Imputation of Missing Data Using Gaussians Mixtures

Darrell Aucoin (1003562316)
Department of Computer Science
University of Toronto
27 King's College Cir, Toronto, ON M5S
daucoin@cs.toronto.edu

Abstract

Data imputation is an imperative task in modern data analysis, and while data imputation through MCMC is one of the more accurate techniques it still requires a model to impute the missing data. Unfortunately, a good model is not always easy to discern, in this case we look at the use of gaussian mixtures as a general case model for MCMC to impute missing continuous data. This gaussian mixture model (with enough mixtures) creates a good approximation of the dataset distribution, allowing relatively accurate imputations that is robust against many continuous dataset distributions. Furthermore, since MCMC returns multiple imputations this can be used to gauge the uncertainty of imputations and it's affects on the downstream data analysis.

1 Introduction

Missing data is prevalent in most datasets and one of the more troublesome aspects of data analysis. Missing data can refer to missing entries, or missing records. Missing entries can refer to missing column values, data entered incorrectly, or truncated values. Handling missing data can be a difficult but necessary task as most data mining algorithms require complete data records. Simple solutions like list-wise deletion of records with missing values reduces the power of a study and can introduce problems like the Simpson's paradox where a subset of the data has a trend but the data as a whole does not. While mean substitution might be viable in some circumstances, a good imputation procedure should take into account the multivariate joint distribution of variables and account for both sampling and imputation variation. A poor imputation procedure can distort data giving misleading analysis or create unjustified certainties.

Missing data are usually classified into 3 types: (i) missing completely at random (MCAR), (ii) missing at random (MAR), and (iii) missing not at random (MNAR). MCAR is when the probability that a variable, X_i, is missing does not depend on the rest of the known variable(s). MAR is when the probability that X_i is missing is dependent on the other known variables but not X_i itself. MNAR is when probability of X_i being missing is dependent on the value of X_i itself. Most research is in the areas of MCAR and MAR with MNAR largely considered unrecoverable [Gheyas and Smith, 2010].

$$\begin{aligned}
\text{MCAR} \quad & P\left[X_i = \text{missing} \mid X_{-i} = x_{-i}, X_i = x_i\right] = P\left[X_i = \text{missing}\right] \\
\text{MAR} \quad & P\left[X_i = \text{missing} \mid X_{-i} = x_{-i}, X_i = x_i\right] = P\left[X_i = \text{missing} \mid X_{-i} = x_{-i}\right] \\
\text{MNAR} \quad & P\left[X_i = \text{missing} \mid X_i = x_i\right] \neq P\left[X_i = \text{missing}\right]
\end{aligned}$$

1.1 Project's Goal

This project's goal is to impute missing data entries (MCAR) in database tables automatically using MCMC and a mixture of gaussians model. MCMC imputation techniques requires a data model that

the data comes to do imputation. However, knowledge of the type of distribution the data comes from is not always known. In such cases, we propose a MCMC framework to generalize imputation: a gaussian mixture model. If this MCMC implementation works, the imputed values should look like they came from the conditional distribution of the joint distribution that generated the dataset.

Similar to how various lines can approximate a curve, given enough mixtures, a gaussian mixture model can approximate the distribution of various continuous datasets. If we encode in this model to also impute the missing entries, the MCMC algorithm will generate multiple imputation values. Although it might be tempting to impute the missing value as the mean of the candidate values, the mean will have less variability associated with it, giving less variability to the downstream data analysis. To keep the downstream data analysis as viable as possible it is recommended to perform multiple versions of the downstream data analysis using the various possible imputed datasets and estimating the quality of interest θ [Josse et al., 2011].

1.2 Stan

For implementing the MCMC algorithm, we used the probabilistic programming language stan. Stan has two powerful features: Hamiltonian Monte Carlo (HMC) and variational inference. HMC uses the gradient of the log probability function to hasten both convergence and parameter exploration. Missing data entries can be thought of as parameters in this model. An unknown quantity vectors θ can be thought of as the position of a particle in a vector field determined by the gradient. At each iteration, a random momentum is created, simulating the path of the particle with potential energy set by the negative log probability function. The leapfrog algorithm is used to approximate the continuous changes over time into discrete chunks enabling easier simulation. Simulation error, if it occurs, is then corrected by a Metropolis rejection step [Stan Development Team, 2016]. This allows stan to efficiently explore the parameter space as long as there is a non-zero probability path from the initial starting point to all points in the parameter space. This becomes an important point of consideration when the parameter space is isolated into pockets surrounded by approximately zero probability paths to other pockets. Once initialized in one of these pockets, the "particle" is unlikely to ever leave it (instances of this can be seen in Figure 5).

