $\mathbf{z} = (\mathbf{z}_{\text{cat}}, \mathbf{z}_{\text{ord}}) \sim \mathcal{N}(\mathbf{0}, \Sigma)$,

$$\mathbf{x}_{\text{cat}} = \mathbf{f}^{\text{cat}}(\mathbf{z}_{\text{cat}}; \boldsymbol{\mu}) = (f_1^{\text{cat}}(\mathbf{z}^{(1)}; \boldsymbol{\mu}^{(1)}), \ldots, f_{p_{\text{cat}}}^{\text{cat}}(\mathbf{z}^{(p_{\text{cat}})}; \boldsymbol{\mu}^{(p_{\text{cat}})})), \quad \text{for } \mathbf{z}_{\text{cat}} = (\mathbf{z}^{(1)}, \ldots, \mathbf{z}^{(p_{\text{cat}})}),$$
$$\mathbf{x}_{\text{ord}} = \mathbf{f}^{\text{ord}}(\mathbf{z}_{\text{ord}}) = (f_1^{\text{ord}}(z_1), \ldots, f_{p_{\text{ord}}}^{\text{ord}}(z_{p_{\text{ord}}})), \quad \text{for } \mathbf{z}_{\text{ord}} = (z_1, \ldots, z_{p_{\text{ord}}}).$$

where $f_j^{\text{cat}}(\cdot; \boldsymbol{\mu}^{(j)})$ has the form as in Eq. (7.3) and Σ satisfies the constraints in Eq. (7.4) (assuming $\mathbf{z}_{\text{cat}}$ appears before $\mathbf{z}_{\text{ord}}$ in $\mathbf{z}$).

Compared to the original Gaussian copula, there are a few key differences. First, the latent $\mathbf{z}$ is now longer than the data $\mathbf{x}$. Second, there now exists vector to scalar transformation functions. Third, previously unrestricted copula correlation Σ now has restrictions. For simplicity, we also write $\mathbf{x} = \mathbf{f}(\mathbf{z}) = (\mathbf{f}^{\text{cat}}(\mathbf{z}^{\text{cat}}; \boldsymbol{\mu}), \mathbf{f}^{\text{ord}}(\mathbf{z}^{\text{ord}}))$ and use $\mathbf{f}$ to refer to all transformation parameters, i.e. $(\boldsymbol{\mu}, \mathbf{f}^{\text{ord}})$.

7.3 Missing data imputation

In the extended Gaussian copula, we want to impute the missing entries using all the observed entries. Here we first show how to impute missing values with known model parameters. For model parameter estimation, see Section 7.4.

Note each categorical x_j corresponds to K consecutive entries in $\mathbf{z}$, i.e. $\mathbf{z}^{(j)}$ that generates x_j. When we use a set I to index $(p_{\text{cat}} + p_{\text{ord}})$-dim $\mathbf{x}$, we use $[I]$ to denote the corresponding entries in $(p_{\text{cat}}K + p_{\text{ord}})$-dim $\mathbf{z}$. For example, $[j]$ in Eq. (7.4) follows this notation with $I = \{j\}$. We will use two special sets of I: O and $\mathcal{M}$ as the observed and the missing entries of $\mathbf{x}$, respectively. Thus $[O]$ and

$[\mathcal{M}]$ are the latent dimensions in $\mathbf{z}$ that generates $\mathbf{x}_O$ and $\mathbf{x}_\mathcal{M}$, respectively.

We use the same imputation strategy used in [103]. That is, the imputation of $\mathbf{x}_\mathcal{M}$ follows the order of

$$\mathbf{x}_O \xrightarrow{\mathbf{f}_O^{-1}} \mathbf{z}_{[O]} \xrightarrow{\Sigma} \mathbf{z}_{[\mathcal{M}]} \xrightarrow{\mathbf{f}_\mathcal{M}} \mathbf{x}_\mathcal{M}.$$

In other words, we first translate the observed information into the latent space, then impute the latent dimensions utilizing that $\mathbf{z}_{[\mathcal{M}]}|\mathbf{z}_{[O]}$ is Gaussian distributed, and then transform the imputed latent variables back to the data space.

Multiple imputation. Multiple imputation (MI) creates several imputed dataset by sampling from the distribution of missing entries conditional on the observation. The uncertainty due to imputations can be propagated into subsequent analyses by analyzing each imputed dataset. MI is commonly used in supervised learning when features have missing entries: a researcher first forms a prediction using each imputed dataset, and then pool all predictions into a single prediction using mean or majority vote.

For the extended Gaussian copula, we specify the distribution of $\mathbf{x}_\mathcal{M}$ based on two facts: (1) given $\mathbf{x}_O$, $\mathbf{z}_{[O]}$ is normal distributed and truncated into $\mathbf{f}_O^{-1}(\mathbf{x}_O)$; (2) $\mathbf{z}_{[\mathcal{M}]}|\mathbf{z}_{[O]}$ is normal distributed. The algorithm is presented in Algorithm 8. For sampling the truncated normal $\mathbf{z}_{[O]}|\mathbf{x}_O$, see Section 7.4.3.

Single imputation. Namely, single imputation returns a single imputed dataset. To provide a single imputation under the extended Gaussian copula,

Algorithm 8 Multiple imputation via the extended Gaussian Copula

```
1: Input: # of imputations $m$, data vector $\mathbf{x}$ observed at $\mathcal{O}$, model parameters $\mathbf{f}$
   and $\Sigma$.
2: for $s = 1, 2, \ldots, m$ do
3:     Sample $\hat{\mathbf{z}}_{[\mathcal{O}]}^{(s)} \sim \mathbf{z}_{[\mathcal{O}]}|\mathbf{x}_{\mathcal{O}}$ :   $\mathcal{N}(\mathbf{0}, \hat{\Sigma}_{[\mathcal{O}],[\mathcal{O}]})$ truncated to $\mathbf{f}_{\mathcal{O}}^{-1}(\mathbf{x}_{\mathcal{O}})$
4:     Sample $\hat{\mathbf{z}}_{[\mathcal{M}]}^{(s)} \sim \mathbf{z}_{[\mathcal{M}]}|\mathbf{z}_{[\mathcal{O}]}$ :   $\mathcal{N}\left(\Sigma_{[\mathcal{M}],[\mathcal{O}]}\hat{\mathbf{z}}_{[\mathcal{O}]}^{(s)}, \Sigma_{[\mathcal{M}],[\mathcal{M}]} - \Sigma_{[\mathcal{M}],[\mathcal{O}]}\Sigma_{[\mathcal{O}],[\mathcal{O}]}^{-1}\Sigma_{[\mathcal{O}],[\mathcal{M}]}\right)$
5:     Compute $\hat{\mathbf{x}}_{\mathcal{M}}^{(s)} = \mathbf{f}_{\mathcal{M}}(\hat{\mathbf{z}}_{[\mathcal{M}]}^{(s)})$
6: end for
7: Output: $\{\hat{\mathbf{x}}_{\mathcal{M}}^{(s)}|s = 1, .., m\}$.
```

we use

$$\hat{\mathbf{x}}_{\mathcal{M}} = \mathbf{f}_{\mathcal{M}}(\mathbb{E}[\mathbf{z}_{[\mathcal{M}]}|\mathbf{x}_{\mathcal{O}}, \Sigma, \mathbf{f}]) = \mathbf{f}_{\mathcal{M}}(\Sigma_{[\mathcal{M}],[\mathcal{O}]}\Sigma_{[\mathcal{O}],[\mathcal{O}]}^{-1}\mathbb{E}[\mathbf{z}_{[\mathcal{O}]}|\mathbf{x}_{\mathcal{O}}, \Sigma, \mathbf{f}]), \tag{7.5}$$

where we do mean computation instead of sampling for $\mathbf{z}_{[\mathcal{M}]}$. For computing $\mathbb{E}[\mathbf{z}_{[\mathcal{O}]}|\mathbf{x}_{\mathcal{O}}, \Sigma, \mathbf{f}]$, see Section 7.4.3. With a known $\mathbb{E}[\mathbf{z}_{[\mathcal{O}]}|\mathbf{x}_{\mathcal{O}}, \Sigma, \mathbf{f}]$ and $\mathbf{f}_{\mathcal{M}}$, computing $\hat{\mathbf{x}}_{\mathcal{M}}$ is straightforward. Specifically, for missing categorical variable x_j, we first compute $\hat{\mathbf{z}}_{[j]} := \mathbb{E}[\mathbf{z}_{[j]}|\mathbf{x}_{\mathcal{O}}, \Sigma, \mathbf{f}] \in \mathbb{R}^K$, then we compute $f_j(\hat{\mathbf{z}}_{[j]}; \boldsymbol{\mu}^{(j)})$ as in Eq. (7.3).

Online imputation. Online imputation refers to the task of imputing missing entries in data streams arriving at different time. We can conduct online imputation for the extended Gaussian copula similar to that for the Gaussian copula [105]: at each newly arrived incomplete data, we impute the missing entries with current saved model, and then update model parameters $\mathbf{f}$ and Σ using the new observation. The model updating rules share the same form as in [105].

7.4 Parameter estimation

The parameter estimation for the extended Gaussian copula consists of two steps: we first estimate the marginal transformation $\mathbf{f}$, and then estimate the copula correlation Σ given estimated marginal $\hat{\mathbf{f}}$. The marginal transformation differs for ordered and categorical variables. For an ordered x_j, since $f_j = F_j^{-1} \circ \Phi$, we can simply estimate F_j^{-1} using the empirical quantile function on the observed entries of x_j as in [103]. In Section 7.4.1, we show how to estimate $f_j(\cdot; \boldsymbol{\mu}^{(j)})$ in Eq. (7.3) for a categorical variable x_j. For the copula correlation, we show that its estimation under extended Gaussian copula can be reduced to that under the Gaussian copula for ordered variables, and thus we can use existing estimation algorithms in [103] for our task.

7.4.1 Marginal estimation for categorical variables

We drop the variable subscript j in this section to ease the notation. Estimating the marginal parameters $\boldsymbol{\mu}$ in Eq. (7.2) requires solving $K-1$ nonlinear equations with $K-1$ unknown values. Although the true category frequency p_k is not available, we can estimate them using the observed frequency. Unfortunately, Eq. (7.2) does not admit a closed form solution. Thus we resort to iterative root finding algorithms. However, iterative root finding algorithms require both value and derivation evaluation of the left hand side (LHS) of Eq. (7.2), $\mathbf{P}(\text{argmax}(\mathbf{z}+\boldsymbol{\mu}) = k)$, which is a K-dimensional multivariate Gaussian integral. Value and derivation evaluation for this integral function is cumbersome. We apply two approximation techniques to achieve efficient evaluation.

First we approximate probability of argmax as an expected softmax by noticing:

$$\mathbf{P}(\text{argmax}(\mathbf{z}+\boldsymbol{\mu}) = k) = \lim_{\beta\to\infty} \mathbb{E}\left[\text{softmax}(\beta(z_k+\mu_k); \beta(\mathbf{z}+\boldsymbol{\mu}))\right],$$

where $\text{softmax}(z_k; \mathbf{z}) = \exp(z_k) / \sum_{l=1}^{K} \exp(z_l)$. This approximation is accurate for large β. Second, using the reparameterization trick [54], we approximate the expected softmax using Monte Carlo samples:

$$\mathbb{E}\left[\text{softmax}(\beta(z_k + \mu_k); \beta(\mathbf{z} + \boldsymbol{\mu}))\right] \approx \frac{1}{M} \sum_{i=1}^{M} \text{softmax}(\beta(\bar{z}_k^i + \mu_k); \beta(\bar{\mathbf{z}}^i + \boldsymbol{\mu})), \tag{7.6}$$

where $\bar{\mathbf{z}}^i \overset{i.i.d.}{\sim} \mathcal{N}(\mathbf{0}, \mathbf{I}_K)$. The RHS of Eq. (7.6) now has easy to evaluate value and gradient. In summary, after approximation, we solve $\boldsymbol{\mu}$ in the nonlinear systems Eq. (7.7). We find the modified Powell method implemented in `SciPy` computes an accurate solution of Eq. (7.7).

$$\frac{1}{M} \sum_{i=1}^{M} \text{softmax}(\beta(\bar{z}_k^i + \mu_k); \beta(\bar{\mathbf{z}}^i + \boldsymbol{\mu})) = p_k, \text{ with } \mu_1 = 0, \tag{7.7}$$

for $k = 2, \ldots, K$.

The two approximation techniques share many similarities to that used for the Gumbel-softmax distribution [49]. The difference lies in the underlying distribution of the latent continuous $\mathbf{z}$: the Gumbel-softmax uses independent Gumbel distribution, while we use independent normal. Different choices of underlying distribution serve for different goals. Using Gaussian distribution, it is easy to model the joint distribution of categorical variables as well as their mix with ordered variables.Using Gumbel distribution, the argmax probability (LHS in Eq. (7.2)) has closed form expression on its parameter (like our $\boldsymbol{\mu}$). Thus it is simple to compute the gradients of the argmax probability w.r.t. its parameter.

7.4.2 Copula correlation estimation

To estimate the copula correlation, we maximize the likelihood at observed entries. Suppose there are i.i.d. samples $\mathbf{x}^i \overset{i.i.d.}{\sim} \text{EGC}(\Sigma, \mathbf{f}^{\text{ord}}, \boldsymbol{\mu})$, observed at $\mathcal{O}_i$ and missing at $\mathcal{M}_i$ for $i = 1, \ldots, n$. The observed likelihood integrates over the latent space that maps to the observation $\mathbf{x}^i_{\mathcal{O}_i}$, as in Eq. (7.8):

$$\ell_{\text{obs}}(\Sigma; \mathbf{x}^i_{\mathcal{O}_i}) = \frac{1}{n}\sum_{i=1}^{n} \int_{\mathbf{z}^i_{[\mathcal{O}_i]} \in \mathbf{f}^{-1}_{\mathcal{O}_i}(\mathbf{x}^i_{\mathcal{O}_i})} \phi(\mathbf{z}^i_{[\mathcal{O}_i]}; \mathbf{0}, \Sigma_{\mathcal{O}_i,\mathcal{O}_i})\, d\mathbf{z}^i_{[\mathcal{O}_i]}. \tag{7.8}$$

The likelihood in the Gaussian copula for ordered variables [103] has the same form as Eq. (7.8) except that $\mathcal{O}_i$ has the same length as $[\mathcal{O}_i]$ there. The similarity has an important implication: the EM algorithm in [103] also works in our case, after replacing the index in the latent space properly. The idea is that at EM iteration $t+1$, we update $\Sigma^{(t+1)}$ as the corresponding correlation when covariance is the expected "empirical covariance matrix" of $\mathbf{z}^i$. The expectation is over the distribution of $\mathbf{z}^i$ conditional on the observation $\mathbf{x}^i_{\mathcal{O}_i}$ and previous correlation estimate $\Sigma^{(t)}$. Concretely, we follow the update:

$$\Sigma^{(t+1)} \leftarrow P_{\mathcal{E}}\left(\frac{1}{n}\sum_{i=1}^{n} \mathbb{E}[\mathbf{z}^i(\mathbf{z}^i)^\top | \mathbf{x}^i_{\mathcal{O}_i}, \Sigma^{(t)}]\right), \tag{7.9}$$

where $P_{\mathcal{E}}(\Sigma)$ returns the correlation matrix corresponding to a covariance matrix Σ (positive definite). EM algorithm is guaranteed to strictly increase the likelihood in Eq. (7.8) and converges to a staionary point. The main computation is to compute the expected sample covariance of $\mathbf{z}^i$. Dropping the row index, evaluating $\mathbb{E}[\mathbf{z}\mathbf{z}^\top | \mathbf{x}_{\mathcal{O}}, \Sigma^{(t)}]$ reduces to evaluate $\mathbb{E}[\mathbf{z}_{[\mathcal{O}]} | \mathbf{x}_{\mathcal{O}}, \Sigma^{(t)}]$ and $\text{Cov}[\mathbf{z}_{[\mathcal{O}]} | \mathbf{x}_{\mathcal{O}}, \Sigma^{(t)}]$, which we discuss in Section 7.4.3.

However, with categorical variables, there requires one additional step in the EM iteration to ensure $\Sigma^{(t+1)}$ satisfies the constraint in Eq. (7.4). To achieve so, note for $\mathbf{z} = (\mathbf{z}^1, \ldots, \mathbf{z}^{p_{\text{cat}}}, \mathbf{z}_{\text{ord}}) \sim \mathcal{N}(\mathbf{0}, \Sigma)$ and each categorical variable j, we have $\mathbf{z}^j \sim \mathcal{N}(\mathbf{0}, \Sigma_{[j],[j]})$ and $\Sigma_{[j],[j]}^{-1/2}\mathbf{z}^j \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Thus for $A = \text{diag}(\Sigma_{[1],[1]}^{-1/2}, \ldots, \Sigma_{[p_{\text{cat}}],[p_{\text{cat}}]}^{-1/2}, \mathbf{I}_{\text{ord}})$, we have $A\mathbf{z} \sim \mathcal{N}(\mathbf{0}, A\Sigma A^\top)$ with $A\Sigma A^\top$ satisfies the constraint in Eq. (7.4). This update is a deterministic function of Σ, as shown below.

$$\Sigma \leftarrow A\Sigma A^\top \text{ where } A = \text{diag}(\Sigma_{[1],[1]}^{-1/2}, \ldots, \Sigma_{[p_{\text{cat}}],[p_{\text{cat}}]}^{-1/2}, \mathbf{I}_{\text{ord}}). \tag{7.10}$$

We now summarize the complete model estimation algorithm in Algorithm 9.

Algorithm 9 Correlation estimation for the extended Gaussian Copula

```
1: Input: Partial data observation $\{\mathbf{x}^i_{O_i}\}_{i=1}^n$, estimated marginal $\mathbf{f}$, initial estimate $\Sigma^{(0)}$
2: for $t = 0, 1, 2, \ldots$ until convergence do
3:     Compute $\mathbb{E}[\mathbf{z}^i(\mathbf{z}^i)^\top | \mathbf{x}^i_{O_i}, \Sigma^{(t)}]$ for $i = 1, \ldots, n$.
4:     Update $\Sigma^{(t+1)}$ as in Eq. (7.9).
5:     Adjust $\Sigma^{(t+1)}$ as in Eq. (7.10) with $\Sigma = \Sigma^{(t+1)}$.
6: end for
7: Output: $\hat{\Sigma} = \Sigma^{(t+1)}$.
```

7.4.3 Truncated normal with categorical variables

In this section, we introduce operations on $\mathbf{z}_{[O]}|\mathbf{x}_O$, normal distribution truncated into the region $\mathbf{f}_O^{-1}(\mathbf{x}_O) = \prod_{j \in O} f_j^{-1}(x_j)$. Without observed categorical variables, each $f_j^{-1}(x_j)$ is an interval (trivial single point interval for continuous x_j), and thus $\mathbf{f}_O^{-1}(\mathbf{x}_O)$ is a Cartesian product of intervals. We refer this distribution as interval truncated Gaussian. For interval truncated Gaussian, accurate sampling method has been developed in [11] and efficient mean and covariance estimation method has been developed in [103].

With observed categorical variables, the truncation region is no longer a Cartesian product intervals. Fortunately, there exists a matrix A_x (depending on $\mathbf{x}_O$) satisfying $A_x^2 = \mathbf{I}$, such that $A_x\mathbf{z}_{[O]}|\mathbf{x}_O$ is interval truncated Gaussian. Consequently, we can compute:

$$\mathbb{E}[\mathbf{z}_{[O]}|\mathbf{x}_O] = A_x\mathbb{E}[A_x\mathbf{z}_{[O]}|\mathbf{x}_O], \quad \mathrm{Cov}[\mathbf{z}_{[O]}|\mathbf{x}_O] = A_x\mathrm{Cov}[A_x\mathbf{z}_{[O]}|\mathbf{x}_O]A_x^\top. \tag{7.11}$$

Similarly, to sample $\mathbf{z}_i \overset{i.i.d.}{\sim} \mathbf{z}_{[O]}|\mathbf{x}_O$, we can first sample $\tilde{\mathbf{z}}_i \overset{i.i.d.}{\sim} A_x\mathbf{z}_{[O]}|\mathbf{x}_O$ and then use $\mathbf{z}_i = A_x\tilde{\mathbf{z}}_i$.

To see what A_x is, it helps to first see the expression of $f_j^{-1}(x_j)$ for categorical x_j. We drop the subscript j and add its dependency on a mean parameter $\boldsymbol{\mu}$ here. By definition in Eq. (7.3),

$$f^{-1}(x;\boldsymbol{\mu}) = \{\mathbf{z} : \mathbf{z}_{-k} + \boldsymbol{\mu}_{-k} < (z_k + \mu_k)\mathbf{1}_{K-1}\}, \text{ when } x = k \in \{1, \ldots, K\}. \tag{7.12}$$

Define $\tilde{\mathbf{z}} \in \mathbb{R}^K$ as $\tilde{\mathbf{z}}_{-k} = z_k\mathbf{1}_{K-1} - \mathbf{z}_{-k}$ and $\tilde{z}_k = z_k$, i.e. $\tilde{\mathbf{z}} = A_k\mathbf{z}$ where $A_k = -\mathbf{I}_K + \sum_{i=1}^K E_{ik} + E_{kk}$ (matrix E_{ij} has 1 at (i, j)-th entry and 0 elsewhere). Similarly define $\tilde{\boldsymbol{\mu}} = A_k\boldsymbol{\mu}$, we can rewrite Eq. (7.12) to $\{\tilde{\mathbf{z}} + \tilde{\boldsymbol{\mu}} > \mathbf{0}\}$, which is a Cartesian product of intervals. We can find matrix A_j as shown above for every categorical variable x_j. Now let A_x be lock-diagonal, $\mathrm{diag}(A_1, \ldots, A_{p_{\mathrm{cat}}}, \mathbf{I}_{p_{\mathrm{ord}}})$, then $A_x\mathbf{z}_{[O]}$ is truncated into a Cartesian production of intervals and A_x satisfies that $A_x^2 = \mathbf{I}$.