Stan also speeds up convergence by employing variational inference to approximate the posterior $p(\theta \mid y)$ with a parameterized distribution $q(\theta \mid \phi)$. This is done minimizing the Kullback-Leibler divergence,

$$\phi^* = \arg\max_{\phi} KL\left[q(\theta \mid \phi) \parallel p(\theta \mid y)\right]$$

Stan implements this in the programming background of it's computations using stan's Automatic Differentiation Variational Inference (ADVI) algorithm, leveraging stan's libarary of transformations and automatic differentiation techniques. This works with any stan model and requires no specifications by the user [Stan Development Team, 2016].

2 Review of Existing Data Imputation Techniques

There are various techniques for data imputation. These techniques can have multiple imputations or a single imputation for a missing entry. MCMC data imputation is an example of a model-based imputation technique giving multiple imputations. A model of how the data is generated is specified, and the MCMC algorithm returns multiple imputations based on it's iterations. Multiple imputation techniques communicate the variability of the imputation and allows incorporation of that variability in the downstream analysis but usually has a greater computational cost than single imputation [Gheyas and Smith, 2010]. Model-based techniques are only as good as the model they use. However, if the model is viable and the MCMC algorithm converges, then the imputations are usually very accurate. In a comparison by Gheyas and Smith [2010] between 25 imputation algorithms (and their proposed algorithm: GEMI) MCMC consistently ranked 2nd (behind GEMI) until about 40% of the dataset was missing [Gheyas and Smith, 2010].

Other imputation techniques include mean substitution (MS), hot deck imputation (HD), expectation maximization (EM), as well as techniques based on principal component analysis (PCA) and generalized regression neural networks (GRNN). Gheyas and Smith [2010] proposed algorithm is based on generalized regression neural networks. Mean substitution replaces the missing values with the

means of those variables. This is only viable for MCAR data and even then doesn't take into account the joint multivariate distribution of the data or the variance of the imputation.

Hot deck imputation replaces the missing value with an observed value from a "similar" complete record. This "similarity" is based on k-nearest neighbors algorithm (KNN) and imputes the missing data from the "nearest neighbors" of complete records. This is either done by random selection or the "nearest neighbor" based on a prescribed metric. Hot deck is model independent and thus resistant to model misspecification but is highly reliant on the number of complete records for imputation accuracy. However, hot deck is not free from assumptions as the choice of metric matching donors to recipients and the variables included in this metric creates implicit assumptions [Andridge and Little, 2010].

PCA for data imputation typically gives a single imputation by approximating a low ranked matrix minimizing the reconstruction error (usually though least-squares). This is similar to collaborative filtering used with many recommendation systems. Josse et al. [2011] have also produced work on using multiple imputations using PCA.

Gheyas and Smith [2010] proposed a novel imputation algorithm based on generalized regression neural networks (GRNN). GRNN is a special case of a radial basis network (RBN), using 2 hidden layers: a pattern (radial basis layer) and a summation layer (linear layer), as shown in Figure 1. The relationship between each record ($\mathbf{x}^{(i)}$) and the response ($y^{(i)}$) is memorized in a unit of the pattern layer, leaving only 1 tuning parameter, σ^2, for the GRNN. This means that each node in the pattern layer must store a record ($\mathbf{x}^{(i)}$) and it's corresponding $y^{(i)}$ value in one of it's summation layers. The summation layer has 2 nodes: numerator and denominator. GRNN operates similar to KNN in that it is a local approximation algorithm. In the case of GRNN, it will look at all comparisons between $\mathbf{x}$ and $\mathbf{x}^{(i)}$ and weight the response ($y^{(i)}$) inversely proportional to the euclidean distance between $\mathbf{x}$ and $\mathbf{x}^{(i)}$. There are some important considerations when comparing GRNN's with regular feed-forward neural networks. GRNNs have only 1 parameter to tune, σ^2, and it's optimization function is convex making GRNN's comparatively easy to train. However, due number of nodes in the pattern layer and that each node in the pattern layer requires a record to be stored in it, GRNNs tend to have high memory and computational cost for finding $\hat{y}$ for each new $\mathbf{x}$ [Gheyas and Smith, 2010].