CHAPTER 8

SOFTWARE

In this chapter, we demonstrate how to use our software through multiple examples. This chapter is based on [104].

8.1 Introduction

Many software implementations are available for missing data imputation. The options are most plentiful in R[1]. In contrast, most advanced Python imputation packages re-implement earlier R packages. We implemented all methodology presented earlier in both Python as the **gcimpute**[2] package and R as the **gcimputeR**[3] package. Their functionality include imputation on continuous, binary, ordinal, count, and truncated mixed datasets, imputation confidence intervals, multiple imputation, large-scale imputation using the low rank Gaussian copula model, online imputation and online change point detection. Here we demonstrate how to use the Python version **gcimpute**. The usage for **gcimputeR** is similar.

Related software

There are also a few copula based imputation packages in R. **sbgcop** [43] uses the same model as our software but provides a Bayesian implementation using

[1]See `https://cran.r-project.org/web/views/MissingData.html`, `https://rmisstastic.netlify.app/rpkg/`.

[2]`https://github.com/udellgroup/gcimpute`

[3]`https://github.com/udellgroup/gcimputeR`

a MCMC algorithm. As shown in Section 2.5, our software achieves the same level of accuracy as **sbgcop** much more quickly [103]. **mdgc** [22] amends the algorithm in Chapter 2 by using a higher quality approximation for certain steps in the computation, improving model accuracy but significantly increasing the runtime when the number of variables is large ($n > 100$). **CoImp** [57] uses only complete cases to fit the copula model and is unstable when most instances have missing values. In contrast, our software can robustly fit the model even when every instance contains missing values. Moreover, our software are the first Gaussian copula package to fit extremely large datasets and streaming datasets.

8.2 Software usage

Our examples rely on some basic Python modules for data manipulation and plotting:

```
>>> import numpy as np
>>> import pandas as pd
>>> import time
>>> import matplotlib.pyplot as plt
>>> import seaborn as sns
>>> from tabulate import tabulate
```

8.2.1 Basic usage

To demonstrate the basis usage of **gcimpute**, we use demographic data from the 2014 General Social Survey (GSS) data: we consider the variables age (`AGE`), highest degree (`DEGREE`), income (`RINCOME`), subjective class identification (`CLASS`), satisfaction with the work (`SATJOB`), weeks worked last year (`WEEKSWRK`), general happiness (`HAPPY`), and condition of health (`HEALTH`). All variables are ordinal variables encoded as integers, with varying number of ordinal categories. The integers could represent numbers, such as $0, 1, \cdots, 52$ for `WEEKSWRK`, or ordered categories, such as 1 ("Very happy"), 2 ("Pretty happy"), 3 ("Not too happy") for the question "how would you say things are these days" (`HAPPY`). Many missing entries appear due to answers like "Don't know", "No answer", "Not applicable", etc. Variable histograms are plotted in Fig. 8.1 using the following code:

```
>>> from gcimpute.helper_data import load_GSS
>>> data_gss = load_GSS()
>>> fig, axes = plt.subplots(2, 4, figsize=(12,6))
>>> for i,col in enumerate(data_gss):
...     if col in ['AGE', 'WEEKSWRK']:
...         to_plot=data_gss[col].dropna()
...         to_plot.hist(ax=axes[i//4, i%4], bins=60)
...     else:
...         to_plot=data_gss[col].dropna()
...         to_plot=to_plot.value_counts().sort_index()
...         to_plot.plot(kind='bar', ax=axes[i//4, i%4])
...     _t = f'{col}, {data_gss[col].isna().mean():.2f} missing'
```

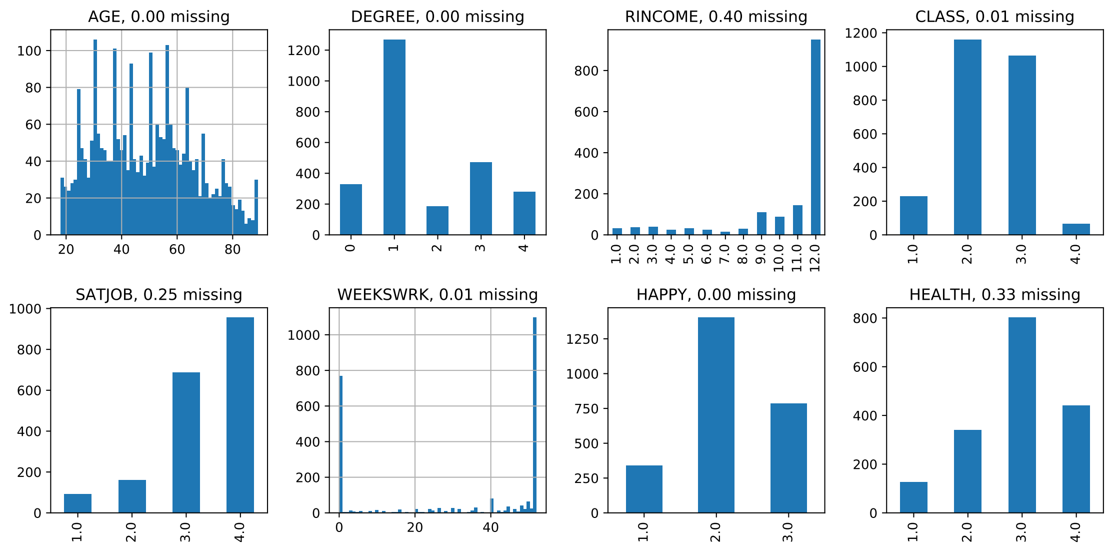


Figure 8.1: Histogram plots for GSS variables. There are 2538 samples in total.

```
...     axes[i//4, i%4].set_title(_t)
>>> plt.tight_layout()
```

We mask 10% of the observed entries uniformly at random as a test set to evaluate our imputations.

```
>>> from gcimpute.helper_mask import mask_MCAR
>>> gss_masked = mask_MCAR(X=data_gss, mask_fraction=.1)
```

The Python package has an API consistent with the `sklearn.impute` module [12]. To impute the missing entries in an incomplete dataset, we simply create a model and call `fit_transform()`. The default choice uses `training_mode='standard'`, corresponding to the algorithm in Chapter 2.

```
>>> from gcimpute.gaussian_copula import GaussianCopula
>>> model = GaussianCopula()
>>> Ximp = model.fit_transform(X=gss_masked)
```

To compare imputation performance across variables with different scales, we use the average SMAE over all variables. As shown below, the Gaussian copula imputation improves over median imputation by 10.9%.

```
>>> from gcimpute.helper_evaluation import get_smae
>>> e = get_smae(x_imp=Ximp, x_true=data_gss, x_obs=gss_masked)
>>> print(f'SMAE average over all variables: {e.mean():.3f}')
```

```
SMAE average over all variables: 0.891
```

We can also extract the copula correlation estimates to see which variables are correlated, as in Fig. 8.2. Interestingly, `DEGREE` and `CLASS` have the largest positive correlation 0.39, while `WEEKSWRK` and `AGE` have the largest negative correlation −0.37.

```
>>> copula_corr_est = model.get_params()['copula_corr']
>>> mask = np.zeros_like(copula_corr_est)
>>> mask[np.triu_indices_from(mask)] = True
>>> names = data_gss.columns
>>> sns.heatmap(np.round(copula_corr_est,2),
                xticklabels=names, yticklabels=names,
                annot=True, mask=mask, square=True, cmap='vlag')
```

Determining the variable types

The choice of variable type can have a strong effect on inference and imputation. **gcimpute** defines five variable types: `'continuous'`, `'ordinal'`,

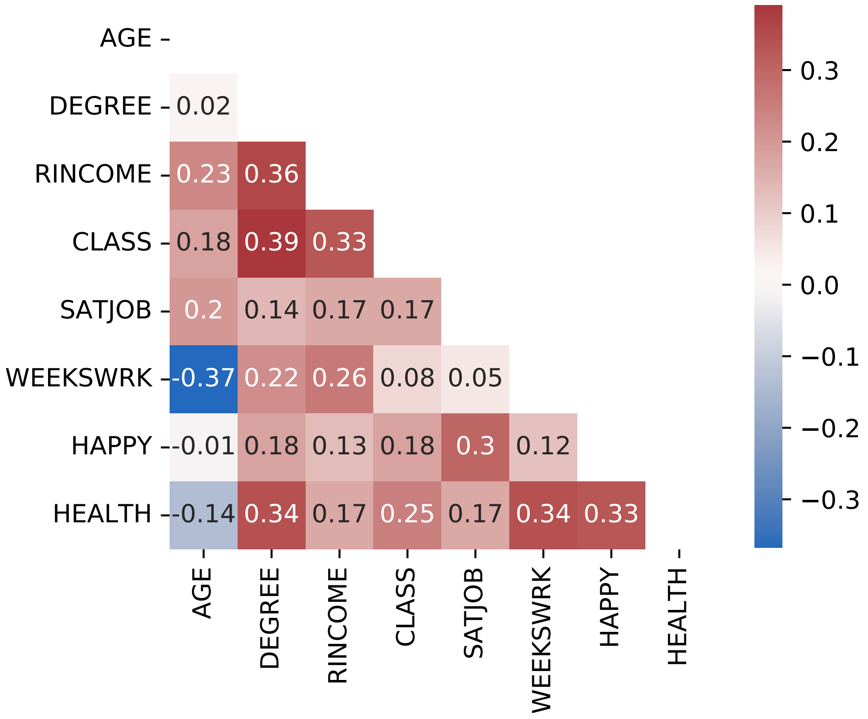


Figure 8.2: The estimated latent copula correlation among GSS variables.

`'lower_truncated'`, `'upper_truncated'` and `'twosided_truncated'`. **gcimpute** provides good default guesses of data types, which we used in the previous call. After fitting the model, we can query the model to ask which variable type was chosen as shown below. Only `AGE` is treated as continuous; all other variables are treated as ordinal. No variable is treated as truncated.

```
>>> for k,v in model.get_vartypes(feature_names=names).items():
>>>     print(f'{k}: {v}')

continuous: ['AGE']
ordinal: ['DEGREE', 'RINCOME', 'CLASS', 'SATJOB',
                         'WEEKSWRK', 'HAPPY', 'HEALTH']
lower_truncated: []
upper_truncated: []
```

```
twosided_truncated: []
```

We can specify the type of each variable in `model.fit_transform()` directly. Otherwise, the default setting works well. It guesses the variable type based on the frequency of observed unique values. A variable is treated as continuous if its mode's frequency is less than 0.1. A variable is treated as lower/upper/two sided truncated if its minimum's/maximum's/minimum's and maximum's frequency is more than 0.1 and the distribution, excluding these values, is continuous by the previous rule. All other variables are ordinal. The default threshold value 0.1 works well in general, but can be changed using the parameter `min_ord_ratio` in the model call `GaussianCopula()`. For example, let us look at the frequency of the min, max, and mode for each GSS variable.

```
>>> def key_freq(col):
...     freq = col.value_counts(normalize=True)
...     key_freq = {'mode_freq':freq.max()}
...     _min, _max = col.min(), col.max()
...     key_freq['min_freq'] = freq[_min]
...     key_freq['max_freq'] = freq[_max]
...     f = freq.drop(index = [_min, _max])
...     key_freq['mode_freq_nominmax'] = f.max()/f.sum()
...     return pd.Series(key_freq).round(2)
>>> table = data_gss.apply(lambda x: key_freq(x.dropna())).T
>>> print(tabulate(table, headers='keys', tablefmt='psql'))
```

```
+----------+--------+-------+-------+-----------------+
|          |   mode |   min |   max |   mode_nominmax |
|----------+--------+-------+-------+----------------|
| AGE      |   0.02 | 0     |  0.01 |           0.02 |
| DEGREE   |   0.5  | 0.13  |  0.11 |           0.66 |
| RINCOME  |   0.62 | 0.02  |  0.62 |           0.26 |
| CLASS    |   0.46 | 0.09  |  0.03 |           0.52 |
| SATJOB   |   0.5  | 0.5   |  0.05 |           0.81 |
| WEEKSWRK |   0.44 | 0.31  |  0.44 |           0.13 |
| HAPPY    |   0.55 | 0.31  |  0.13 |           1    |
| HEALTH   |   0.47 | 0.26  |  0.07 |           0.7  |
+----------+--------+-------+-------+----------------+
```

Only `AGE` has mode frequency below 0.1 and thus is treated as continuous. All other variables have strong concentration on a single value, even after removing the min and max, so these are treated as ordinal. `WEEKSWRK` is an interesting example. It has 53 levels, yet 75% of the population works either 0 or 52 weeks per year: thus it is not treated as a continuous variable. Interestingly, if we insist that `WEEKWRK` be treated as continuous, the algorithm diverges! We discuss this phenomenon in an online vignette[4].

Monitoring the algorithm fitting

gcimpute considers the model to have converged when the model parameters no longer change rapidly: It terminates when $||\Sigma^{t+1} - \Sigma^t||_F/||\Sigma^t||_F$ falls below the specified `tol`, where Σ^t is the model parameter estimate at the t-th iteration and $||\cdot||_F$ denotes the Frobenius norm. In practice, the default value `tol=0.01` works

[4]`https://github.com/udellgroup/gcimpute/blob/master/Examples/Trouble_shooting.ipynb`

well and the algorithm converges in less than 30 iterations in most cases.

Tracking the objective value may also be useful. The objective value is the marginal likelihood at the observed locations, averaged over all instances. When all variables are continuous, **gcimpute** computes the exact likelihood. In other cases, **gcimpute** computes an approximation to the likelihood. The approximation behaves well in most cases including those with all ordinal variables: it monotonically increases during the fitting process and finally converges.

To monitor the parameter update and the objective during fitting, simply set `verbose=1` in the model call.

```
>>> model = GaussianCopula(verbose=1)
>>> Ximp = model.fit_transform(X=gss_masked)
```

```
Iter 1: copula parameter change 0.1168, likelihood -9.6913
Iter 2: copula parameter change 0.0644, likelihood -9.5869
Iter 3: copula parameter change 0.0366, likelihood -9.5278
Iter 4: copula parameter change 0.0220, likelihood -9.4942
Iter 5: copula parameter change 0.0140, likelihood -9.4744
Iter 6: copula parameter change 0.0093, likelihood -9.4623
Convergence achieved at Iter 6
```

Using a tolerance `tol` that is too small can require many more iterations and can cause overfitting. Hence users may wish to tune `tol` for a specific dataset for best performance using `fit_transform_evaluate()`. This function runs the EM algorithm for specified `n_iter` iterations and evaluates the imputed dataset using the provided `eval_func` at each iteration. The function `eval_func` should take an imputed dataset as input and output the desired evaluation results. We can design `eval_func` to evaluate the imputation accuracy or the prediction accuracy of a supervised learning pipeline with the imputed dataset as feature matrix. For example, to evaluate the mean SMAE of the GSS dataset for up to 15 iterations, we can run the following code:

```
>>> m = GaussianCopula(verbose=1)
>>> def get_err(x):
>>>     e = get_smae(x, x_true=data_gss, x_obs=gss_masked)
>>>     return e.mean()
>>> r = m.fit_transform_evaluate(X=gss_masked,
>>>                                 eval_func=get_err,
>>>                                 num_iter=15)
>>> plt.plot(list(range(1, 16, 1)), r['evaluation'])
>>> plt.title('Imputation error versus run iterations')
>>> plt.xlabel("Run iterations")
>>> plt.ylabel("SMAE")
```

```
Iter 1: copula parameter change 0.1168, likelihood -9.6913
Iter 2: copula parameter change 0.0644, likelihood -9.5869
Iter 3: copula parameter change 0.0366, likelihood -9.5278
Iter 4: copula parameter change 0.0220, likelihood -9.4942
Iter 5: copula parameter change 0.0140, likelihood -9.4744
Iter 6: copula parameter change 0.0093, likelihood -9.4623
Iter 7: copula parameter change 0.0063, likelihood -9.4545
Iter 8: copula parameter change 0.0044, likelihood -9.4494
```

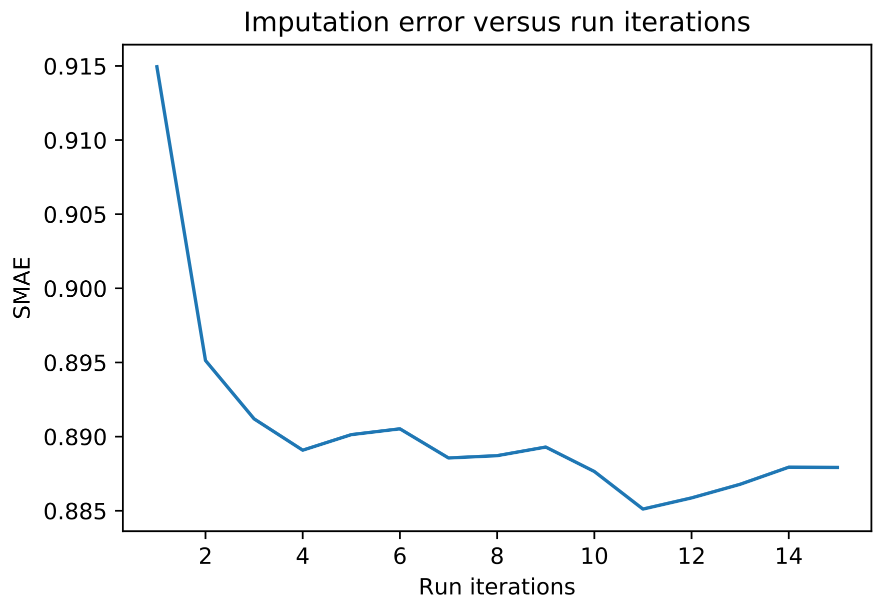


Figure 8.3: The imputation error among GSS variables is plotted w.r.t. the number of iterations run in **gcimpute**. Satisfactory results emerge after four iterations.

```
Iter 9: copula parameter change 0.0032, likelihood -9.4460

Iter 10: copula parameter change 0.0023, likelihood -9.4437

Iter 11: copula parameter change 0.0017, likelihood -9.4421

Iter 12: copula parameter change 0.0012, likelihood -9.4409

Iter 13: copula parameter change 0.0009, likelihood -9.4401

Iter 14: copula parameter change 0.0007, likelihood -9.4396

Iter 15: copula parameter change 0.0005, likelihood -9.4392
```

Shown in Fig. 8.3, the imputation error fluctuates in a small range from 0.885 to 0.890 after four iterations. The default parameter setting stops at iteration 6.

8.2.2 Acceleration for large datasets

In this section, we will see how to speed up **gcimpute** . To use parallelism with *m* cores, we simply set `n_jobs=m` in the model call `GaussianCopula()`. To use mini-batching training, we set `training_mode` as `‘minibatch-offline’` also in the model call `GaussianCopula()`. The low rank Gaussian copula is invoked using a different model call `LowRankGaussianCopula(rank=k)` with desired rank k. Mini-batch training for the low rank Gaussian copula is more challenging and remains for future work, as the low rank update is nonlinear. Nevertheless, for large n and large p, the parallel low rank Gaussian copula already converges quite rapidly.

Accelerating datasets with many samples: mini-batch training

Mini-batch training requires choosing a decaying step size $\{\gamma_t\}$ in Eq. (5.6), a batch size and a maximum number of iterations. The default setting can be simply invoked by calling

```
m = GaussianCopula(training_mode=‘minibatch-offline’)
```

or explicitly as below:

```
m = GaussianCopula(training_mode=’minibatch-offline’,
                   stepsize_func = lambda t,c=5:c/(c+t),
                   batch_size = 100,
                   num_pass = 2
                   )
```

The step size sequence γ_t must satisfy $\gamma_t \in (0,1)$ for all t and $\sum_{t=1}^{\infty} \gamma_t^2 < \sum_{t=1}^{\infty} \gamma_t = \infty$. By default, we recommend using $\gamma_t = c/(c+t)$ with $c > 0$. We find it generally suffices to tune c in the range $(0, 10)$. The default setting $c = 5$ works well in many of our experiments.

Mini-batch training requires a batch size $s \geq p$ to avoid inverting a singular matrix [105]. In practice, it is easy to select $s \geq p$, since problems with large p should use `LowRankGaussianCopula()` instead.

The maximum number of iterations matters more for mini-batch methods, because the stochastic fluctuation over mini-batches makes it hard to decide convergence based on the parameter update. Instead of specifying an exact maximum number of iterations, it may be more convenient to select a desired number of complete passes through the data (epochs), i.e. `max_iter=` $\lceil \frac{n}{s} \rceil$`×num_pass` with s as the mini-batch size. Often using `num_pass=` 2 (the default setting) or 3 gives satisfying results.

We now run mini-batch training with the defaults on the GSS dataset:

```
>>> t1=time.time()
>>> m = GaussianCopula(training_mode='minibatch-offline')
>>> Ximp = m.fit_transform(X=gss_masked)
>>> t2=time.time()
>>> print(f'Runtime: {t2-t1:.2f} seconds')
>>> e = get_smae(Ximp, x_true=data_gss, x_obs=gss_masked)
>>> print(f'Imputation error: {e.mean():.3f}')
```

```
Runtime: 15.27 seconds
Imputation error: 0.886
```

Let us also re-run and record the runtime of the standard training mode:

```
>>> t1=time.time()
>>> _ = GaussianCopula().fit_transform(X=gss_masked)
>>> t2=time.time()
>>> print(f'Runtime: {t2-t1:.2f} seconds')
```

```
Runtime: 39.47 seconds
```

Mini-batch training not only reduces runtime by 61% but also improves the imputation error (from 0.891 to 0.886)!

Accelerating datasets with many variables: low rank structure

The low rank Gaussian copula (LRGC) model accelerates convergence by decreasing the number of model parameters. Here we showcase its performance on a subset of the MovieLens1M dataset [40]: the 400 movies with the most ratings and users who rates at least 150 of these movies in the scale of $\{1, 2, 3, 4, 5\}$. That yields a dataset consisting of 914 users and 400 movies with 53.3% of ratings observed. We further mask 10% entries for evaluation.

```
>>> gcimpute.helper_data import load_movielens1m
>>> d_movie = load_movielens1m(num=400, min_obs=150)
>>> dm_masked = mask_MCAR(X=d_movie, mask_fraction=0.1)
```

We run `GaussianCopula()` and `LowRankGaussianCopula(rank=10)`. Here our goal is not to choose the optimal rank, but rather show the runtime comparison between two models.

```
>>> from gcimpute.low_rank_gaussian_copula
                         import LowRankGaussianCopula
>>> a = time.time()
>>> model_movie_lrgc = LowRankGaussianCopula(rank=10)
>>> imp_lrgc = model_movie_lrgc.fit_transform(X=dm_masked)
>>> print(f'LRGC runtime {(time.time()-a)/60:.2f} mins.')
>>> a = time.time()
>>> model_movie_gc = GaussianCopula()
>>> imp_gc = model_movie_gc.fit_transform(X=dm_masked)
>>> print(f'GC runtime {(time.time()-a)/60:.2f} mins.')
```

```
LRGC runtime 7.86 mins.
GC runtime 11.66 mins.
```

Here we already see that LRGC already reduces the runtime by 34% compared to the standard Gaussian copula, although the number of variables $p = 400$ is not particularly large. When the number of variables is much larger, the acceleration is also more significant. Moreover, LRGC improves the imputation error from 0.616 to 0.583, as shown below.

```
>>> from gcimpute.helper_evaluation import get_mae
>>> e_gc=get_mae(imp_gc, x_true=d_movie, x_obs=dm_masked)
>>> e_lrgc=get_mae(imp_lrgc, x_true=d_movie, x_obs=dm_masked)
>>> print(f'LRGC imputation MAE: {e_lrgc:.3f}')
>>> print(f'GC imputation MAE: {e_gc:.3f}')
```

```
LRGC imputation MAE: 0.583
GC imputation MAE: 0.616
```

8.2.3 Imputation for streaming datasets

gcimpute's `'minibatch-online'` training mode performs streaming imputation: as new samples arrive, it imputes the missing data immediately and then updates the model parameters. We showcase its performance on eight daily recorded economic time series variables from federal reserve bank of St. Louis (FRED), consisting of 3109 days from 2008-06-03 to 2020-12-31. The selected eight variables are diverse and among the most popular economic variables in FRED: gold volatility index, stock volatility index, bond spread, dollar index, inflation rate, interest rate, crude oil price, and US dollar to Euro rate, shown in Fig. 8.4.

```
>>> from gcimpute.helper_data import load_FRED
>>> fred = load_FRED()
>>> fred.plot(subplots = True, layout = (2,4),
...            figsize = (16,6), legend = False,
...            title = fred_data.columns.to_list()
...           )
```

Here we consider a scenario in which some variables are observed as soon as they are generated, while others are observed after a lag of one day. The goal is to predict the unobserved variables each day. We use stock `StockVolatility` and `CrudeOilPrice` as two unobserved variables. Each day, using a fitted Gaussian copula model, we predict their values based on both their historical values (through the marginal) and the six other observed variables at that day (through the copula correlation). After we make our prediction, the actual values are revealed and used to update the Gaussian copula model. **gcimpute** con-

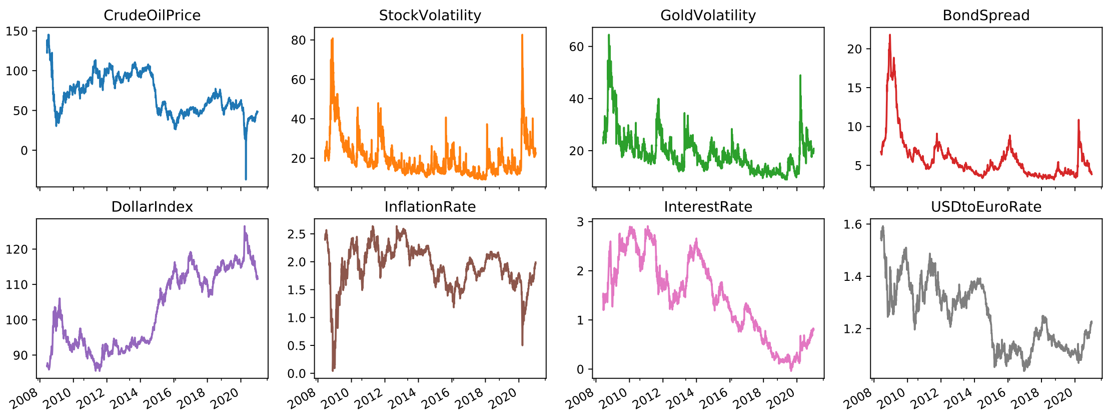


Figure 8.4: Values of eight selected FRED economic variables from 2008-06-03 to 2020-12-31 are plotted.

veniently supports this task. Let us first create a Gaussian copula model to impute streaming datasets (`training_mode='minibatch-online'`), shown as below.

```
>>> m = GaussianCopula(training_mode='minibatch-online',
...                    window_size=10,
...                    const_stepsize=0.1,
...                    batch_size=10,
...                    decay=0.01
...                    )
```

Three hyperparameters control the learning rate of the model: `window_size` controls the number of recent observations used for marginal estimation; `const_stepsize` controls the size of the copula correlation update; and `batch_size` is the frequency of the copula correlation update. In contrast, `decay` only controls the imputation and does not influence the model update (decay rate d defeind in Section 5.2.1). Smaller values of `decay` put less weight on old observations, i.e., forget stale data faster. In economic time series, yesterday's observation often predicts today's value well. We use a small value decay=.01, so that the imputation depends most strong on yesterday's observation, but interpolates all values in the window. These parameters can be tuned for best performance.

Next, to conduct the experiment described above, we prepare two data matrices with one row for each temporal observation: X for imputing missing entries and X_true for updating the model. We use first 25 rows to initialize the model.

```
>>> fred_m = fred.assign(StockVolatility=np.nan,
...                      CrudeOilPrice=np.nan)
>>> Ximp = m.fit_transform(X=fred_m, X_true=fred, n_train=25)
```

More concretely, a Gaussian copula model receives the t-th row of X, imputes its missing entries, and then is asked to update parameters of the model using the t-th row of X_true. X_true must agree with X at all observed entries in X, but may reveal additional entries that are missing in X. By default, X_true=None, indicating no additional entries beyond X are available. In this example, two columns of fred_masked are missing: StockVolatility and CrudeOilPrice. All other columns fully observed. fred_data has all columns fully observed.

We now evaluate the imputation performance and compare against a simple but powerful alternative, yesterday's observation. The predicted series of both methods are almost visually indistinguishable from the true values in Fig. 8.4, but the Gaussian copula predictions perform better on average, with MSE.

```
>>> n_train = 25
>>> names = ['CrudeOilPrice', 'StockVolatility']
>>> for i, col in enumerate(names):
...     _true = fred_data[col][n_train:].to_numpy()
...     _yesterday = fred_data[col][n_train-1:-1].to_numpy()
...     e_yes = np.power(_yesterday-_true, 2).mean()
...     e_GC = np.power(Ximp[n_train:,i]-_true, 2).mean()
...     print(f'For {col}:')
...     print(f'Gaussian Copula Pred MSE: {e_yes:.3f}')
...     print(f'Yesterday Value Pred MSE: {e_GC:.3f}')
```

```
For CrudeOilPrice:
Gaussian Copula Pred MSE: 3.672
Yesterday Value Pred MSE: 4.313
For StockVolatility:
Gaussian Copula Pred MSE: 3.998
Yesterday Value Pred MSE: 4.368
```

8.2.4 Imputation uncertainty

So far we have seen several methods to impute missing data. **gcimpute** also provides functionality to quantify the uncertainty of the imputations: multiple imputation, confidence interval for a single imputation, and relative reliability for a single imputation. We present the first two notions here, since they are widely used. The third, relative reliability, aims to rank the imputation quality among all imputed entries Section 4.2. It is well suited for the top-k recommendation task in collaborative filtering.

Multiple imputation

Multiple imputation creates several imputed copies of the original dataset, each having potentially different imputed values. The uncertainty due to imputations can be propagated into subsequent analyses by analyzing each imputed dataset. Multiple imputation is commonly used in supervised learning when features may have missing entries: a researcher creates multiple imputed feature datasets, then trains a model with each imputed training feature dataset and predicts with each imputed test feature vector. Finally, they pool all predictions into a single prediction, for example, using the mean or majority vote. An ensemble model like this often outperforms a single model trained from a single imputation.

We show to use multiple imputation in **gcimpute** on a regression task from UCI datasets, the white wine quality dataset [24]. This dataset has 11 continuous features and a rating target for 4898 samples.

```
>>> gcimpute.helper_data import load_whitewine
>>> df = load_whitewine()
```

We now randomly mask 30% of entries and fit a Gaussian copula model to the masked dataset.

```
>>> X = df.to_numpy()[:,:-1]
>>> X_masked = mask_MCAR(X_wine, mask_fraction=0.3)
>>> m = GaussianCopula()
>>> Ximp = m.fit_transform(X=X_masked)
```

Now we use the first 4000 instances as a training dataset and the remaining 898 instances as test dataset. Since the goal is to show how to use multiple imputation, we use simple linear model as the prediction model. Now, let us first examine the MSE of the linear model fitted on the complete feature dataset.

```
>>> from sklearn.metrics import mean_squared_error as MSE
>>> from sklearn.linear_model import LinearRegression as LR
>>> X_train,X_test = X[:4000],X[4000:]
>>> y_train,y_test = df['quality'][:4000],df['quality'][4000:]
>>> y_pred = LR().fit(X=X_train, y=y_train).predict(X=X_test)
>>> np.round(MSE(y_test, y_pred),4)

0.5121
```

Now let us examine the MSE of the linear model fitted on the single imputed dataset.

```
>>> Ximp_train, Ximp_test = Ximp[:4000], Ximp[4000:]
>>> m_LR = LR().fit(X=Ximp_train, y=y_train)
>>> y_pred = m_LR.predict(X=Ximp_test)
>>> np.round(MSE(y_test, y_pred),4)

 0.5295
```

Not surprisingly, replacing 30% feature values with the corresponding imputation does hurt the prediction accuracy. Now let us draw 5 imputed datasets, train a linear model and get prediction for each imputed dataset, and derive the final prediction as the average across 5 different prediction. As shown below, the mean-pooled prediction improves the results from single imputation and performs very close to the results using the complete dataset.

```
>>> Ximp_mul = m.sample_imputation(X=X_masked, num=5)
>>> y_pred_mul = []
>>> for i in range(5):
...     _Ximp = Ximp_mul[...,i]
...     _Ximp_train, _Ximp_test = _Ximp[:4000], _Ximp[4000:]
...     _m_LR = LR().fit(X=_Ximp_train, y=y_train)
...     _y_pred = m_LR.predict(X=_Ximp_test)
...     y_pred_mul.append(_y_pred)
>>> y_pred_mul = np.array(y_pred_mul).mean(axis=0)
>>> np.round(MSE(y_test, y_pred_mul), 4)
```

```
 0.5152
```

Imputation confidence intervals

Confidence intervals (CI) are another important measure of uncertainty. **gcimpute** can return a CI for each imputed value: for example, a 95% CI should contain the true missing data with probability 95%. In general, these CI are not symmetric around the imputed value due to the nonlinear transformation **f**.

We will continue to use the white wine dataset to illustration. After fitting the Gaussian copula model, we can obtain the imputation CI as shown below.

```
>>> ct = m.get_imputed_confidence_interval()
>>> upper, lower = ct['upper'], ct['lower']
```

By default, the method `get_imputed_confidence_interval()` extracts the imputation CI of the data used to fit the Gaussian copula model, with significance level `alpha=0.05`. The empirical coverage of the returned CI is 0.943, as shown below. Hence we see the constructed CI are well calibrated on this dataset.

```
>>> missing = np.isnan(X_masked)
>>> X_m = X[missing]
>>> cover = (lower[missing]<X_m) & (upper[missing]>X_m)
>>> np.round(cover.mean(),3)
```

```
0.943
```

The default setting uses an analytic expression to obtain the CI. As stated in Section 4.2, when some variables are not continuous, a safer approach builds CI using empirical quantiles computed from multiple imputed values. Let us now construct the quantile CI and compare them with the analytical counterparts. As shown below, the quantile CI has almost the same empirical coverage rate as the analytical CI, validating that the CI are well calibrated.

```
>>> ct_q = m.get_imputed_confidence_interval(type='quantile')
>>> upper_q, lower_q = ct_q['upper'], ct_q['lower']
>>> cover_q = (lower_q[missing]<X_m) & (upper_q[missing]>X_m)
>>> np.round(cover_q.mean(),3)

0.942
```

8.3 Discussion

Although this chapter focuses on missing data imputation, **gcimpute** can also be used to fit a Gaussian copula model to complete mixed datasets. The resulting latent correlations may be useful to understand multi-view data collected on the same subjects from different sources. As far as we know, no other software supports Gaussian copula estimation for mixed continuous, binary, ordinal and truncated variables. Fan et al. [30] only supports continuous and binary mixed data; Feng and Ning [34] supports continuous, binary and ordinal mixed data; Yoon et al. [99] supports continuous, binary and zero-inflated (a special case of truncated) mixed data.

One major area for future research is the appropriate treatment of nominal values. **gcimpute** currently encodes nominal variables using a one-hot encoding, however, this encoding is not self-consistent for the copula model since our estimation procedure ignores the one-hot constraint. We advise users to model their features directly as ordinal or binary, if possible.

APPENDIX A

APPENDIX OF CHAPTER 2

A.1 Truncated normal moments approximations

Denote the observation $\{\mathbf{x}_O, \Sigma\}$ i.e. $\{\mathbf{z}_C = \mathbf{f}_C^{-1}(\mathbf{x}_C), \mathbf{z}_{\mathcal{D}} \in \mathbf{f}_{\mathcal{D}}^{-1}(\mathbf{x}_{\mathcal{D}}), \Sigma\}$ as $\{*\}$. Since the task is to compute the marginal mean and variance of a multivariate truncated normal, we suppose $\mathcal{M} = \emptyset$ here without loss of generality. For each $j \in \mathcal{D}$, we use the law of total expectation by conditioning on $\mathbf{z}_{\mathcal{D}-j}$ first. Given $\{*, \mathbf{z}_{\mathcal{D}-j}\}$, z_j is univariate normal with mean $\tilde{\mu}_j = \Sigma_{j,-j}\Sigma_{-j,-j}^{-1}\mathbf{z}_{-j}$ and variance $\tilde{\sigma}_j^2 = 1 - \Sigma_{j,-j}\Sigma_{-j,-j}^{-1}\Sigma_{-j,j}$, truncated to the region $f_j^{-1}(x_j)$, where the index $-j$ means all dimensions but j, i.e., $[p]\setminus j$. The region $f_j^{-1}(x_j)$ is an interval: $f_j^{-1}(x_j) = (a_j, b_j]$. Here are three cases: (1) $a_j, b_j \in \mathbb{R}$; (2) $a_j \in \mathbb{R}, b_j = \infty$; (3) $a_j = -\infty, b_j \in \mathbb{R}$. The computation for all cases are similar. We take the first case as an example. First we introduce a lemma describing the first and second moments of a truncated univariate normal.

Lemma 8. Suppose a univariate random variable $z \sim \mathcal{N}(\mu, \sigma^2)$. For constants $a < b$, let $\alpha = (a - \mu)/\sigma$ and $\beta = (b - \mu)/\sigma$. Then the mean and variance of z truncated to the interval $(a, b]$ are:

$$\mathrm{E}(z|a < z \leq b) = \mu + \frac{\phi(\alpha) - \phi(\beta)}{\Phi(\beta) - \Phi(\alpha)} \cdot \sigma$$

$$\mathrm{Var}(z|a < z \leq b) = \left(1 + \frac{\alpha\phi(\alpha) - \beta\phi(\beta)}{\Phi(\beta) - \Phi(\alpha)} - \left(\frac{\phi(\alpha) - \phi(\beta)}{\Phi(\beta) - \Phi(\alpha)}\right)^2\right)\sigma^2.$$

Plugging $\mu = \tilde{\mu}_j, \sigma^2 = \tilde{\sigma}_j^2$ and $(a, b] = f_j^{-1}(x_j)$ into the above mean and variance formulas, we obtain the expression of $f_j(\mathbf{z}_{\mathcal{D}-j}; x_j, \Sigma)$ defined in Section 2.4.3, and the univariate truncated normal variance $\mathrm{Var}[z_j|\mathbf{z}_{\mathcal{D}-j}, \mathbf{x}_{\mathcal{O}}, \Sigma] =: h_j(\mathbf{z}_{\mathcal{D}-j}; x_j, \Sigma)$, a nonlinear function $\mathbb{R}^{|\mathcal{D}|-1} \to \mathbb{R}$, parameterized by x_j and Σ. Write down the formula for marginal variance conditional on observation:

$$\begin{aligned}\mathrm{Var}[z_j|*] &= \mathrm{E}\left[\mathrm{Var}[z_j|\mathbf{z}_{\mathcal{D}-j}, *]\Big|*\right] + \mathrm{Var}\left[\mathrm{E}[z_j|\mathbf{z}_{\mathcal{D}-j}, *]\Big|*\right] \\ &= \mathrm{E}\left[h_j(\mathbf{z}_{\mathcal{D}-j}; x_j, \Sigma)\Big|*\right] + \mathrm{Var}\left[f_j(\mathbf{z}_{\mathcal{D}-j}; x_j, \Sigma)\Big|*\right]\end{aligned}$$

We approximate the first term as $h_j(\mathrm{E}[\mathbf{z}_{\mathcal{D}-j}|*]; x_j, \Sigma)$. In [39], the second term, is approximated as $\mathrm{Var}[\tilde{\mu}_j|*]$ based on $\mathrm{E}\left[f_j^2(\mathbf{z}_{\mathcal{D}-j}; x_j, \Sigma)\Big|*\right] \approx f_j^2(\mathrm{E}[\mathbf{z}_{\mathcal{D}-j}|*]; x_j, \Sigma)$. However, we found in practice simply dropping the second term performs better.

In summary, given an estimate $\hat{\mathbf{z}}_{\mathcal{D}}^{(t)} \approx \mathrm{E}[\mathbf{z}_{\mathcal{D}}|\mathbf{x}_{\mathcal{O}}, \Sigma^{(t)}]$ and $\Sigma^{(t+1)}$, for $j \in \mathcal{D}$, we update $\mathrm{E}[z_j|\mathbf{x}_{\mathcal{O}}, \Sigma^{(t+1)}] \approx f_j(\hat{\mathbf{z}}_{\mathcal{D}-j}^{(t)}; x_j, \Sigma^{(t+1)})$ and $\mathrm{Var}[z_j|\mathbf{x}_{\mathcal{O}}, \Sigma^{(t+1)}] \approx h_j(\hat{\mathbf{z}}_{\mathcal{D}-j}^{(t)}; x_j, \Sigma^{(t+1)})$. In other words, we update the conditional mean and variance of z_j as the univariate truncated normal mean and variance with all other observed ordinal dimensions equal to their mean from last iteration, i.e. $\mathbf{z}_{\mathcal{D}-j} = \hat{\mathbf{z}}_{\mathcal{D}-j}^{(t)}$.

A.2 Computational details

Given $\mathrm{E}[\mathbf{z}_{\mathcal{O}}|*], \mathrm{E}[\mathbf{z}_{\mathcal{M}}|*]$ and $\mathrm{Cov}[\mathbf{z}_{\mathcal{O}}|*]$, it suffices to compute $\mathrm{E}[\mathbf{z}_{\mathcal{M}}\mathbf{z}_{\mathcal{O}}^\top|*]$ and $\mathrm{E}[\mathbf{z}_{\mathcal{M}}\mathbf{z}_{\mathcal{M}}^\top|*]$ for $\mathrm{Cov}[\mathbf{z}_{\mathcal{M}}, \mathbf{z}_{\mathcal{O}}|*]$ and $\mathrm{Cov}[\mathbf{z}_{\mathcal{M}}|*]$. Using the law of total expectation, we have:

$$\begin{aligned}\mathrm{E}[\mathbf{z}_{\mathcal{M}}\mathbf{z}_{O}^{\top}|*] &= \mathrm{E}\left[\mathrm{E}[\mathbf{z}_{\mathcal{M}}\mathbf{z}_{O}^{\top}|\mathbf{z}_{O},*]\middle|*\right] = \mathrm{E}\left[\mathrm{E}[\mathbf{z}_{\mathcal{M}}|\mathbf{z}_{O},*]\cdot\mathbf{z}_{O}^{\top}\middle|*\right] = \mathrm{E}\left[\Sigma_{\mathcal{M},O}\Sigma_{O,O}^{-1}\mathbf{z}_{O}\cdot\mathbf{z}_{O}^{\top}\middle|*\right]\\ &= \Sigma_{\mathcal{M},O}\Sigma_{O,O}^{-1}\mathrm{E}[\mathbf{z}_{O}\mathbf{z}_{O}^{\top}|*].\end{aligned}$$

$$\begin{aligned}\mathrm{E}[\mathbf{z}_{\mathcal{M}}\mathbf{z}_{\mathcal{M}}^{\top}|*] &= \mathrm{E}\left[\mathrm{E}[\mathbf{z}_{\mathcal{M}}\mathbf{z}_{\mathcal{M}}^{\top}|\mathbf{z}_{O},*]\middle|*\right]\\ &= \mathrm{E}\left[\mathrm{Cov}[\mathbf{z}_{\mathcal{M}}|\mathbf{z}_{O},*]\middle|*\right] + \mathrm{E}\left[\mathrm{E}[\mathbf{z}_{\mathcal{M}}|\mathbf{z}_{O},*]\cdot\mathrm{E}[\mathbf{z}_{\mathcal{M}}^{\top}|\mathbf{z}_{O},*]\middle|*\right]\\ &= \Sigma_{\mathcal{M},\mathcal{M}} - \Sigma_{\mathcal{M},O}\Sigma_{O,O}^{-1}\Sigma_{O,\mathcal{M}} + \Sigma_{\mathcal{M},O}\Sigma_{O,O}^{-1}\mathrm{E}[\mathbf{z}_{O}|*]\mathrm{E}[\mathbf{z}_{O}^{\top}|*]\Sigma_{O,O}^{-1}\Sigma_{O,\mathcal{M}}.\end{aligned}$$

A.3 Experimental details

Implementation details For `softImpute`, we first center the rows and columns, then select the penalization parameter in the path from 45 (rank 12) to 6 (rank 207) with 50 points. For `GLRM`, we use quadratic regularization on X factor and ordinal regularization on Y factor. The model is fitted with SVD initialization and offest term. After a small grid search, we select the quadratic regularization parameter as $n_{\text{obs}}\times 1.2\times 10^{-4}$ where n_{obs} is the number of observed entries. Then the rank is selected through an exhaustive search. For `xPCA` and `imputeFAMD`, the rank is selected through an exhaustive search.

Results of sbgcop on real datasets For GSS data, `Copula-EM` takes 24s, while `sbgcop` with 1000 iterations takes 87s, with imputation error: `CALSS`, 0.992(0.13); `LIFE`, 0.924(0.7); `HEALTH`, 1.132(0.15); `HAPPY`, 1.231(0.11); `INCOME`, 0.931(0.03).

For movielens data, `Copula-EM` takes 9 mins, while `sbgcop` with 200 iterations takes 33 mins, with imputation error: MAE, 0.752(0.004); RMSE, 1.030(0.005).

For CAL500exp data, `Copula-EM` takes 80s, while `sbgcop` with 500 iterations takes 290s, with imputation error: 1.301(0.019) for 40% missing ratio; 1.328(0.015) for 50% missing ratio; 1.379(0.016) for 60% missing ratio.

For four small datasets used in Section 2.5.5, the time `sbgcop` with 1000 iterations takes is 2 times to 9 times (varying over datasets) of the time `Copula-EM` takes. The corresponding imputation error is: ESL label 0.466(0.04), feature 0.649(0.02); LEV label 0.849(0.03), feature 0.936(0.01); GBSG ordinal 0.992(0.03), continuous 0.953(0.02); TIPS ordinal 0.984(0.06), continuous 0.768(0.05).

Datasets description for Section 2.5.5

ESL This dataset contains profiles of applicants for certain jobs. The recruiting company, based upon psychometric test results and interviews with the candidates, determined the values of the input attributes. The output is an overall score corresponding to the degree of fitness of the candidate.

LEV This dataset contains lecturer evaluations. Students evaluate their lecturers according to four attributes such as oral skills and contribution to their professional/general knowledge. The output is an overall score of the lecturer's performance.

GBSG This dataset contains the information of women with breast cancer concerning the status of the tumours and the hormonal system of the patient.

TIPS This dataset concerns the tips given to a waiter in a restaurant collected from customers. Recording variables contains the price of the meal, the tip amount and the conditions of the restaurant meal (number of guests, time of data, etc.).

A.4 Proof of Lemmas

Proof of Lemma 1

Proof. For any $j \in [p]$, $x_j \overset{d}{=} f_j(z_j)$ if and only if (iff) x_j and $f_j(z_j)$ have the same CDF. For each $j \in [p]$, since f_j^{-1} exists for any strictly monotone f_j, we can calculate the CDF of $f_j(z_j)$:

$$F_{f_j(z_j)}(t) = \mathrm{P}(f_j(z_j) \leq t) = \mathrm{P}(z_j \leq f_j^{-1}(t)) = \Phi(f_j^{-1}(t)).$$

Then $x_j \overset{d}{=} f_j(z_j)$ iff $\Phi \circ f_j^{-1} = F_j$, equivalently, $f_j = F_j^{-1} \circ \Phi$. □

Proof of Lemma 2

Proof. It suffices to show for monotone function f, $x \overset{d}{=} f(z)$ iff $f(z) = \text{cutoff}(z; \mathbf{S})$ with $\mathbf{S} = \{s_l = F_z^{-1}\left(\sum_{t=1}^{l} p_t\right) : l \in [k-1]\}$. Notice $x \overset{d}{=} f(z)$ iff the range of $f(z)$ is $[k]$ and $p_l = \mathrm{P}(f(z) = l)$ for any $l \in [k]$. When $f(z) = \text{cutoff}(z; \mathbf{S})$, further define $s_k = \infty$ and $s_0 = -\infty$. Since z is continuous with CDF F_z, it suffices to show:

$$\mathrm{P}(f(z) = l) = \mathrm{P}(s_{l-1} < z \leq s_l) = F_z(s_l) - F_z(s_{l-1}) = p_l, \text{ for } l \in [k]$$

When $x \overset{d}{=} f(z)$, $f(z)$ has range $[k]$. For $l \in [k]$, define $A_l = \{z : f(z) = l\}$, $s_l = \sup_{z \in A_l} z$ and $s_0 = \inf_{z \in A_1} z$. Since $\mathrm{P}(f(z) = l) = p_l > 0$, we have $\inf_{z \in A_l} z < s_l$. Since f is monotone, we have $s_{l-1} \leq \inf_{z \in A_l} z$. Claim $s_{l-1} = \inf_{z \in A_l} z$. If not, there exists $s_{l-1} < z^* < \inf_{z \in A_l} z$ satisfying $(l-1) \leq f(z^*) \leq l$. Since $f(z)$ has range $[k]$, $f(z^*)$ can only be l or $l-1$. Equivalently $z^* \in A_l$ or $z^* \in A_{l-1}$, which contradicts $s_{l-1} < z^* < \inf_{z \in A_l} z$. Thus $s_{l-1} = \inf_{z \in A_l} z$, $f(z) = 1 + \sum_{l=1}^{k-1} \mathbb{1}(z > s_l)$,

$$p_l = \mathrm{P}(f(z) = l) = \mathrm{P}(z \in A_l) = \mathrm{P}(s_{l-1} \leq z \leq s_l) = F_z(s_l) - F_z(s_{l-1}),$$

Thus we have $F_z(s_l) = \sum_{t=1}^{l} p_t \Rightarrow s_l = F_z^{-1}(\sum_{t=1}^{l} p_t)$.

□

Proof of Lemma 3 Before we prove Lemma 3, we introduce the Dvoretzky-Kiefer-Wolfowitz inequality [29], also introduced in [55].

The Dvoretzky-Kiefer-Wolfowitz Inequality. *For any i.i.d. sample $x^1, \ldots, x^n$ with distribution F, then when $\epsilon > 0$,*

$$\mathrm{P}\left(\sup_{t \in \mathbb{R}} |\mathbb{F}_n(t) - F(t)| \geq \epsilon\right) \leq 2e^{-2n\epsilon^2}, \text{ where } \mathbb{F}_n(t) = \frac{\sum_{i=1}^{n} 1\{x^i \leq t\}}{n}$$

Proof. Applying the Dvoretzky-Kiefer-Wolfowitz inequality, for any $\epsilon > 0$, $\Pr(\sup_{t \in \mathbb{R}} |\mathbb{F}_n(t) - F(t)| < \epsilon) \geq 1 - -2e^{-2n\epsilon^2}$.

Take $\epsilon > n^{-1}$, $\sup_{t \in \mathbb{R}} \left|\frac{n}{n+1}\mathbb{F}_n(t) - F(t)\right| < 2\epsilon$. Further let $\epsilon < K_1 \triangleq \min\{\frac{F(m)}{4}, \frac{1-F(M)}{4}\}$, we have $\frac{n}{n+1}\mathbb{F}_n(t) \in [\frac{F(m)}{2}, \frac{1+F(M)}{2}]$ for $t \in [m, M]$. Then,

$$\begin{aligned}
&\sup_{t\in[m,M]} \left|\hat{f}^{-1}(t) - f^{-1}(t)\right| = \sup_{t\in[m,M]} \left|\Phi^{-1}\left(\frac{n}{n+1}\mathbb{F}_n(t)\right) - \Phi^{-1}(F(t))\right| \\
&\leq \sup_{r\in[\frac{F(m)}{2},\frac{1+F(M)}{2}]} \left|\left(\Phi^{-1}(r)\right)'\right| \cdot \sup_{t\in[m,M]} \left|\frac{n}{n+1}\mathbb{F}_n(t) - F(t)\right| \\
&< 2\epsilon \cdot \sup_{r\in[\frac{F(m)}{2},\frac{1+F(M)}{2}]} \left|\left(\Phi^{-1}(r)\right)'\right|
\end{aligned}$$

Since $\left(\Phi^{-1}(r)\right)' = \frac{1}{\phi(\Phi^{-1}(r))}$, we get $\sup_{r\in[\frac{F(m)}{2},\frac{1+F(M)}{2}]} \left|\left(\Phi^{-1}(r)\right)'\right| = K_2 \triangleq 1/\min\left\{\phi\left(\Phi^{-1}(\frac{F(m)}{2})\right), \phi\left(\Phi^{-1}(\frac{F(M)+1}{2})\right)\right\}$. Adjusting the constants, for $2K_2n^{-1} < \epsilon < 2K_1K_2$, we have

$$\mathrm{P}\left(\sup_{t\in[m,M]} \left|\hat{f}^{-1}(t) - f^{-1}(t)\right| > \epsilon\right) \leq 2\exp\left\{-\frac{n\epsilon^2}{2K_2^2}\right\}.$$

□

Proof of Lemma 4 Before we prove Lemma 4, we introduce the Bretagnolle-Huber-Carol inequality introduced in [92].

The Bretagnolle-Huber-Carol Inequality. *If the random vector* $(N_1, \ldots, N_k)$ *is multinomially distributed with parameters n and* $(p_1, \ldots, p_k)$*, then*

$$\mathrm{P}\left(\sum_{i=1}^{k} |N_i/n - p_i| \geq \epsilon\right) \leq 2^k e^{-\frac{1}{2}n\epsilon^2}, \qquad \epsilon > 0.$$

Proof. According to Lemma 2, the cutoff function $f(z) = \text{cutoff}(z; \mathbf{S})$ is unique and $\mathbf{S} = \{s_l : s_l = \Phi^{-1}(\sum_{t=1}^{l} p_t), l \in [k-1]\}$. Define $s_l^* = \Phi^{-1}\left(\frac{\sum_{i=1}^{n} \mathbb{1}(x^i \leq l)}{n}\right)$ for $l \in [k-1]$,

$s_0^* = -\infty$, $s_k^* = \infty$, and $\Delta_l^* = \Phi(s_l^*) - \Phi(s_{l-1}^*) = \sum_{i=1}^n \mathbb{1}(x^i = l)/n$. Notice $(n\Delta_1^*, \ldots, n\Delta_k^*)$ is multinomially distributed with parameters n and $(p_1, \ldots, p_k)$, applying the Bretagnolle-Huber-Carol inequality, for any $\epsilon > 0$, with probability at least $1 - 2^k e^{-\frac{1}{2}n\epsilon^2}$, $\sum_{l=1}^k |\Delta_l^* - p_l| < \epsilon$. First for each $l \in [k]$, $|\Phi(s_l^*) - \Phi(s_l)| \le \sum_{t=1}^k |\Delta_t^* - p_t| < \epsilon$. Take $\epsilon > n^{-1}$, we have

$$\left|\Phi(s_l^*) \cdot \frac{n}{n+1} - \Phi(s_l)\right| \le |\Phi(s_l^*) - \Phi(s_l)| + \frac{\Phi(s_l^*)}{n+1} < 2\epsilon$$

$$\Phi(s_l) - 2\epsilon < \Phi(s_l^*) \cdot \frac{n}{n+1} = \frac{\sum_{i=1}^n \mathbb{1}(x^i \le l)}{n+1} < \Phi(s_l) + 2\epsilon$$

When $l \in [k-1]$, we have $p_1 \le \Phi(s_l) \le \sum_{t=1}^{k-1} p_t$. Further let $\epsilon < K_1 \triangleq \min\{\frac{p_1}{4}, \frac{p_k}{4}\}$, we have $\frac{p_1}{2} \le \Phi(s_l^*) \cdot \frac{n}{n+1} \le 1 - \frac{p_k}{2}$. Thus:

$$\begin{aligned}
\|\hat{\mathbf{S}} - \mathbf{S}\|_1 = \sum_{l=1}^{k-1} |\hat{s}_l - s_l| &= \sum_{l=1}^{k-1} \left|\Phi^{-1}\left(\frac{\sum_{i=1}^n \mathbb{1}(x^i \le l)}{n+1}\right) - \Phi^{-1}(\Phi(s_l))\right| \\
&\le \sup_{r \in [\frac{p_1}{2}, 1-\frac{p_k}{2}]} \left|\left(\Phi^{-1}(r)\right)'\right| \cdot \sum_{l=1}^{k-1} \left|\frac{\sum_{i=1}^n \mathbb{1}(x^i \le l)}{n+1} - \Phi(s_l)\right| \\
&\le \frac{1}{\min\left\{\phi\left(\Phi^{-1}(\frac{p_1}{2})\right), \phi\left(\Phi^{-1}(1-\frac{p_k}{2})\right)\right\}} \cdot 2(k-1)\epsilon
\end{aligned}$$

Let $K_2 = 1/\min\left\{\phi\left(\Phi^{-1}(\frac{p_1}{2})\right), \phi\left(\Phi^{-1}(1-\frac{p_k}{2})\right)\right\}$. Adjusting the constants, for $2(k-1)K_2 n^{-1} < \epsilon < 2(k-1)K_1K_2$, we have

$$\mathrm{P}\left(\|\hat{\mathbf{S}} - \mathbf{S}\|_1| > \epsilon\right) \le 2\exp\left\{-\frac{1}{8K_2^2} \cdot \frac{n\epsilon^2}{(k-1)^2}\right\}.$$

□

APPENDIX B
APPENDIX OF CHAPTER 3

B.1 Proofs

Setup Suppose a p-dimensional vector $\mathbf{x} \sim \mathrm{LRGC}(\mathbf{W}, \sigma^2, \mathbf{f})$ is observed at locations $\mathcal{O} \subset [p]$ and missing at $\mathcal{M} = [p]/\mathcal{O}$. Then according to the definition of LRGC, for $\mathbf{t} \sim \mathcal{N}(\mathbf{0}, \mathrm{I}_k)$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathrm{I}_p)$, and $\mathbf{z} = \mathbf{W}\mathbf{t} + \boldsymbol{\epsilon}$, we know $\mathbf{x} = \mathbf{f}(\mathbf{z})$ and $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ with $\boldsymbol{\Sigma} = \mathbf{W}\mathbf{W}^\top + \sigma^2\mathbf{I}_p$. Here we say two random vectors are equal if they have the same CDF.

A key fact we use is that conditional on known $\mathbf{z}_\mathcal{O}$, $\mathbf{z}_\mathcal{M}$ has a normal distribution:

$$\mathbf{z}_\mathcal{M}|\mathbf{z}_\mathcal{O} \sim \mathcal{N}(\boldsymbol{\Sigma}_{\mathcal{M},\mathcal{O}}\boldsymbol{\Sigma}_{\mathcal{O},\mathcal{O}}^{-1}\mathbf{z}_\mathcal{O}, \boldsymbol{\Sigma}_{\mathcal{M},\mathcal{M}} - \boldsymbol{\Sigma}_{\mathcal{M},\mathcal{O}}\boldsymbol{\Sigma}_{\mathcal{O},\mathcal{O}}^{-1}\boldsymbol{\Sigma}_{\mathcal{O},\mathcal{M}}). \tag{B.1}$$

Here we use $\boldsymbol{\Sigma}_{I,J}$ to denote the submatrix of $\boldsymbol{\Sigma}$ with rows in I and columns in J. Plugging in $\boldsymbol{\Sigma} = \mathbf{W}\mathbf{W}^\top + \sigma^2\mathbf{I}_p$, we obtain

$$\mathrm{E}[\mathbf{z}_\mathcal{M}|\mathbf{z}_\mathcal{O}] = \mathbf{W}_\mathcal{M}\mathbf{W}_\mathcal{O}^\top(\mathbf{W}_\mathcal{O}\mathbf{W}_\mathcal{O}^\top + \sigma^2\mathbf{I})^{-1}\mathbf{z}_\mathcal{O} = \mathbf{W}_\mathcal{M}(\sigma^2\mathbf{I} + \mathbf{W}_\mathcal{O}^\top\mathbf{W}_\mathcal{O})^{-1}\mathbf{W}_\mathcal{O}^\top\mathbf{z}_\mathcal{O}. \tag{B.2}$$

In last equation, we use the Woodbury matrix identity. Similarly, we obtain:

$$\mathrm{Cov}[\mathbf{z}_\mathcal{M}|\mathbf{z}_\mathcal{O}] = \sigma^2\mathbf{I} + \sigma^2\mathbf{W}_\mathcal{M}(\sigma^2\mathbf{I} + \mathbf{W}_\mathcal{O}^\top\mathbf{W}_\mathcal{O})^{-1}\mathbf{W}_\mathcal{M}^\top. \tag{B.3}$$

Proof for Lemma 1 Using the law of total expectation,

$$\begin{aligned}
\mathrm{E}[\mathbf{z}_{\mathcal{M}}|\mathbf{x}_{O}] = \mathrm{E}\left[\mathrm{E}[\mathbf{z}_{\mathcal{M}}|\mathbf{z}_{O}]\middle|\mathbf{x}_{O}\right] &= \mathrm{E}[\mathbf{W}_{\mathcal{M}}(\sigma^2\mathbf{I} + \mathbf{W}_{O}^{\top}\mathbf{W}_{O})^{-1}\mathbf{W}_{O}^{\top}\mathbf{z}_{O}|\mathbf{x}_{O}] \\
&= \mathbf{W}_{\mathcal{M}}(\sigma^2\mathbf{I} + \mathbf{W}_{O}^{\top}\mathbf{W}_{O})^{-1}\mathbf{W}_{O}^{\top}\mathrm{E}[\mathbf{z}_{O}|\mathbf{x}_{O}].
\end{aligned}$$

For the first equality, we use Eq. (B.2). Similarly we can compute the second moments,

$$\begin{aligned}
\mathrm{E}[\mathbf{z}_{\mathcal{M}}\mathbf{z}_{\mathcal{M}}^{\top}|\mathbf{x}_{O}] &= \mathrm{E}\left[\mathrm{E}[\mathbf{z}_{\mathcal{M}}\mathbf{z}_{\mathcal{M}}^{\top}|\mathbf{z}_{O}]\middle|\mathbf{x}_{O}\right] \\
&= \mathrm{E}\left[\mathrm{E}[\mathbf{z}_{\mathcal{M}}|\mathbf{z}_{O}]\mathrm{E}[\mathbf{z}_{\mathcal{M}}|\mathbf{z}_{O}]^{\top} + \mathrm{Cov}[\mathbf{z}_{\mathcal{M}}|\mathbf{z}_{O}]|\mathbf{x}_{O}\right] \\
&= \mathrm{E}\left[\mathrm{E}[\mathbf{z}_{\mathcal{M}}|\mathbf{z}_{O}]\mathrm{E}[\mathbf{z}_{\mathcal{M}}|\mathbf{z}_{O}]^{\top}|\mathbf{x}_{O}\right] + \mathrm{E}\left[\mathrm{Cov}[\mathbf{z}_{\mathcal{M}}|\mathbf{z}_{O}]|\mathbf{x}_{O}\right] \\
&= \mathrm{E}\left[\mathrm{E}[\mathbf{z}_{\mathcal{M}}|\mathbf{z}_{O}]\mathrm{E}[\mathbf{z}_{\mathcal{M}}|\mathbf{z}_{O}]^{\top}|\mathbf{x}_{O}\right] + \mathrm{Cov}[\mathbf{z}_{\mathcal{M}}|\mathbf{z}_{O}]. \qquad \text{(B.4)}
\end{aligned}$$

From the last equation, we use the fact that $\mathrm{Cov}[\mathbf{z}_{\mathcal{M}}|\mathbf{x}_{O}]$ is fully determined by $\mathbf{W}$ and σ^2 and thus does not depend on $\mathbf{x}_{\mathcal{M}}$. Plug Eq. (B.2) and Eq. (B.3) into Eq. (B.4) to obtain

$$\begin{aligned}
&\mathrm{E}\left[\mathrm{E}[\mathbf{z}_{\mathcal{M}}|\mathbf{z}_{O}]\mathrm{E}[\mathbf{z}_{\mathcal{M}}|\mathbf{z}_{O}]^{\top}|\mathbf{x}_{O}\right] = \\
&\qquad \mathbf{W}_{\mathcal{M}}(\sigma^2\mathbf{I} + \mathbf{W}_{O}^{\top}\mathbf{W}_{O})^{-1}\mathbf{W}_{O}^{\top}\mathrm{E}[\mathbf{z}_{O}\mathbf{z}_{O}^{\top}|\mathbf{x}_{O}]\mathbf{W}_{O}(\sigma^2\mathbf{I} + \mathbf{W}_{O}^{\top}\mathbf{W}_{O})^{-1}\mathbf{W}_{\mathcal{M}}^{\top}.
\end{aligned}$$

Then using $\mathrm{Cov}[\mathbf{z}_{\mathcal{M}}|\mathbf{x}_{O}] = \mathrm{E}[\mathbf{z}_{\mathcal{M}}\mathbf{z}_{\mathcal{M}}^{\top}|\mathbf{x}_{O}] - \mathrm{E}[\mathbf{z}_{\mathcal{M}}|\mathbf{x}_{O}]\mathrm{E}[\mathbf{z}_{\mathcal{M}}^{\top}|\mathbf{x}_{O}]$, we have

$$\begin{aligned}
&\mathrm{Cov}[\mathbf{z}_{\mathcal{M}}|\mathbf{x}_{O}] = \sigma^2\mathbf{I}_{|\mathcal{M}|} + \sigma^2\mathbf{W}_{\mathcal{M}}(\sigma^2\mathbf{I} + \mathbf{W}_{O}^{\top}\mathbf{W}_{O})^{-1}\mathbf{W}_{\mathcal{M}}^{\top} + \\
&\qquad \mathbf{W}_{\mathcal{M}}(\sigma^2\mathbf{I} + \mathbf{W}_{O}^{\top}\mathbf{W}_{O})^{-1}\mathbf{W}_{O}^{\top}\mathrm{Cov}[\mathbf{z}_{O}|\mathbf{x}_{O}]\mathbf{W}_{O}(\sigma^2\mathbf{I} + \mathbf{W}_{O}^{\top}\mathbf{W}_{O})^{-1}\mathbf{W}_{\mathcal{M}}^{\top}.
\end{aligned}$$

Proof of Lemma 6

Proof. We use the law of total expectation similar as in Appendix B.1 by first treating $\mathbf{z}_O$ as known. Since $\mathrm{E}[\mathbf{t}|\mathbf{z}_O] = \mathbf{M}_O^{-1}\mathbf{W}_O^\top\mathbf{z}_O$ and $\mathrm{Cov}[\mathbf{t}|\mathbf{z}_O] = \sigma^2\mathbf{M}_O^{-1}$, we have

$$\mathrm{E}[\mathbf{t}|\mathbf{x}_O] = \mathrm{E}[\mathrm{E}[\mathbf{t}|\mathbf{z}_O]|\mathbf{x}_O] = \mathrm{E}[\mathbf{M}_O^{-1}\mathbf{W}_O^\top\mathbf{z}_O|\mathbf{x}_O] = \mathbf{M}_O^{-1}\mathbf{W}_O^\top\mathrm{E}[\mathbf{z}_O|\mathbf{x}_O].$$

Then

$$\begin{aligned}\mathrm{E}[\mathbf{t}(\mathbf{z}_O)^\top|\mathbf{x}_O] &= \mathrm{E}[\mathrm{E}[\mathbf{t}(\mathbf{z}_O)^\top|\mathbf{z}_O]|\mathbf{x}_O] = \mathrm{E}[\mathrm{E}[\mathbf{t}|\mathbf{z}_O](\mathbf{z}_O)^\top|\mathbf{x}_O] \\ &= \mathbf{M}_O^{-1}\mathbf{W}_O^\top\mathrm{E}[\mathbf{z}_O(\mathbf{z}_O)^\top|\mathbf{x}_O] = \mathbf{M}_O^{-1}\mathbf{W}_O^\top\left(\mathrm{Cov}[\mathbf{z}_O|\mathbf{x}_O] + \mathrm{E}[\mathbf{z}_O|\mathbf{x}_O]\mathrm{E}[(\mathbf{z}_O)^\top|\mathbf{x}_O]\right) \\ &= \mathbf{M}_O^{-1}\mathbf{W}_O^\top\mathrm{Cov}[\mathbf{z}_O|\mathbf{x}_O] + \mathrm{E}[\mathbf{t}|\mathbf{x}_O]\mathrm{E}[(\mathbf{z}_O)^\top|\mathbf{x}_O].\end{aligned}$$

and

$$\begin{aligned}\mathrm{E}[\mathbf{t}(\mathbf{t})^\top|\mathbf{x}_O] &= \mathrm{E}[\mathrm{E}[\mathbf{t}(\mathbf{t})^\top|\mathbf{z}_O]|\mathbf{x}_O] = \mathrm{E}[\mathbf{M}_O^{-1}\mathbf{W}_O^\top\mathbf{z}_O(\mathbf{z}_O)^\top\mathbf{W}_O\mathbf{M}_O^{-1}|\mathbf{x}_O] + \mathrm{E}[\mathrm{Cov}[\mathbf{t}|\mathbf{z}_O]|\mathbf{x}_O] \\ &= \mathbf{M}_O^{-1}\mathbf{W}_O^\top\mathrm{E}[\mathbf{z}_O(\mathbf{z}_O)^\top|\mathbf{x}_O]\mathbf{W}_O\mathbf{M}_O^{-1} + \mathrm{E}[\sigma^2\mathbf{M}_O^{-1}|\mathbf{x}_O] \\ &= \mathbf{M}_O^{-1}\mathbf{W}_O^\top\left(\mathrm{Cov}[\mathbf{z}_O|\mathbf{x}_O] + \mathrm{E}[\mathbf{z}_O|\mathbf{x}_O]\mathrm{E}[(\mathbf{z}_O)^\top|\mathbf{x}_O]\right)\mathbf{W}_O\mathbf{M}_O^{-1} + \sigma^2\mathbf{M}_O^{-1} \\ &= \mathbf{M}_O^{-1}\mathbf{W}_O^\top\mathrm{Cov}[\mathbf{z}_O|\mathbf{x}_O]\mathbf{W}_O\mathbf{M}_O^{-1} + \mathrm{E}[\mathbf{t}|\mathbf{x}_O]\mathrm{E}[(\mathbf{t})^\top|\mathbf{x}_O] + \sigma^2\mathbf{M}_O^{-1}.\end{aligned}$$

□

Proof of Theorem 3 To prove Theorem 3, we introduce a lemma which provides a concentration inequality on quadratic forms of sub-Gaussian vectors. For a detailed treatment of sub-Gaussian random distributions, see [95]. A random variable $x \in \mathbb{R}$ is called sub-Gaussian if $(\mathrm{E}[|x|^p])^{1/p} \leq K\sqrt{p}$ for all

$p \geq 1$ with some $K > 0$. The sub-Gaussian norm of x is defined as $\|x\|_{\psi_2} = \sup_{p\geq 1} p^{-1/2}(\mathrm{E}[|x|^p])^{1/p}$.

Denote the inner product of vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ as $\langle \mathbf{x}_1, \mathbf{x}_2 \rangle$. A random vector $\mathbf{x} \in \mathbb{R}^p$ is called sub-Gaussian if the one-dimensional marginals $\langle \mathbf{x}, \mathbf{a} \rangle$ are all sub-Gaussian random variables for any constant vector $\mathbf{a} \in \mathbb{R}^p$. The sub-Gaussian norm of $\mathbf{x}$ is defined as $\|\mathbf{x}\|_{\psi_2} = \sup_{\mathbf{a}\in\mathbb{S}^{p-1}} \|\langle \mathbf{x}, \mathbf{a}\rangle\|_{\psi_2}$. A Gaussian random vector is also sub-Gaussian.

Lemma 2. Let $\Sigma \in \mathbb{R}^{p\times p}$ be a positive semidefinite matrix. Let $\mathbf{x} = (x_1, \ldots, x_p)$ be a sub-Gaussian random vector with mean zero and covariance matrix $\mathbf{I}_p$. For all $t > 0$,

$$\Pr\left[\mathbf{x}^\top \Sigma \mathbf{x} > (\sqrt{\mathrm{tr}(\Sigma)} + \sqrt{2\lambda_1(\Sigma)t})^2\right] \leq e^{-t}.$$

Our Lemma 2 is Lemma 17 in [97], which is also a simplified version of Theorem 1 in [47].

Proof. Since $\mathbf{f}$ is elementwise Lipschitz with constant L,

$$\mathrm{MSE}(\hat{\mathbf{x}}) = \frac{\|\mathbf{f}_{\mathcal{M}}(\mathbf{z}_{\mathcal{M}}) - \mathbf{f}_{\mathcal{M}}(\hat{\mathbf{z}}_{\mathcal{M}})\|_2^2}{\|\mathcal{M}\|} \leq L^2 \frac{\|\mathbf{z}_{\mathcal{M}} - \hat{\mathbf{z}}_{\mathcal{M}}\|_2^2}{\|\mathcal{M}\|}. \tag{B.5}$$

Denote the covariance matrix of $\mathbf{z}_{\mathcal{M}}$ conditional on $\mathbf{z}_{\mathcal{O}}$ as $\mathbf{\Sigma}_{(\mathcal{M})}$. Apply the above inequality with $\mathbf{\Sigma} = \mathbf{\Sigma}_{(\mathcal{M})}$ and $\mathbf{x} = \mathbf{\Sigma}_{(\mathcal{M})}^{-1/2}\mathbf{z}_{\mathcal{M}}$, we obtain:

$$\Pr\left(\|\mathbf{z}_{\mathcal{M}} - \hat{\mathbf{z}}_{\mathcal{M}}\|_2^2 > \left(\sqrt{\mathrm{tr}(\mathbf{\Sigma}_{(\mathcal{M})})} + \sqrt{2\lambda_1(\mathbf{\Sigma}_{(\mathcal{M})})t}\right)^2\right) \leq e^{-t}. \tag{B.6}$$

Notice

$$\begin{aligned}
\operatorname{tr}(\Sigma_{(\mathcal{M})}) &= \operatorname{tr}\left(\sigma^2\mathbf{I} + \sigma^2\mathbf{W}_{\mathcal{M}}(\sigma^2\mathbf{I} + \mathbf{W}_{O}^{\top}\mathbf{W}_{O})^{-1}\mathbf{W}_{\mathcal{M}}^{\top}\right) \\
&= \sigma^2|\mathcal{M}| + \sigma^2\operatorname{tr}\left((\sigma^2\mathbf{I} + \mathbf{W}_{O}^{\top}\mathbf{W}_{O})^{-1}\mathbf{W}_{\mathcal{M}}^{\top}\mathbf{W}_{\mathcal{M}}\right) \\
&\leq \sigma^2|\mathcal{M}| + \sigma^2\lambda_1(\sigma^2\mathbf{I} + \mathbf{W}_{O}^{\top}\mathbf{W}_{O})^{-1})\operatorname{tr}\left(\mathbf{W}_{\mathcal{M}}^{\top}\mathbf{W}_{\mathcal{M}}\right) \\
&= \sigma^2|\mathcal{M}| + \sigma^2\frac{1}{\sigma^2 + \lambda_k^2(\mathbf{W}_O)}(1-\sigma^2)|\mathcal{M}|.
\end{aligned}$$

In the inequality, we use the fact $\operatorname{tr}(\mathbf{AB}) \leq \lambda_1(\mathbf{A})\operatorname{tr}(\mathbf{B})$ for any real symmetric positive semidefinite matrices $\mathbf{A}$ and $\mathbf{B}$. In the last equation, we use the unit diagonal constraints of $\mathbf{W}\mathbf{W}^{\top}+\sigma^2\mathbf{I}_p$ such that $\operatorname{tr}(\mathbf{W}_{\mathcal{M}}^{\top}\mathbf{W}_{\mathcal{M}}) = \operatorname{tr}(\mathbf{W}_{\mathcal{M}}\mathbf{W}_{\mathcal{M}}^{\top}) = |\mathcal{M}|(1-\sigma^2)$. Also notice

$$\begin{aligned}
\lambda_1(\Sigma_{(\mathcal{M})}) &= \lambda_1(\sigma^2\mathbf{I} + \sigma^2\mathbf{W}_{\mathcal{M}}(\sigma^2\mathbf{I} + \mathbf{W}_{O}^{\top}\mathbf{W}_{O})^{-1}\mathbf{W}_{\mathcal{M}}^{\top}) \\
&\leq \sigma^2 + \sigma^2\lambda_1(\mathbf{W}_{\mathcal{M}}(\sigma^2\mathbf{I} + \mathbf{W}_{O}^{\top}\mathbf{W}_{O})^{-1}\mathbf{W}_{\mathcal{M}}^{\top}) \\
&\leq \sigma^2 + \sigma^2\lambda_1^2(\mathbf{W}_{\mathcal{M}})\lambda_1((\sigma^2\mathbf{I} + \mathbf{W}_{O}^{\top}\mathbf{W}_{O})^{-1}) \\
&= \sigma^2 + \sigma^2\frac{\lambda_1^2(\mathbf{W}_{\mathcal{M}})}{\sigma^2 + \lambda_k^2(\mathbf{W}_O)}.
\end{aligned}$$

Thus,

$$\|\mathbf{z}_{\mathcal{M}} - \hat{\mathbf{z}}_{\mathcal{M}}\|_2^2 \leq \sigma^2|\mathcal{M}| \cdot \left(\sqrt{1 + \frac{1-\sigma^2}{\sigma^2 + \lambda_k^2(\mathbf{W}_O)}} + \sqrt{\left(1 + \frac{\lambda_1^2(\mathbf{W}_{\mathcal{M}})}{\sigma^2 + \lambda_k^2(\mathbf{W}_O)}\right)\frac{2t}{|\mathcal{M}|}}\right)^2. \tag{B.7}$$

Combining Eq. (B.5), Eq. (B.6) and Eq. (B.7), we finish the proof. □

Proof of Corollary 1 We first introduce a result from [95, Theorem 5.39] characterizing the singular values of long random matrices with independent sub-

Gaussian rows.

Lemma 3. Let $\mathbf{A} \in \mathbb{R}^{p\times k}$ be a matrix whose rows $\mathbf{a}_j$ are independent sub-Gaussian random vectors in $\mathbb{R}^k$ whose covariance matrix is $\mathbf{\Sigma}$. Then for every $t > 0$, with probability as least $1 - 2\exp(-ct^2)$ one has

$$\lambda_1\left(\frac{1}{p}\mathbf{A}^\top\mathbf{A} - \mathbf{\Sigma}\right) \leq \max(\delta, \delta^2)\lambda_1(\mathbf{\Sigma}), \quad \text{where } \delta = C\sqrt{\frac{k}{p}} + \frac{t}{\sqrt{p}}.$$

Here c, $C > 0$ depend only on the subgaussian norm $K = \max_j \|\Sigma^{-1/2}\mathbf{a}_j\|_{\psi_2}$.

Proof. Apply Lemma 3 to submatrix $\mathbf{W}_O$ and $\mathbf{W}_\mathcal{M}$ respectively with covariance matrix $\mathbf{\Sigma} = \frac{1-\sigma^2}{k}\mathbf{I}_k$, we obtain with probability at least $1-2\exp(-ct_1^2)-2\exp(-ct_2^2)$,

$$\left|\frac{1}{|O|}\lambda_k^2(\mathbf{W}_O) - \frac{1-\sigma^2}{k}\right| \leq \frac{1-\sigma^2}{k}\epsilon_1 \text{ and } \left|\frac{1}{|\mathcal{M}|}\lambda_1^2(\mathbf{W}_\mathcal{M}) - \frac{1-\sigma^2}{k}\right| \leq \frac{(1-\sigma^2)\epsilon_2}{k},$$

where $\epsilon_1 = \max(\delta_1, \delta_1^2)$ with $\delta_1 = \frac{C\sqrt{k}+t_1}{\sqrt{|O|}}$ and $\epsilon_2 = \max(\delta_2, \delta_2^2)$ with $\delta_2 = \frac{C\sqrt{k}+t_2}{\sqrt{|\mathcal{M}|}}$. Constants c, $C > 0$ only depend on the subgaussian norm $\max_j \|\sqrt{\frac{k}{1-\sigma^2}}\mathbf{w}_j\|_{\psi_2}$.

For any $0 < \epsilon < 1$, let $t_1 = \frac{\epsilon\sqrt{|O|}}{2}$ and $t_2 = \frac{\epsilon\sqrt{|O|}}{2\sqrt{c_2}}$. Suppose the sufficiently large constant c_1 satisfies $c_1 > \frac{4C^2\max(1,c_2)}{\epsilon^2}$. Then we have

$$\epsilon_1 = \delta_1 = \frac{C}{\sqrt{|O|/k}} + \frac{t}{\sqrt{|O|}} < \frac{C}{\sqrt{c_1}} + \frac{\epsilon}{2} < \frac{C}{\sqrt{4C^2/\epsilon^2}} + \frac{\epsilon}{2} = \epsilon,$$

and

$$\epsilon_2 = \delta_2 = \frac{C}{\sqrt{|\mathcal{M}|/k}} + \frac{t}{\sqrt{|\mathcal{M}|}} < \frac{C}{\sqrt{|O|/c_2 k}} + \frac{\epsilon}{2} < \frac{C}{\sqrt{4C^2/\epsilon^2}} + \frac{\epsilon}{2} = \epsilon.$$

Thus we have with probability at least $1-2\exp(-c\epsilon^2|\mathcal{O}|/4)-2\exp(-c\epsilon^2|\mathcal{O}|/4c_2)$,

$$\lambda_k^2(\mathbf{W}_{\mathcal{O}}) > (1-\sigma^2)(1-\epsilon)\frac{|\mathcal{O}|}{k} \quad \text{and} \quad \lambda_1^2(\mathbf{W}_{\mathcal{M}}) \leq (1-\sigma^2)(1+\epsilon)\frac{|\mathcal{M}|}{k}. \tag{B.8}$$

Combining Eq. (B.7) and Eq. (B.8), then with probability at least $1-\exp(-t)-2\exp(-c\epsilon^2|\mathcal{O}|/4)-2\exp(-c\epsilon^2|\mathcal{O}|/4c_2)$,

$$\begin{aligned}\frac{\|\mathbf{z}_{\mathcal{M}}-\hat{\mathbf{z}}_{\mathcal{M}}\|_2^2}{|\mathcal{M}|} &\leq \sigma^2\left(\sqrt{1+\frac{1}{\frac{\sigma^2}{1-\sigma^2}+(1-\epsilon)|\mathcal{O}|/k}}+\sqrt{\frac{2t}{|\mathcal{M}|}+\frac{2(1+\epsilon)t}{\frac{k\sigma^2}{1-\sigma^2}+(1-\epsilon)|\mathcal{O}|}}\right)^2\\ &\leq \sigma^2\left(\sqrt{1+\frac{1}{\frac{\sigma^2}{1-\sigma^2}+(1-\epsilon)|\mathcal{O}|/k}}+\sqrt{\frac{2c_2t}{|\mathcal{O}|}+\frac{2(1+\epsilon)t}{\frac{k\sigma^2}{1-\sigma^2}+(1-\epsilon)|\mathcal{O}|}}\right)^2.\end{aligned} \tag{B.9}$$

Now take $t=\log|\mathcal{O}|$, with fixed k and σ^2, the right hand side is $1+O\left(\sqrt{\frac{\log|\mathcal{O}|}{|\mathcal{O}|}}\right)$.

Notice $|\mathcal{O}| > c_1k \geq c_1$. Then there exists some constant $c_3 > 0$ such that $|\mathcal{O}|$ satisfies:

$$\log|\mathcal{O}| < c_3\frac{c\epsilon^2|\mathcal{O}|}{4\max(1,c_2)}$$

thus Eq. (B.9) holds with probability at least $1-\frac{1+2c_3}{|\mathcal{O}|}$. Combing the result with Eq. (B.5) completes the proof. □

B.2 Additional experiments

B.2.1 LRGC imputation under correct model

For LRGC imputation, we show the random variation of the error (due to error in the estimate of $\mathbf{z}_{\mathcal{M}}$) dominates the estimation error (due to errors in the esti-

Table B.1: Imputation error (NRMSE) on synthetic continuous data over 20 repetitions.

Setting	`LRGC`	`LRGC-Oracle`
Low Rank	0.347(.004))	.330(.004)
High Rank	0.517(.011)	.433(.007)

mates of the parameters $\mathbf{W}$ and σ). To do so, we compare the imputation error of LRGC imputation with estimated model parameters (`LRGC`) and true model parameters (`LRGC-Oracle`). For ordinal data, imputation requires approximating truncated normal moments, which may blur the improvement of using true model parameters. Thus we conduct the comparison on the same continuous synthetic dataset described in Section 4. The results are reported in Table B.1.

Compared to `LRGC`, `LRGC-Oracle` only improves slightly (1%) over low rank data. Thus the model estimation error is dominated by the random variation of the imputation error. For high rank data, the improvement (8%) is still small compared to the gap between LRGC imputation and LRMC algorithms ($\geq$ 18%). Also notice the marginal transformation $g_j(z) = z^3$ for high rank data is not Lipschitz, so the theory presented in this paper does not bound the LRGC imputation error.

The result here indicates there is still room to improve LRGC imputation when the marginals are not Lipschitz. We leave that important work for the future.

B.2.2 Imputation error versus reliability shape with varying number of ordinal levels

We show in this section the imputation error versus reliability curve shape on ordinal data with many ordinal levels will match that on continuous data. The results here indicate the prediction power of `LRGC` reliability depends on the imputation task. The prediction power is larger for easier imputation task. In the synthetic experiments, imputing continuous data is harder than imputing ordinal data, and imputing 1-5 ordinal data is harder than imputing binary data.

We follow the synthetic experiments setting used in Section 4, but vary the number of ordinal levels to $\{5, 8, 10\}$. We adopt high SNR setting for ordinal data and low rank setting for continuous data. To make the imputation error comparable between continuous data and ordinal data, we measure the ratio of the imputation error over the $m\%$ entries to the imputation error over all missing entries. Shown in Fig. B.1, the curve shape for low rank continuous data is similar to that for ordinal data with 5–8 levels. Also notice, NRMSE for continuous data involves the observed data values while MAE does not, which may cause small difference in the curve shape.

B.3 Experimental details

Synthetic data To select the best value of the key tuning parameter for each method, we first run some initial experiments to determine a proper range such that the best value lies strictly inside that range.

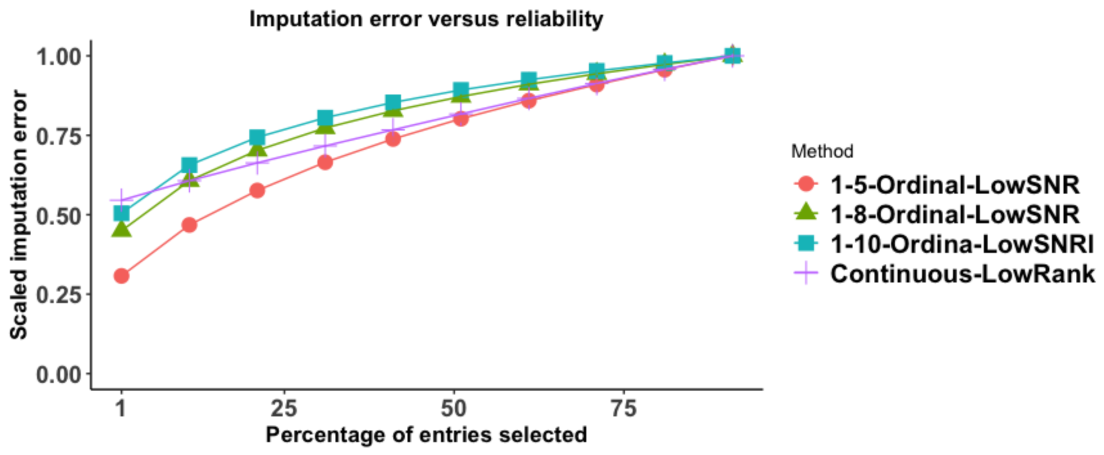


Figure B.1: Imputation error on the subset of $m\%$ most reliable entries, reported over 5 repetitions.

For `LRGC`, the only tuning parameter is rank. We find that a range of 6 – 14 for continuous data (both low and high rank), and 3 – 11 for ordinal data with 5 levels and binary data (high SNR and low SNR), suffices to ensure the best value is strictly inside the range. Notice this range is still quite small, so it is rather easy to search over.

For `softImpute`, the only tuning parameter is the penalization parameter. As suggested by the vignette of the R package [42], we first center the rows and columns of the observations using the function `biScale()` and then compute λ_0 as an upper bound on the penalization parameter using the function `lambda0()`. The penalization parameter range is set as the exponentially decaying path between λ_0 and $\lambda_0/100$ with nine points for all cases:

```
exp(seq(from=log(lam0),to=log(lam0/100),length=9)).
```

We found increasing the path length from 9 to 20 only slightly improves the performance (up to .01 across all cases) on best performance on test set.

Table B.2: Run time (in seconds) for synthetic data at the best tuning parameter; mean (variance) reported over 20 repetitions.

Continuous	LRGC	PPCA	softImpute	MMMF-ℓ_2	MMC
Low Rank	5.7(0.2)	2.9(0.4)	0.7(0.0)	3.3(1.0)	457.9(10.4)
High Rank	6.5(0.3)	0.3(0.1)	1.1(0.2)	7.6(2.0)	554.4(32.0)
1-5 ordinal	LRGC	PPCA	softImpute	MMMF-BvS	MMMF-ℓ_1
High SNR	27.2(0.7)	1.0(0.2)	1.2(0.1)	19.2(1.5)	17.4(1.2)
Low SNR	19.8(0.8)	0.3(0.1)	1.3(0.0)	17.4(1.5)	17.0(1.4)
Binary	LRGC	PPCA	MMMF-hinge	MMMF-logistic	MMMF-ℓ_1
High SNR	66.7(3.0)	0.9(0.5)	3.8(0.3)	4.4(0.6)	2.1(0.3)
Low SNR	52.0(4.4)	0.3(0.1)	3.4(0.4)	3.3(0.4)	1.9(0.2)

For `MMMF`, there are two tuning parameters: the rank and the penalization parameter. We set the rank to be allowed maximum rank 199. We set the range of penalization parameter as we do for `softImpute`, with left and right endpoints that depend on the data. For `MMMF`-ℓ_2 on continuous data and `MMMF-BvS` on ordinal data, we use $\lambda_0/4$ as start point and $\lambda_0/100$ as end point. For all other `MMMF` methods, we use λ_0 as start point and $\lambda_0/100$ as end point.

For `MMC`, following the authors' suggestions regarding the code, we use the following settings: (1) the number of gradient steps used to update the Z matrix is 1; (2) the tolerance parameter is set as 0.01; In addition, we set the initial rank as 50, the increased rank at each step as 5, the maximum rank as 199, the maximum number of iterations as 80 and the Lipshitz constant as 10. Finally, the key tuning parameter we search over is the constant step size as suggested by the authors of [36]. The range is set as $\{3, 5, 7, \ldots, 17, 19\}$.

The complete results are plotted in Fig. B.2. Clearly, `LRGC` does not overfit even for high ranks, across all settings. We also provide the runtime for each method at the best tuning parameter in Table B.2. Notice our current implementation is written entirely in R, and thus further acceleration is possible.

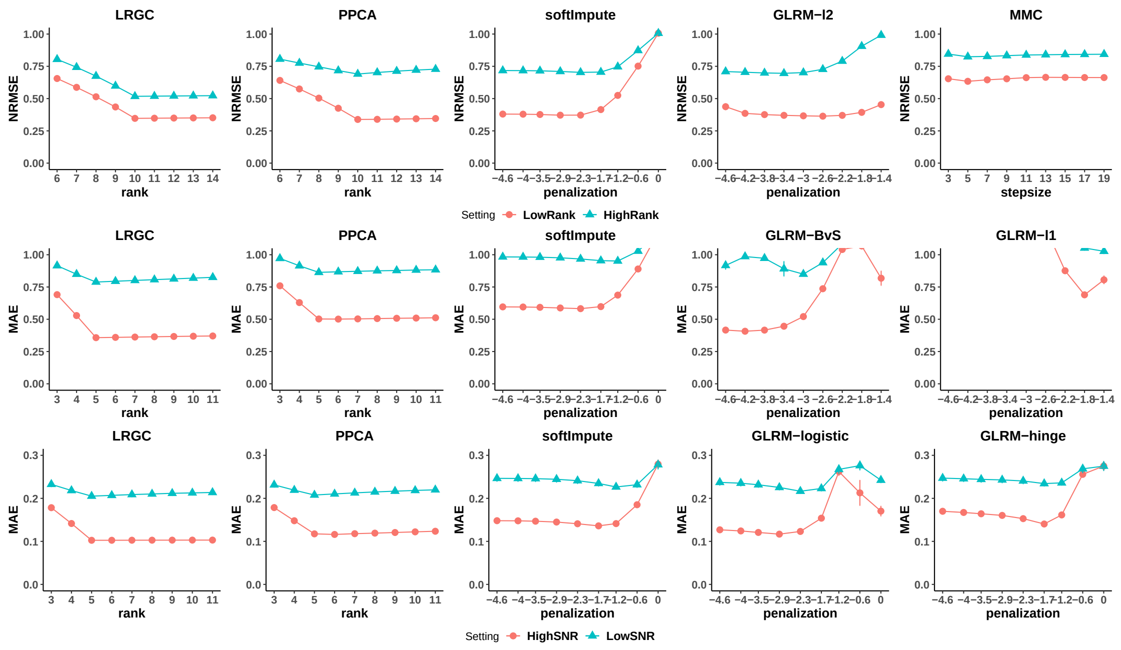


Figure B.2: Imputation error over a key tuning parameter reported over 20 repetitions. The error bars ara invisible. The penalization parameter λ is plotted over the log-ratios $\log(\alpha)$ which satisfies $\lambda = \alpha\lambda_0$.

MovieLens 1M The dataset can be found at `https://grouplens.org/datasets/movielens/1m/`. Similar to the synthetic experiments, we choose the tuning parameter for each method on a proper range determined through some initial experiments. For `LRGC`, we choose the rank from $\{8, 10, 12, 14\}$ to be 10. With λ_0 calculated as for the synthetic data, for `softImpute`, we select the penalization parameter from $\{\frac{\lambda_0}{2}, \frac{\lambda_0}{4}, \frac{\lambda_0}{6}, \frac{\lambda_0}{8}\}$ to be $\frac{\lambda_0}{4}$; for `MMMF-BvS`, we set the rank as 200 and select the penalization parameter from $\{\frac{\lambda_0}{10}, \frac{\lambda_0}{12}, \frac{\lambda_0}{14}, \frac{\lambda_0}{16}, \frac{\lambda_0}{18}\}$ to be $\frac{\lambda_0}{14}$.

APPENDIX C

APPENDIX OF CHAPTER 4

C.1 Proofs

Proof of Theorem 2

Proof. Theorem 2 is an immediate consequence of the normality of $\mathbf{z}_{\mathcal{M}}$ conditional on $\mathbf{z}_O = \mathbf{f}_O^{-1}(\mathbf{x}_O)$ (see Eq. (B.1)) and the elementwise strictly monotone $\mathbf{f}$. □

Proof of Theorem 3 Suppose $\mathbf{x} = (x_1, \ldots, x_p)$ where x_j is ordinal with $k_j(\geq 2)$ ordinal levels encoded as $\{1, \ldots, k_j\}$ for $j \in [p]$. For ordinal data, the conditional distribution of $\mathbf{z}_{\mathcal{M}}|\mathbf{z}_O \in \mathbf{f}_O^{-1}(\mathbf{x}_O)$ is intractable. Consequently, we cannot establish distribution-based confidence intervals for $\mathbf{z}_{\mathcal{M}}$.

Instead, for each marginal j, we can lower bound the probability of event $|\hat{x}_j - x_j| \leq d$ for the Gaussian copula imputation $\hat{x}_j$ and $d \in \mathbb{Z}$. Since $\Pr(|\hat{x}_j - x_j| \leq k_j - 1) = 1$, it suffices to consider $d < k_j - 1$. In practice, the result is more useful for small d, such as $d = 0$. Let us first state a generalization ofTheorem 3.

Theorem 4. Suppose $\mathbf{x} \sim \mathrm{GC}(\Sigma, \mathbf{f})$ with observations $\mathbf{x}_O$ and missing entries $\mathbf{x}_{\mathcal{M}}$. Also for each marginal $j \in [p]$, x_j takes values from $\{1, \ldots, k_j\}$ and thus the f_j is a step function with cut points $\mathbf{S}_j = \{s_1, \ldots, s_{k_j-1}\}$:

$$f_j(z) = 1 + \sum_{k=1}^{k_j-1} \mathbb{1}(z > s_k), \quad \text{where } -\infty =: s_0 < s_1 < \ldots < s_{k_j-1} < s_{k_j} := \infty.$$

For a missing entry x_j, $j \in \mathcal{M}$, the set of values for z_j that would yield the same imputed value $\hat{x}_j = f_j(\mathrm{E}[z_j|\mathbf{x}_O])$ is $f_j^{-1}(\hat{x}_j) = (s_{\hat{x}_j-1}, s_{\hat{x}_j}]$. Then the following holds:

$$\Pr(|\hat{x}_j - x_j| \leq d) \geq 1 - \frac{\mathrm{Var}[z_j|\mathbf{x}_O]}{d_j^2},$$

with

$$d_j = \min(|\mathrm{E}[z_j|\mathbf{x}_O] - s_{\max(\hat{x}_j-1-d,0)}|, |\mathrm{E}[z_j|\mathbf{x}_O] - s_{\min(\hat{x}_j+d,k_j)}|),$$

where $\mathrm{E}[z_j|\mathbf{x}_O], \mathrm{Var}[z_j|\mathbf{x}_O]$ are given in Lemma 1 with $\mathcal{M}$ replaced by j.

Proof. The proof applies to each missing dimension $j \in \mathcal{M}$. Let us further define $s_k = -\infty$ for any negative integer k and $s_k = \infty$ for any integer $k > k_j$ for convenience. Then $s_k = s_{\max(k,0)}$ for negative integer k and $s_k = s_{\min(k,k_j)}$ for integer k larger than k_j.

First notice $|\hat{x}_j - x_j| \leq d$ if and only if $z_j \in (s_{\hat{x}_j-1-d}, s_{\hat{x}_j+d}]$ for the latent normal z_j satisfying $x_j = f_j(z_j)$. Specifically, when $d = 0$, $\hat{x}_j = x_j$ if and only if $z_j \in (s_{\hat{x}_j-1}, s_{\hat{x}_j}]$, i.e. $f_j^{-1}(x_j) = (s_{\hat{x}_j-1}, s_{\hat{x}_j}] = f_j^{-1}(\hat{x}_j)$. Notice we have,

$$\mathrm{E}[z_j|\mathbf{x}_O] \in (s_{\hat{x}_j-1}, s_{\hat{x}_j}] \subset (s_{\hat{x}_j-1-d}, s_{\hat{x}_j+d}].$$

Thus a sufficient condition for $z_j \in (s_{\hat{x}_j-1-d}, s_{\hat{x}_j+d}]$ is that z_j is sufficiently close to its conditional mean $\mathrm{E}[z_j|\mathbf{x}_O]$. More precisely,

$$|\mathrm{E}[z_j|\mathbf{x}_O] - z_j| \leq \min(|\mathrm{E}[z_j|\mathbf{x}_O] - s_{\hat{x}_j-1-d}|, |\mathrm{E}[z_j|\mathbf{x}_O] - s_{\hat{x}_j+d}|) \to |\hat{x}_j - x_j| \leq d.$$

Define $d_j := \min(|\mathrm{E}[z_j|\mathbf{x}_O] - s_{\hat{x}_j-1-d}|, |\mathrm{E}[z_j|\mathbf{x}_O] - s_{\hat{x}_j+d}|)$. Notice when $d = 0$,

$$d_j = \min(|\mathrm{E}[z_j|\mathbf{x}_O] - s_{\hat{x}_j-1}|, |\mathrm{E}[z_j|\mathbf{x}_O] - s_{\hat{x}_j}|) = \min_{s\in\mathbf{S}} |\mathrm{E}[z_j|\mathbf{x}_O] - s|.$$

Use the Markov inequality together with the conditional distribution of z_j given $\mathbf{x}_O$ to bound

$$\Pr(|\mathrm{E}[z_j|\mathbf{x}_O] - z_j| > d_j) \leq \frac{\mathrm{Var}[z_j|\mathbf{x}_O]}{d_j^2},$$

which completes our proof. □

APPENDIX D

APPENDIX OF CHAPTER 5

D.1 Proofs

Proof of Lemma 7

Proof. First note

$$\begin{aligned}
Q(\Sigma; \Sigma^t, \{\mathbf{x}^i_{O_i}\}_{t\in S_t}) =& \frac{1}{|S_t|}\sum_{i\in S_t}\mathbb{E}[\ell(\Sigma; \{\mathbf{z}^i, \mathbf{x}^i_{O_i}\}_{i\in S_t})|\mathbf{x}^i_{O_i}, \Sigma^{t-1}, \hat{\mathbf{f}}]\\
=& \frac{1}{|S_t|}\sum_{i\in S_t}\mathbb{E}\left[c - \frac{\log|\Sigma|}{2} - \frac{(\mathbf{z}^i)^\top\Sigma^{-1}\mathbf{z}^i}{2}\middle|\mathbf{x}^i_{O_i}, \Sigma^{t-1}, \hat{\mathbf{f}}\right]\\
=& \mathbb{E}\left[c - \frac{\log|\Sigma| + \operatorname{tr}\left(\Sigma^{-1}\frac{1}{|S_t|}\sum_{i\in S_t}\mathbf{z}^i(\mathbf{z}^i)^\top\right)}{2}\middle|\mathbf{x}^i_{O_i}, \Sigma^{t-1}, \hat{\mathbf{f}}\right]\\
=& c - \frac{\log|\Sigma| + \operatorname{tr}\left(\Sigma^{-1}\mathbb{E}\left[\frac{1}{|S_t|}\sum_{i\in S_t}\mathbf{z}^i(\mathbf{z}^i)^\top|\mathbf{x}^i_{O_i}, \Sigma^{t-1}, \hat{\mathbf{f}}\right]\right)}{2}
\end{aligned}$$

where $c > 0$ is a constant.

Denote

$$E_t = \mathbb{E}\left[\frac{\sum_{i\in S_t}\mathbf{z}^i(\mathbf{z}^i)^\top}{|S_t|}|\mathbf{x}^i_{O_i}, \Sigma^{t-1}, \hat{\mathbf{f}}\right]$$

It is easy to show through induction that $Q_t(\Sigma)$ can be written as

$$Q_t(\Sigma) = \sum_{l=1}^{t}\alpha^t_l Q(\Sigma; \Sigma^{l-1}, \{\mathbf{x}^i\}_{i\in S_l}),$$

with $\sum_{l=1}^t \alpha_l^t = 1$ and $\alpha_l^t > 0$. Thus we have

$$Q_t(\Sigma) = c - \frac{\log|\Sigma| + \mathrm{tr}(\Sigma^{-1} \sum_{l=1}^t \alpha_l^t E_t)}{2}$$

Then solving $\mathrm{argmax}_\Sigma\, Q_t(\Sigma)$ is the classical problem of the MLE of Gaussian covariance matrix, which yields $\Sigma^t = \mathrm{argmax}\, Q_t(\Sigma) = \sum_{l=1}^t \alpha_l^t E_l$, when $\sum_{l=1}^t \alpha_l^t E_l$ is positive definite. Since we require $|S_l| > p$, we have E_l as positive definite matrix and thus $\sum_{\ell=1}^t \alpha_l^t E_t$ is also positive definite.

For the first data batch, Σ^1 is estiamed as in the offline setting: $\Sigma^1 = E_1$ wit initial estimate Σ^0, thus we set $\gamma_0 = 1$ to satisfy $\Sigma^{t+1} = (1-\gamma_t)\Sigma^t + \gamma_t E_{t+1}$ for $t = 0$. For any $t > 1$ and $\gamma_t \in (0,1)$, note by the definition of $Q_t(\Sigma)$:

$$\alpha_l^{t+1} = \alpha_l^t(1-\gamma_t), \text{ for } l = 1, \ldots, t, \text{ and } \alpha_{t+1}^{t+1} = \gamma_t.$$

then

$$\begin{aligned}\Sigma^{t+1} &= \sum_{l=1}^{t+1} \alpha_l^{t+1} E_l = \sum_{l=1}^{t} \alpha_l^{t+1} E_l + \alpha_{t+1}^{t+1} E_{t+1} \\ &= \sum_{l=1}^{t} \alpha_l^t (1-\gamma_t) E_l + \gamma_t E_{t+1} = (1-\gamma_t)\Sigma^t + \gamma_t E_{t+1}.\end{aligned}$$

which finishes the proof. □

Proof of Theorem 4

We first formally define some concepts, and then rigorously restate Theorem 4 with complete description. In the proof, we show our Theorem 4 is a special case of Theorem 1 in [18] by verifying the their required assumptions are satisfied in our setting.

Distribution function for mixed data For a mixed data vector $\mathbf{x} = (\mathbf{x}_C, \mathbf{x}_\mathcal{D})$ with $\mathbf{x}_C$ as continuous random variables and $\mathbf{x}_\mathcal{D}$ as ordinal random variables, we use the notion of distribution function for $\mathbf{x}$ as $f(\mathbf{x}) = f(\mathbf{x}_C)\mathrm{P}(\mathbf{x}_\mathcal{D}) \in \mathbb{R}$, with $f(\mathbf{x}_C)$ as the PDF of $\mathbf{x}_C$ and $\mathbf{P}(\mathbf{x}_\mathcal{D})$ as the PMF (probability mass function) of $\mathbf{x}_\mathcal{D}$.

Distribution over incomplete data Let $\tilde{\mathbf{x}} = (\tilde{x}_1, ..., \tilde{x}_p) = (\tilde{\mathbf{x}}_O, \tilde{\mathbf{x}}_\mathcal{M})$ be a underlying complete vector that is observed at $O \subset [p]$, $\mathbf{m}$ be the associated observed-data indicator vector: $\mathbf{m} = (m_1, .., m_p)$ where $m_j = 1$ if $\tilde{x}_j$ is observed ($j \in O$) and $m_j = 0$ if $\tilde{x}_j$ is missing ($j \in \mathcal{M}$). Also define $\mathbf{x} = (x_1, ..., x_p)$ be the incomplete version of $\tilde{\mathbf{x}}$ with a special category `NA` at missing locations: $x_j = \tilde{x}_j$ if $m_j = 1$ and $x_j = \mathtt{NA}$ if $m_j = 0$. Denote the deterministic mapping from $(\tilde{\mathbf{x}}, \mathbf{m})$ to $\mathbf{x}$ as $T(\tilde{\mathbf{x}}, \mathbf{m}) = \mathbf{x}$.

Once we are given an incomplete data vector $\mathbf{x}$, the actual observation is $(\mathbf{x}_O, \mathbf{m})$. Our goal is to learn the distribution associated with the underlying complete vector $\tilde{\mathbf{x}}$ instead of the distribution of $\mathbf{x}$, since the latter also requires characterizing the distribution of $\mathbf{m}$. Under the missing at random (MAR) assumption, we have

$$\begin{aligned} f(\mathbf{x}_O, \mathbf{m}) &= \int f(\mathbf{x}_O, \mathbf{x}_\mathcal{M})\mathbf{P}(\mathbf{m}|\mathbf{x}_O, \mathbf{x}_\mathcal{M})d\mathbf{x}_\mathcal{M} \\ &= \int f(\mathbf{x}_O, \mathbf{x}_\mathcal{M})d\mathbf{x}_\mathcal{M}\mathbf{P}(\mathbf{m}|\mathbf{x}_O) = f(\mathbf{x}_O)\mathbf{P}(\mathbf{m}|\mathbf{x}_O). \end{aligned}$$

To distinguish a few definitions, there is a distribution $\pi^*(\tilde{\mathbf{x}})$ for the true underlying complete vector $\tilde{\mathbf{x}}$, a (joint) distribution $\pi(\mathbf{x})$ over the observed entries $\mathbf{x}_O$ and the missing locations $\mathbf{m}$, and a (marginal) distribution $\pi(\mathbf{x}_O)$ over the observed entries $\mathbf{x}_O$. There is a one-to-one correspondence between the complete data distribution $\pi^*(\tilde{\mathbf{x}})$ and the observed data distribution $\pi(\mathbf{x}_O)$, since $\pi(\mathbf{x}_O)$ is the marginal distribution of $\pi^*(\tilde{\mathbf{x}})$ over dimensions O. The joint distribution $\pi(\mathbf{x}) = \pi(\mathbf{x}_O)\mathbf{P}(\mathbf{m}|\mathbf{x}_O)$, further requires the conditional distribution of $\mathbf{m}|\mathbf{x}_O$, which is unknown.

When we say true data distribution over the observed entries, we refer to the (marginal) distribution $\pi(\mathbf{x}_O)$. With a Gaussian copula model $\mathrm{GC}(\Sigma, \mathbf{f})$, we denote the underlying complete distribution as $g^*_\Sigma(\tilde{\mathbf{x}})$, and the distribution of observed data as $g_\Sigma(\mathbf{x}_O)$. We further construct the joint distribution over $(\mathbf{x}_O, \mathbf{m})$ as $g_\Sigma(\mathbf{x})$, using the same conditional distribution $\mathbf{P}(\mathbf{m}|\mathbf{x}_O)$ as in $\pi(\mathbf{x})$, for the purpose of proof. We ignore the dependence on $\mathbf{f}$ because it is kept fixed during EM iterations, while Σ is updated at each iteration. Define $d(x, A) = \inf(y \in A, |x - y|)$ with $|\cdot|$ as the ℓ_2 norm. Now we are ready to restate our Theorem 4.

Theorem 5. Let $\pi(\mathbf{x}_O)$ be the distribution function of the true data-generating distribution of the observations and $g_\Sigma(\mathbf{x}_O)$ be the distribution function of the observed data from $\mathrm{GC}(\Sigma, \mathbf{f})$, assuming data is missing at random (MAR). Let $\mathcal{L} = \{\Sigma \in S^p_{++} : \nabla_\Sigma \mathrm{KL}(\pi \| g_\Sigma) = 0\}$ be the set of stationary points of $\mathrm{KL}(\pi \| g_\Sigma)$ for a fixed $\mathbf{f}$. Under the following conditions,

1. $\mathbf{f}$ remains unchanged across EM iterations; for all continuous dimensions j, the range of f_j^{-1} is a subset of $[-C, C]$ for some $C > 0$; for each ordinal dimension j, the step function f_j only has finite number of steps, i.e. $f_j(z_j)$ has finite number of ordinal levels.
2. The step-sizes $\gamma_t \in (0, 1)$ satisfy $\sum_{t=1}^{\infty} \gamma_t^2 < \sum_{t=1}^{\infty} \gamma_t = \infty$.

3. With probability 1, $\limsup|\Sigma_t| < \infty$ and $\liminf\{d(\Sigma_t, (S^p_{++})^c)\} > 0$.
4. Let $\mathbf{X}_t$ (size $|S_t| \times p$) denote the data observation in the t-th batch with points i.i.d. $\pi(\mathbf{x})$, and $\mathbf{Z}_t$ as the latent data matrix corresponding to $\mathbf{X}_t$. For the set $\Gamma = \{s \in S^p_{++} : \mathrm{E}_\pi[\mathrm{E}_{\Sigma=s}[\mathbf{Z}_t^\top \mathbf{Z}_t|\mathbf{X}_t]] = s\}$ and $w(\Sigma) := \mathrm{KL}(\pi||g_\Sigma)$, $w(\Gamma)$ is nowhere dense.

then the iterates Σ^t produced by our online EM (Algorithm 5) satisfies that $\lim_{t\longrightarrow\infty} d(\Sigma^t, \mathcal{L}) = 0$ with probability 1.

Proof. We first formally state our latent model: the observed variables and the latent variables, as well as the conditional distribution of the latent given the observed. Then we show our stated convergence result is a special case of the result in Theorem 1 of [18]. The following proof consists of showing our latent model satisfies the assumptions required in Theorem 1 of [18], and thus our results hold according to Theorem 1 of [18].

The employed latent model We treat our defined $\mathbf{x}$ (with `NA` at missing entries) as the "observed variables" in our latent model. Since $\mathbf{x} = T(\tilde{\mathbf{x}}, \mathbf{m})$ and there exists a Gaussian latent variable $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \Sigma)$ such that $\tilde{\mathbf{x}} = \mathbf{f}(\mathbf{z})$, we treat $(\mathbf{z}, \mathbf{m})$ as our "latent variables". Conditional on known $\mathbf{x}$, the distribution of $\mathbf{m}$ reduces to a single point, denoted as $\mathbf{m}_\mathbf{x}$. The distribution of $\mathbf{z}$ is $\mathcal{N}(0, \Sigma)$ truncated to the region $\{\mathbf{z} : T(\mathbf{f}(\mathbf{z}), \mathbf{m}_\mathbf{x}) = \mathbf{x}\}$ for $\tilde{\mathbf{x}}$. With slight abuse of notation, we write $\{\mathbf{z} : T(\mathbf{f}(\mathbf{z}), \mathbf{m}_\mathbf{x}) = \mathbf{x}\}$ as $\mathbf{f}^{-1}(\mathbf{x})$ and $f_j^{-1}(\mathtt{NA}) = \mathbb{R}$ for any $j \in [p]$. Now note

$$\mathrm{KL}(\pi(\mathbf{x})||g_\Sigma(\mathbf{x})) = \mathrm{E}_{\mathbf{x}_O,\mathbf{m}} \log \frac{\pi(\mathbf{x}_O)\mathbf{P}(\mathbf{m}|\mathbf{x}_O)}{g_\Sigma(\mathbf{x}_O)\mathbf{P}(\mathbf{m}|\mathbf{x}_O)} = \mathrm{KL}(\pi(\mathbf{x}_O)||g_\Sigma(\mathbf{x}_O)).$$

Thus our stated result matches Theorem 1 of [18]: the online EM estimate converges to the set of stationary points of KL divergence between the true data distribution and the learned data distribution, on "observed variables" $\mathbf{x}$.

Verification of Assumptions 1 in [18] For simplicity, assume each data batch has n points. We drop the time index on the data points and simply write the data points in each batch as $\{\mathbf{x}^i\}_{i=1,\ldots,n}$ or the corresponding matrix $\mathbf{X}$. Also denote the latent data as $\{\mathbf{z}^i, \mathbf{m}^i\}_{i=1,\ldots,n}$ and the corresponding matrix data $\mathbf{Z}, \mathbf{M}$. The complete data likelihood for a batch is

$$L(\Sigma; \{\mathbf{x}^i, \mathbf{z}^i, \mathbf{m}^i\}) = ce^{\log|\Sigma| + \frac{1}{n}\operatorname{tr}(\Sigma^{-1}\mathbf{Z}^\top\mathbf{Z})},$$

where c is a constant w.r.t. Σ:

$$c = \prod_{i=1}^{n} 1(T(\mathbf{f}(\mathbf{z}^i), \mathbf{m}^i) = \mathbf{x}^i) f(\mathbf{m}^i)(2\pi)^{-\frac{np}{2}} e^{-\frac{n}{2}}.$$

Thus it belongs to the exponential family, with sufficient statistics $s = \frac{1}{n}\mathbf{Z}^\top\mathbf{Z}$, $\psi(\Sigma) = -\log|\Sigma|$ and $\phi(\Sigma) = \Sigma^{-1}$. Thus Assumption 1a is satisfied.

Denote the sample space for Σ as the set of all positive definite symmetric matrices: $\Theta = S^p_{++}$. The function $\bar{s}(\mathbf{X}; \Sigma) := \mathrm{E}_\Sigma[\mathbf{Z}^\top\mathbf{Z}|\mathbf{X}]$ is well defined for all $\mathbf{X}$ and all $\Sigma \in \Theta$. To see why, note $\mathrm{E}[z_j|x_j]$ and $\mathrm{Var}[z_j|x_j]$ have finite closed form expression for any x_j, and thus $\mathrm{E}[z_j^2|x_j]$ has finite closed form expression for any x_j for all $j \in [p]$. Then for any $i, j \in [p]$, $\mathrm{E}[z_i z_j|\mathbf{x}]$ is finite and thus well defined using Cauchy inequality. Thus Assumption 1b is satisfied.

Let $\mathcal{S} = S^p_{++}$, then clearly $\mathcal{S}$ is a convex open subset of all symmetric matrices.

For any $\Sigma, \Sigma' \in \mathcal{S}$, any $\mathbf{X}$ and $\gamma \in [0, 1)$, we have $\bar{s}(\bar{X}; \Sigma)$ as positive semidefinite matrix and thus $(1-\gamma)\Sigma' + \gamma\bar{s}(\mathbf{X}; \Sigma) \in \mathcal{S}$. In our situation, $\ell(s; \Sigma) = \log|\Sigma| + \mathrm{tr}(\Sigma^{-1}s)$. Note solving $\max_{\Sigma\in\mathcal{S}} \ell(s; \Sigma)$ is equivalent to solving the MLE of multivariate normal covariance. By classical results, for any $s \in \mathcal{S}$, $\ell(s; \Sigma)$ has a unique global maximum over Θ at $\Sigma = s$, denoted as $\bar{\Sigma}(s) = s$. Thus Assumption 1c is satisfied.

Verification of Assumptions 2 in [18] For (2a), in our situation, $\psi(\Sigma) = -\log|\Sigma|$ and $\phi(\Sigma) = \Sigma^{-1}$ are clearly twice continuous differentiable in $\Theta = S^p_{++}$. For (2b), $\bar{\Sigma}(s)$ is simply the identity function and thus continuously differentiable in $\mathcal{S}$. For (2c), first note

$$\bar{s}(\mathbf{X}; \bar{\Sigma}(s)) = \bar{s}(\mathbf{X}; s) = \mathrm{E}_{\Sigma=s}[\mathbf{Z}^\top\mathbf{Z}|\mathbf{X}]$$

Now we bound the max-norm of $\mathrm{E}_{\Sigma=s}[\mathbf{Z}^\top\mathbf{Z}|\mathbf{X}]$ uniformly over all possible $\mathbf{X}$ for a given s. To do so, it suffices to bound the max-norm of $\mathrm{E}_{\Sigma=s}[\mathbf{z}^i(\mathbf{z}^i)^\top|\mathbf{x}^i]$ for a single point i. We ignore i for notation simplicity. It suffices to bound the max-norm of the diagonal entries, since we can bound the max-norm all off-diagonal entries using the max-norm of the diagonal entries through Cauchy-Schwarz inequality. Now if x_j is an observed continuous entry, by regularity condition on $\mathbf{f}$, we have $z_j = f_j^{-1}(x_j) \in [-C, C]$ and thus $\mathrm{E}_{\Sigma=s}[z_j^2|x_j]$ is finite. If x_j is a missing entry, we have $\mathrm{E}_{\Sigma=s}[z_j^2|x_j] = s_{jj}^2$, the (j, j)-th entry of s. At last if x_j is an observed ordinal entry, we have

$$\begin{aligned}\mathrm{E}_{\Sigma=s}[z_j^2|x_j] =& \frac{\int_{z_j\in f_j^{-1}(x_j)} z_j^2\phi(z_j; 0, s_{jj}^2)dz_j}{\int_{z_j\in f_j^{-1}(x_j)} \phi(z_j; 0, s_{jj}^2)dz_j} \\ \le& C_j \int_{z_j\in f_j^{-1}(x_j)} z_j^2\phi(z_j; 0, s_{jj}^2)dz_j \\ \le& C_j \int_{z_j\in\mathbb{R}} z_j^2\phi(z_j; 0, s_{jj}^2)dz_j = C_j s_{jj}^2.\end{aligned}$$

where $C_j = \frac{1}{\min_j \int_{z_j \in f_j^{-1}(x_j)} \phi(z_j; 0, s_{jj}^2) dz_j}$. Note C_j is finite and depends only on the step-wise function f_j which has only finite number of steps. Thus we can uniformly bound the max-norm of diagonal entries of $\mathrm{E}_{\Sigma=s}[z_j^2|x_j]$ using diagonal entries of s. For all compact subsets $\mathcal{K} \subset \mathcal{S}$, and for all $s \in \mathcal{K}$, the diagonal entries of s are bounded and thus $\mathrm{E}_{\Sigma=s}[\mathbf{Z}^\top \mathbf{Z}|\mathbf{X}]$ are bounded. In other words, we can bound $\sup_{s\in\mathcal{K}} |\bar{s}(\mathbf{X}; \bar{\Sigma}(s))|$ uniformly over $\mathbf{X}$ for a given $\mathcal{K}$. Using similar arguments, for any $k > 2$, we can bound $\sup_{s\in\mathcal{K}} |\bar{s}(\mathbf{X}; \bar{\Sigma}(s))|^k$ uniformly over $\mathbf{X}$ for a given $\mathcal{K}$ and thus $\mathrm{E}_\pi(\sup_{s\in\mathcal{K}} |\bar{s}(\mathbf{X}; \bar{\Sigma}(s))|^k) < \infty$ for fixed $\mathbf{f}$.

Now we have verified all the assumptions of Theorem 1 in [18] which we do not include as our assumptions, and thus finishes the proof. □

Discussion on our assumptions Our assumption 1 is easily satisfied in practice. Once with access to moderate number of data points, once can pre-compute $\mathbf{f}$ and fix it among EM iterations. Our experiments show 200 data points can provide good performance. In practice, one use the scaled empirical CDF to estimate f^{-1}, which ensures f_j^{-1} to be a subset of $[-C, C]$ for sufficiently large C (depending on data size n). Also it is reasonable to model all ordinal variables to have finite number of ordinal levels, since we can only observed finite number of levels in practice.

Our assumptions 2-4 follow the assumptions in Theorem 1 of [18]. Assumption 2 is standard for decreasing step size stochastic approximation and $\gamma_t = c/t$ with some constant $c > 0$ satisfies the condition. Assumption 3 corresponds to a stability assumption which is not trivial. In practice, we enforce the stability by projecting the estimated covariance matrix to a correlation matrix.

	Offline Sim (5000×15)	Movielens (6027×207)
Offline EM	188(1), 87(2)	1690(9), 781(2)
Minibatch EM	48(1), 28(0)	252(2), 142(2)
Online EM	52(0), 34(1)	269(3), 169(11)

Table D.1: Mean(sd) runtime of Gaussian copula methods for offline datasets over 10 trials. In each cell, the runtime with 2 cores follows that with 1 core.

D.2 Additional experiments

D.2.1 Acceleration of parallelism

Here we report the acceleration achieved using parallelism for Gaussian copula methods. We only report the runtime comparison on offline datasets, since the parallelism does not influence the algorithm accuracy. We use 2 cores to implement the parallelism. The results in Appendix D.2.1 show the parallelism brings considerable speedups for all Gaussian copula algorithms.

D.2.2 Robustness to varying data dimension, missing ratio and missing mechanism

We add experiments under MAR and MNAR and also experiments using missing ratios, number of samples, and variable dimensions in our online synthetic experiments. The original setting has $p = 15$ variables, 6000 samples in total ($n = 2000$ samples for each distribution period), and 40% missing entries under MCAR. We vary each of these three setups: n, p, and missing ratio. We also design MNAR such that larger values have smaller missing probabilities, shown as in Table D.2.

Table D.2: A MNAR mechanism used. For each variable, the missing probability p of an entry z solely depends on its own value. Entries with smaller values have high missing probabilities.

Variable type	$p = 0.2$	$p = 0.4$	$p = 0.6$
Continuous	$z > 75\%$ quantile	$z \in (25\%, 75\%)$ quantiles	$z < 25\%$ quantile
Ordinal	$z \in \{4, 5\}$	$z \in \{3\}$	$z \in \{1, 2\}$
Binary	$z = 1$	NA	$z = 0$

Table D.3: Mean(sd) for imputation error for additional synthetic online data experiments w.r.t. different missing ratios over 10 trials.

Method	Cont	Ord	Bin
	20% missing		
OnlineEM	**.76(.08)**	**.81(.10)**	**.63(.12)**
OnlineKFMC	.94(.06)	1.08(.38)	.79(.15)
GROUSE	1.17(.06)	1.70(.36)	1.12(.11)
	40% missing		
OnlineEM	**.84(.07)**	**.89(.07)**	**.72(.11)**
OnlineKFMC	.98(.06)	.17(.46)	.88(.12)
GROUSE	1.20(.07)	1.80(.40)	1.10(.06)
	60% missing		
OnlineEM	**.91(.05)**	**.96(.04)**	**.83(.09)**
OnlineKFMC	1.02(.07)	1.32(.59)	.96(.10)
GROUSE	1.31(.14)	2.09(.49)	1.12(.06)

The results in Table D.3, Table D.4, Table D.5 and Table D.6 record the scaled MAE (SMAE) of the imputation estimator, as used in Table 5.1. They show our method is actually robust to violated missing mechanisms and various number of samples, dimensions and missing ratios.

Table D.4: Mean(sd) for imputation error for additional synthetic online data experiments w.r.t. different number of data points over 10 trials.

Method	Cont	Ord	Bin
		n = 1000	
OnlineEM	**.87(.08)**	**.90(.09)**	**.75(.13)**
OnlineKFMC	1.01(.08)	1.29(.51)	.95(.13)
GROUSE	1.22(.09)	1.80(.45)	1.12(.06)
		n = 2000	
OnlineEM	**.84(.07)**	**.89(.07)**	**.72(.11)**
OnlineKFMC	.98(.06)	1.17(.46)	.88(.12))
GROUSE	1.20(.07)	1.80(.40)	1.10(.06)
		n = 3000	
OnlineEM	**.83(.06)**	**.88(.07)**	**.70(.10)**
OnlineKFMC	.97(.05)	1.10(.38)	.83(.13)
GROUSE	1.20(.06)	1.80(.34)	1.11(.06)

Table D.5: Mean(sd) for imputation error for additional synthetic online data experiments w.r.t. different data dimensions over 10 trials.

Method	Cont	Ord	Bin
		p = 15	
OnlineEM	**.84(.07)**	**.89(.07)**	**.72(.11)**
OnlineKFMC	.98(.06)	1.17(.46)	.88(.12)
GROUSE	1.20(.07)	1.80(.40)	1.10(.06)
		p = 30	
OnlineEM	**.85(.07)**	**.90(.07)**	**.74(.10)**
OnlineKFMC	.98(.05)	**.89(.11)**	.98(.06)
GROUSE	1.13(.05)	1.12(.06)	1.14(.05)
		p = 45	
OnlineEM	**.86(.07)**	**.92(.08)**	**.76(.09)**
OnlineKFMC	.98(.05)	.98(.05)	.98(.05)
GROUSE	1.13(.06)	1.13(.06)	1.12(.06)

Table D.6: Mean(sd) for imputation error for additional synthetic online data experiments w.r.t. different missing mechanisms over 10 trials.

Method	Cont	Ord	Bin
		MCAR	
OnlineEM	**.84(.07)**	**.89(.07)**	**.72(.11)**
OnlineKFMC	.98(.06)	1.17(.46)	.88(.12)
GROUSE	1.20(.07)	1.80(.40)	1.10(.06)
		MNAR	
OnlineEM	**.89(.07)**	**.99(.07)**	.73(.05)
OnlineKFMC	.93(.05)	1.22(.54)	.70(.11)
GROUSE	1.46(.07)	1.95(.40)	**.66(.04)**

APPENDIX E

APPENDIX OF CHAPTER 7

Proof of Theorem 5

Proof. Denote $\Pr(\operatorname{argmax}_{i=0,1,...,K} z_i = k) = p_k(\boldsymbol{\mu})$ for $k = 0, 1, ..., K$ and $\mathrm{P}(\boldsymbol{\mu}) = (p_1(\boldsymbol{\mu}), ..., p_K(\boldsymbol{\mu}))$. Also define e_k: $e_k \in \mathbb{R}^K$ has 1 at coordinate k and zero elsewhere.

We prove existence by contradiction. Suppose there is no satisfying $\boldsymbol{\mu}$. Let

$$\boldsymbol{\mu}^* = \operatorname*{argmin}_{\boldsymbol{\mu}} f(\boldsymbol{\mu}), \text{ where } f(\boldsymbol{\mu}) = \sum_{k=1}^{K} |p_k(\boldsymbol{\mu}) - p_k|. \tag{E.1}$$

We first consider the scenario that $p_k(\boldsymbol{\mu}^*) \leq p_k$ for all $k = 1, .., K$. Then there exists at least one positive i such that $p_i(\boldsymbol{\mu}^*) < p_i$. Since $p_i(\boldsymbol{\mu}^* + \lambda e_i)$ is a continuous function w.r.t. λ and $\lim_{\lambda\to\infty} p_i(\boldsymbol{\mu}^* + \lambda e_i) = 1$, there exists a $\lambda_0 > 0$ such that $p_i(\boldsymbol{\mu}^* + \lambda_0 e_i) = p_i$. Note $p_k(\boldsymbol{\mu}^* + \lambda e_i)$ is strictly monotonic decreasing w.r.t. λ for any $k \neq i$. Thus for $p_k(\boldsymbol{\mu}^* + \lambda_0 e_i) = p_k(\boldsymbol{\mu}^*) - \delta_k$, we have $\delta_k > 0$ when $k \neq i$ and $\delta_i = p_i(\boldsymbol{\mu}^*) - p_i < 0$. Now, we have

$$\begin{aligned}
\sum_{k=1}^{K} |p_k(\boldsymbol{\mu}^* + \lambda_0 e_i) - p_k| &= |p_i(\boldsymbol{\mu}^* + \lambda_0 e_i) - p_i| + \sum_{k\neq i} |p_k(\boldsymbol{\mu}^* + \lambda_0 e_i) - p_k| \\
&= \sum_{k\neq i} (p_k - p_k(\boldsymbol{\mu}^* + \lambda_0 e_i)) = \sum_{k\neq i} (p_k - p_k(\boldsymbol{\mu}^*) + \delta_k) \\
&= \sum_{k\neq i} (p_k - p_k(\boldsymbol{\mu}^*)) + \sum_{k\neq i} \delta_k = f(\boldsymbol{\mu}^*) - (p_i - p_i(\boldsymbol{\mu}^*)) + \sum_{k\neq i} \delta_k \\
&= f(\boldsymbol{\mu}^*) + \sum_{k\neq 0} \delta_k
\end{aligned}$$

Note $\sum_{k=0}^{K} \delta_k = 0$ and $\delta_0 > 0$, we know

$$f(\boldsymbol{\mu}^* + \lambda_0 e_i) = f(\boldsymbol{\mu}^*) - \delta_0 < f(\boldsymbol{\mu}^*),$$

which contradicts our assumption. The contradiction can also be shown similarly if $p_k(\boldsymbol{\mu}^*) \geq p_k$ for all $k = 1, .., K$.

Now the remaining scenario is that both I and I^c are nonempty, where $I = \{i | p_i(\boldsymbol{\mu}^*) < p_i, i \in [K]\}$ and $I^c = [K] - I$. Similarly, we pick a $i \in I$, then there exists a $\lambda_0 > 0$ such that $p_i(\boldsymbol{\mu}^* + \lambda_0 e_i) = p_i$, and we define δ_i same as before. Now

$$\begin{aligned}
\sum_{k=1}^{K} |p_k(\boldsymbol{\mu}^* + \lambda_0 e_i) - p_k| &= |p_i(\boldsymbol{\mu}^* + \lambda_0 e_i) - p_i| + \sum_{k \neq i} |p_k(\boldsymbol{\mu}^* + \lambda_0 e_i) - p_k| \\
&= \sum_{k \in I - \{i\}} |p_k - p_k(\boldsymbol{\mu}^*) + \delta_k| + \sum_{k \in I^c} |p_k - p_k(\boldsymbol{\mu}^*) + \delta_k| \\
&= \sum_{k \in I - \{i\}} (p_k - p_k(\boldsymbol{\mu}^*) + \delta_k) + \sum_{k \in I^c} |p_k - p_k(\boldsymbol{\mu}^*) + \delta_k| \\
&\leq \sum_{k \in I - \{i\}} (p_k - p_k(\boldsymbol{\mu}^*) + \delta_k) + \sum_{k \in I^c} |p_k - p_k(\boldsymbol{\mu}^*)| + \sum_{k \in I^c} \delta_k \\
&= \sum_{k \in I - \{i\}} |p_k - p_k(\boldsymbol{\mu}^*)| + \sum_{k \in I^c} (p_k - p_k(\boldsymbol{\mu}^*)) + \sum_{k \neq i} \delta_k \\
&= \sum_{k \neq i} |p_k - p_k(\boldsymbol{\mu}^*)| + \sum_{k \neq i} \delta_k \\
&= f(\boldsymbol{\mu}^*) - (p_i - p_i(\boldsymbol{\mu}^*)) + \sum_{k \neq i} \delta_k \\
&= f(\boldsymbol{\mu}^*) + \sum_{k \neq 0} \delta_k
\end{aligned}$$

And thus we still have

$$f(\boldsymbol{\mu}^* + \lambda_0 e_i) = f(\boldsymbol{\mu}^*) - \delta_0 < f(\boldsymbol{\mu}^*),$$

which contradicts our assumption and completes our proof for existence.

Now for uniqueness, assume there exists $\boldsymbol{\mu} \neq \tilde{\boldsymbol{\mu}}$ such that $\mathbf{P}(\boldsymbol{\mu}) = \mathbf{P}(\tilde{\boldsymbol{\mu}})$. Define $I_<, I_=, I_>$ to be the set of entries that $\boldsymbol{\mu}$ is smaller, equal, and larger than $\tilde{\boldsymbol{\mu}}$, respectively. We want to show it muse be the case that both $I_<$ and $I_>$ are empty, which contradicts the assumption.

First, if $I_<$ is empty but $I_>$ is not, then for each $i \in I_>$, we define $\boldsymbol{\mu}_i \in \mathbb{R}^k$ such that $\boldsymbol{\mu}_i$ agrees with $\boldsymbol{\mu}$ at all entries but i and agrees with $\tilde{\boldsymbol{\mu}}$ at entry i. Since $p_i(\boldsymbol{\mu})$ is strictly monotonically increasing w.r.t. μ_i when fixing $\boldsymbol{\mu}_{-i}$, we know $p_i(\boldsymbol{\mu}_i) < p_i(\boldsymbol{\mu})$. Further repeatedly switching one more entry of $\boldsymbol{\mu}_i$ in $I_>$ from $\boldsymbol{\mu}$ to $\tilde{\boldsymbol{\mu}}$ until $I_>$ is exhausted, we have $p_i(\tilde{\boldsymbol{\mu}}_i) < p_i(\boldsymbol{\mu})$, which contradicts the assumption. Similarly we can show the contradiction if $I_<$ is not empty but $I_>$ is empty.

Now consider the case that both $I_<$ and $I_>$ are not empty. Define $\boldsymbol{\mu}^*$ such that $\boldsymbol{\mu}^*$ agrees with $\boldsymbol{\mu}$ over $I_<$ and agrees with $\tilde{\boldsymbol{\mu}}$ over $I_>$. According to above, we know for each $i \in I_>$, $p_i(\boldsymbol{\mu}) > p_i(\boldsymbol{\mu}^*) > p_i(\tilde{\boldsymbol{\mu}})$, which contradicts the assumption. Thus we complete our proof. □

BIBLIOGRAPHY

[1] Ryan Prescott Adams and David JC MacKay. Bayesian online changepoint detection. *arXiv preprint arXiv:0710.3742*, 2007.

[2] Samaneh Aminikhanghahi and Diane J Cook. A survey of methods for time series change point detection. *Knowledge and information systems*, 51 (2):339–367, 2017.

[3] Clifford Anderson-Bergman, Tamara G Kolda, and Kina Kincher-Winoto. Xpca: Extending pca for a combination of discrete and continuous variables. *arXiv preprint arXiv:1808.07510*, 2018.

[4] Vincent Audigier, François Husson, and Julie Josse. A principal component method to impute missing values for mixed data. *Advances in Data Analysis and Classification*, 10(1):5–26, 2016.

[5] Vincent Audigier, François Husson, and Julie Josse. Mimca: multiple imputation for categorical variables with multiple correspondence analysis. *Statistics and computing*, 27(2):501–518, 2017.

[6] Haim Avron, Satyen Kale, Shiva Kasiviswanathan, and Vikas Sindhwani. Efficient and practical stochastic subgradient descent for nuclear norm regularization. *arXiv preprint arXiv:1206.6384*, 2012.

[7] Laura Balzano, Robert Nowak, and Benjamin Recht. Online identification and tracking of subspaces from highly incomplete information. In *2010 48th Annual allerton conference on communication, control, and computing (Allerton)*, pages 704–711. IEEE, 2010.

[8] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

[9] Manjunath BG and Stefan Wilhelm. Moments calculation for the double truncated multivariate normal density. *Available at SSRN 1472153*, 2009.

[10] Sonia A Bhaskar. Probabilistic low-rank matrix completion from quantized measurements. *The Journal of Machine Learning Research*, 17(1):2131–2164, 2016.

[11] Zdravko I Botev. The normal law under linear restrictions: simulation and estimation via minimax tilting. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 79(1):125–148, 2017.

[12] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[13] David S Bunch. Estimability in the multinomial probit model. *Transportation Research Part B: Methodological*, 25(1):1–12, 1991.

[14] S van Buuren and Karin Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in r. *Journal of statistical software*, pages 1–68, 2010.

[15] Emmanuel J Candes and Yaniv Plan. Matrix completion with noise. *Proceedings of the IEEE*, 98(6):925–936, 2010.

[16] Emmanuel J Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717, 2009.

[17] Wei Cao, Dong Wang, Jian Li, Hao Zhou, Yitan Li, and Lei Li. Brits: bidirectional recurrent imputation for time series. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 6776–6786, 2018.

[18] Olivier Cappé and Eric Moulines. On-line expectation–maximization algorithm for latent data models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(3):593–613, 2009.

[19] Alexandra Carpentier, Olga Klopp, and Matthias Löffler. Constructing confidence sets for the matrix completion problem. In *Conference of the International Society for Non-Parametric Statistics*, pages 103–118. Springer, 2016.

[20] Alexandra Carpentier, Olga Klopp, Matthias Löffler, Richard Nickl, et al. Adaptive confidence sets for matrix completion. *Bernoulli*, 24(4A):2429–2460, 2018.

[21] Yuxin Chen, Jianqing Fan, Cong Ma, and Yuling Yan. Inference and uncertainty quantification for noisy matrix completion. *Proceedings of the National Academy of Sciences*, 116(46):22931–22937, 2019.

[22] Benjamin Christoffersen. *mdgc: Missing Data Imputation Using Gaussian Copulas*, 2021. URL `https://CRAN.R-project.org/package=mdgc`. R package version 0.1.5.

[23] Benjamin Christoffersen, Mark Clements, Keith Humphreys, and Hedvig Kjellström. Asymptotically exact and fast gaussian copula models for imputation of mixed data types. In *Asian Conference on Machine Learning*, pages 870–885. PMLR, 2021.

[24] Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision support systems*, 47(4):547–553, 2009.

[25] Ruifei Cui, Ioan Gabriel Bucur, Perry Groot, and Tom Heskes. A novel bayesian approach for latent variable modeling from mixed data with missing values. *Statistics and Computing*, 29(5):977–993, 2019.

[26] Mark A Davenport, Yaniv Plan, Ewout Van Den Berg, and Mary Wootters. 1-bit matrix completion. *Information and Inference: A Journal of the IMA*, 3 (3):189–223, 2014.

[27] Anthony Christopher Davison and David Victor Hinkley. *Bootstrap methods and their application*. Number 1. Cambridge university press, 1997.

[28] Charanpal Dhanjal, Romaric Gaudel, and Stéphan Clémençon. Online matrix completion through nuclear norm regularisation. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pages 623–631. SIAM, 2014.

[29] Aryeh Dvoretzky, Jack Kiefer, Jacob Wolfowitz, et al. Asymptotic minimax character of the sample distribution function and of the classical multinomial estimator. *The Annals of Mathematical Statistics*, 27(3):642–669, 1956.

[30] Jianqing Fan, Han Liu, Yang Ning, and Hui Zou. High dimensional semiparametric latent graphical model for mixed data. *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, 79(2):405–421, 2017.

[31] Jicong Fan and Madeleine Udell. Online high rank matrix completion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8690–8698, 2019.

[32] Jicong Fan, Yuqian Zhang, and Madeleine Udell. Polynomial matrix completion for missing data imputation and transductive learning. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 3842–3849, 2020.

[33] Paul Fearnhead and Zhen Liu. On-line inference for multiple changepoint problems. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(4):589–605, 2007.

[34] Huijie Feng and Yang Ning. High-dimensional mixed graphical model with ordinal data: Parameter estimation and statistical inference. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 654–663, 2019.

[35] Vincent Fortuin, Dmitry Baranchuk, Gunnar Rätsch, and Stephan Mandt. Gp-vae: Deep probabilistic time series imputation. In *International Conference on Artificial Intelligence and Statistics*, pages 1651–1661. PMLR, 2020.

[36] Ravi Sastry Ganti, Laura Balzano, and Rebecca Willett. Matrix completion under monotonic single index models. In *Advances in Neural Information Processing Systems*, pages 1873–1881, 2015.

[37] Andrew Goldberg, Ben Recht, Junming Xu, Robert Nowak, and Jerry Zhu. Transduction with matrix completion: Three birds with one stone. In *Advances in neural information processing systems*, pages 757–765, 2010.

[38] Suriya Gunasekar, Pradeep Ravikumar, and Joydeep Ghosh. Exponential family matrix completion under structural constraints. In *International Conference on Machine Learning*, pages 1917–1925, 2014.

[39] Jian Guo, Elizaveta Levina, George Michailidis, and Ji Zhu. Graphical

models for ordinal data. *Journal of Computational and Graphical Statistics*, 24(1):183–204, 2015.

[40] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5 (4):1–19, 2015.

[41] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5 (4):19, 2016.

[42] T Hastie and R Mazumder. softimpute: Matrix completion via iterative soft-thresholded svd. *R package version*, 1, 2015.

[43] Peter Hoff. *sbgcop: Semiparametric Bayesian Gaussian Copula Estimation and Imputation*, 2018. URL `https://CRAN.R-project.org/package=sbgcop`. R package version 0.980.

[44] Peter Hoff and Maintainer Peter Hoff. Package 'sbgcop'. 2018.

[45] Peter D Hoff et al. Extending the rank likelihood for semiparametric copula estimation. *The Annals of Applied Statistics*, 1(1):265–283, 2007.

[46] Florian M Hollenbach, Iavor Bojinov, Shahryar Minhas, Nils W Metternich, Michael D Ward, and Alexander Volfovsky. Multiple imputation using gaussian copulas. *Sociological Methods & Research*, 50(3):1259–1283, 2021.

[47] Daniel Hsu, Sham Kakade, Tong Zhang, et al. A tail inequality for quadratic forms of subgaussian random vectors. *Electronic Communications in Probability*, 17, 2012.

[48] Alexander Ilin and Tapani Raiko. Practical approaches to principal component analysis in the presence of missing values. *Journal of Machine Learning Research*, 11(Jul):1957–2000, 2010.

[49] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[50] Adel Javanmard, Andrea Montanari, et al. Online rules for control of false discovery rate and false discovery exceedance. *The Annals of statistics*, 46 (2):526–554, 2018.

[51] Julie Josse, Jérôme Pagès, and François Husson. Multiple imputation in principal component analysis. *Advances in data analysis and classification*, 5 (3):231–246, 2011.

[52] Julie Josse, François Husson, et al. missmda: a package for handling missing values in multivariate data analysis. *Journal of Statistical Software*, 70 (1):1–31, 2016.

[53] Raghunandan H Keshavan, Andrea Montanari, and Sewoong Oh. Matrix completion from noisy entries. *Journal of Machine Learning Research*, 11 (Jul):2057–2078, 2010.

[54] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[55] Michael R Kosorok. *Introduction to empirical processes and semiparametric inference.* Springer, 2008.

[56] Andrew S Lan, Christoph Studer, and Richard G Baraniuk. Matrix recovery from quantized and corrupted measurements. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4973–4977. IEEE, 2014.

[57] Francesca Marta Lilja Di Lascio and Simone Giannerini. *CoImp: Copula Based Imputation Method*, 2019. URL `https://CRAN.R-project.org/package=CoImp`. R package version 1.0.

[58] Yew Jin Lim and Yee Whye Teh. Variational bayesian approach to movie rating prediction. In *Proceedings of KDD cup and workshop*, volume 7, pages 15–21. Citeseer, 2007.

[59] Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*, volume 793. Wiley, 2019.

[60] Han Liu, John Lafferty, and Larry Wasserman. The nonparanormal: Semiparametric estimation of high dimensional undirected graphs. *Journal of Machine Learning Research*, 10(10), 2009.

[61] Harry M Markowitz. Foundations of portfolio theory. *The journal of finance*, 46(2):469–477, 1991.

[62] Pierre-Alexandre Mattei and Jes Frellsen. Miwae: Deep generative modelling and imputation of incomplete data sets. In *International Conference on Machine Learning*, pages 4413–4423. PMLR, 2019.

[63] David S Matteson and Nicholas A James. A nonparametric approach for multiple change point analysis of multivariate data. *Journal of the American Statistical Association*, 109(505):334–345, 2014.

[64] Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *Journal of machine learning research*, 11(Aug):2287–2322, 2010.

[65] Robert McCulloch and Peter E Rossi. An exact likelihood analysis of the multinomial probit model. *Journal of Econometrics*, 64(1-2):207–240, 1994.

[66] Geoffrey McLachlan and Thriyambakam Krishnan. *The EM algorithm and extensions*, volume 382. John Wiley & Sons, 2007.

[67] Andriy Mnih and Russ R Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2008.

[68] Jared S Murray, David B Dunson, Lawrence Carin, and Joseph E Lucas. Bayesian gaussian copula factor models for mixed data. *Journal of the American Statistical Association*, 108(502):656–665, 2013.

[69] Bernard V North, David Curtis, and Pak C Sham. A note on the calculation of empirical p values from monte carlo procedures. *The American Journal of Human Genetics*, 71(2):439–441, 2002.

[70] Greg Ongie, Rebecca Willett, Robert D Nowak, and Laura Balzano. Algebraic variety models for high-rank matrix completion. In *International Conference on Machine Learning*, pages 2691–2700, 2017.

[71] Andrea Pagotto. *ocp: Bayesian Online Changepoint Detection*, 2019. URL `https://CRAN.R-project.org/package=ocp`. R package version 0.1.1.

[72] Ari Pakman and Liam Paninski. Exact hamiltonian monte carlo for truncated multivariate gaussians. *Journal of Computational and Graphical Statistics*, 23(2):518–542, 2014.

[73] Weiliang Qiu and Harry Joe. clustergeneration: random cluster generation (with specified degree of separation). *R package version*, 1(7):75275–0122, 2009.

[74] Aaditya Ramdas, Fanny Yang, Martin J Wainwright, and Michael I Jordan. Online control of the false discovery rate with decaying memory. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 5655–5664, 2017.

[75] Aaditya Ramdas, Tijana Zrnic, Martin Wainwright, and Michael Jordan. Saffron: an adaptive algorithm for online control of the false discovery rate. In *International conference on machine learning*, pages 4286–4294. PMLR, 2018.

[76] Benjamin Recht, Maryam Fazel, and Pablo A Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM review*, 52(3):471–501, 2010.

[77] Jason DM Rennie and Nathan Srebro. Loss functions for preference levels: Regression with discrete ordered labels. In *Proceedings of the IJCAI multidisciplinary workshop on advances in preference handling*, pages 180–186. Kluwer Norwell, MA, 2005.

[78] Jasson DM Rennie and Nathan Srebro. Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd international conference on Machine learning*, pages 713–719, 2005.

[79] David S Robertson, Jan Wildenhain, Adel Javanmard, and Natasha A Karp. onlinefdr: an r package to control the false discovery rate for growing data repositories. *Bioinformatics*, 35(20):4196–4199, 2019.

[80] Geneviève Robin, Olga Klopp, Julie Josse, Éric Moulines, and Robert Tibshirani. Main effects and interactions in mixed and incomplete data frames. *Journal of the American Statistical Association*, 115(531):1292–1303, 2020.

[81] Donald B Rubin. Multiple imputation after 18+ years. *Journal of the American statistical Association*, 91(434):473–489, 1996.

[82] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th international conference on Machine learning*, pages 880–887, 2008.

[83] Daniel J Stekhoven. Using the missforest package. *R package*, pages 1–11, 2011.

[84] Daniel J Stekhoven and Peter Bühlmann. Missforest—non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118, 2011.

[85] Daniel J Stekhoven and Peter Bühlmann. Missforest—non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118, 2012.

[86] Michael E Tipping and Christopher M Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

[87] Hideatsu Tsukahara. Semiparametric estimation in copula models. *Canadian Journal of Statistics*, 33(3):357–375, 2005.

[88] Douglas Turnbull, Luke Barrington, David Torres, and Gert Lanckriet. Towards musical query-by-semantic-description using the cal500 data set. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 439–446, 2007.

[89] Madeleine Udell and Alex Townsend. Why are big data matrices approximately low rank? *SIAM Journal on Mathematics of Data Science*, 1(1):144–160, 2019.

[90] Madeleine Udell and Alex Townsend. Why are big data matrices approximately low rank? *SIAM Journal on Mathematics of Data Science (SIMODS)*, 1(1):144–160, 2019. URL `https://epubs.siam.org/doi/pdf/10.1137/18M1183480`.

[91] Madeleine Udell, Corinne Horn, Reza Zadeh, Stephen Boyd, et al. Generalized low rank models. *Foundations and Trends® in Machine Learning*, 9 (1):1–118, 2016.

[92] Aad W Vaart and Jon A Wellner. *Weak convergence and empirical processes: with applications to statistics*. Springer, 1996.

[93] Stef Van Buuren and Karin Oudshoorn. *Flexible multivariate imputation by MICE*. Leiden: TNO, 1999.

[94] Gerrit JJ van den Burg and Christopher KI Williams. An evaluation of change point detection algorithms. *arXiv preprint arXiv:2003.06222*, 2020.

[95] Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices. *arXiv preprint arXiv:1011.3027*, 2010.

[96] Shuo-Yang Wang, Ju-Chiang Wang, Yi-Hsuan Yang, and Hsin-Min Wang. Towards time-varying music auto-tagging based on cal500 expansion. In *2014 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2014.

[97] Bing Xing, Ning Yang, and Xu Yaosheng. Adaptive estimation of multivariate regression with hidden variables. *The annals of statistics*, 2021.

[98] Chengrun Yang, Yuji Akimoto, Dae Won Kim, and Madeleine Udell. Oboe: Collaborative filtering for automl model selection. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1173–1183, 2019.

[99] Grace Yoon, Raymond J Carroll, and Irina Gaynanova. Sparse semiparametric canonical correlation analysis for data of mixed types. *Biometrika*, 107(3):609–625, 2020.

[100] Jinsung Yoon, James Jordon, and Mihaela Schaar. Gain: Missing data imputation using generative adversarial nets. In *International Conference on Machine Learning*, pages 5689–5698. PMLR, 2018.

[101] Kai Yu, Shenghuo Zhu, John Lafferty, and Yihong Gong. Fast nonparametric matrix factorization for large-scale collaborative filtering. In *Proceedings of the 32nd ACM SIGIR International Conference on Research and Development in Information Retrieval*, pages 211–218, 2009.

[102] Yuxuan Zhao and Madeleine Udell. Matrix completion with quantified uncertainty through low rank gaussian copula. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[103] Yuxuan Zhao and Madeleine Udell. Missing value imputation for mixed data via gaussian copula. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 636–646, 2020.

[104] Yuxuan Zhao and Madeleine Udell. gcimpute: A package for missing data imputation. *arXiv preprint arXiv:2203:05089*, 2022.

[105] Yuxuan Zhao, Eric Landgrebe, Eliot Shekhtman, and Madeleine Udell. Online missing value imputation and change point detection with the gaussian copula. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022.