$$\text{Pattern Layer Unit} \quad w_i(x) = \exp\left\{-\frac{1}{2}\frac{\left(\mathbf{x}-\mathbf{x}^{(i)}\right)^\top\left(\mathbf{x}-\mathbf{x}^{(i)}\right)}{\sigma^2}\right\}$$

$$\text{Output Layer} \quad f(x) = \frac{\sum_{i=1}^{N} y^{(i)} \times w_i(\mathbf{x})}{\sum_{i=1}^{N} w_i(\mathbf{x})}$$

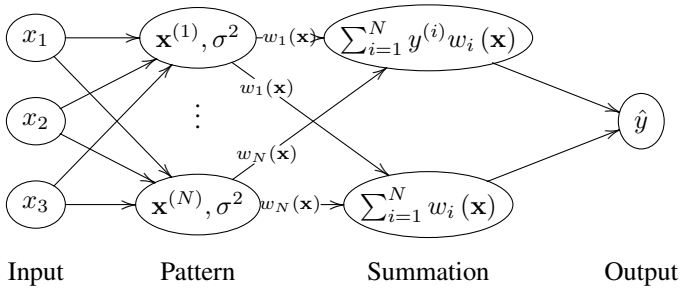


Figure 1: Generalized Regression Neural Networks (GRNN)

Gheyas and Smith [2010] proposed algorithm is for each variable creating an ensemble of GRNN's where each base GRNN is trained on a different subset of the dataset. The $\hat{Y}_k$'s of these GRNN's are then combined to form $\hat{Y}$, the imputed data entry. Gheyas and Smith [2010] also cited Valdovinos and Sánchez [2009] work showing that when ensemble members have non-uniform performance the efficiency of simple majority voting is adversely affected. To combat this Gheyas and Smith [2010], recommended a GRNN combiner to output $\hat{Y}$, giving the proposed model in Figure 2.

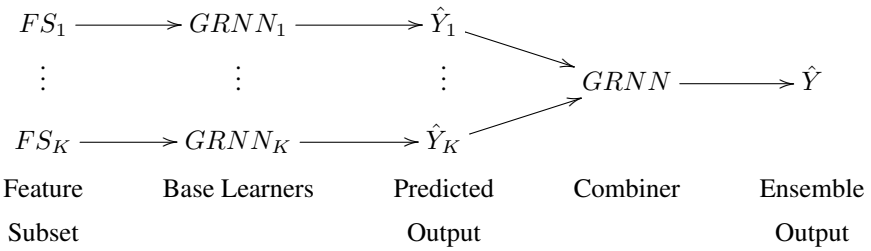


Figure 2: Gheyas and Smith [2010] proposed GRNN-based algorithm

3 Experiments

We will be studying the use of MCMC, via stan, and gaussian mixtures to impute missing data on simulated dataset, including datasets that are not a gaussian mixture. For MCMC imputations, a model of the data is needed, but this is not always easily inferred. In the case of continuous missing data we are proposing gaussian mixtures to approximate the data distribution and impute the missing data entries from. For our experiments we will be using MCAR continuous missing data entries but this can be later augmented to work with MAR data as well. Following the suggestions of stan documentation we formed Σ_k from $\sigma_{k,i}$ and Ω_k using priors as described below.

$$\begin{bmatrix} x \\ y \end{bmatrix} \sim \theta_1 MVN\left(\mu_1, \Sigma_1\right) + \theta_2 MVN\left(\mu_2, \Sigma_2\right) + \cdots + \theta_K MVN\left(\mu_K, \Sigma_K\right)$$

$$\mu_k = \begin{bmatrix} \mu_{k,1} \\ \mu_{k,2} \end{bmatrix}$$

$$\Sigma_k = \begin{bmatrix} \sigma_{k,1} & \\ & \sigma_{k,2} \end{bmatrix} \Omega_k \begin{bmatrix} \sigma_{k,1} & \\ & \sigma_{k,2} \end{bmatrix}$$

$$\sum_{k=1}^{K} \theta_k = 1$$

$$\mu_{k,i} \sim N\left(0, 5\right)$$

$$\sigma_{k,i} \sim \text{half-log Normal}\left(0, 1\right)$$

$$\theta_k \sim \text{dirichlet}\left(0.3, 0.3, 0.3\right)$$

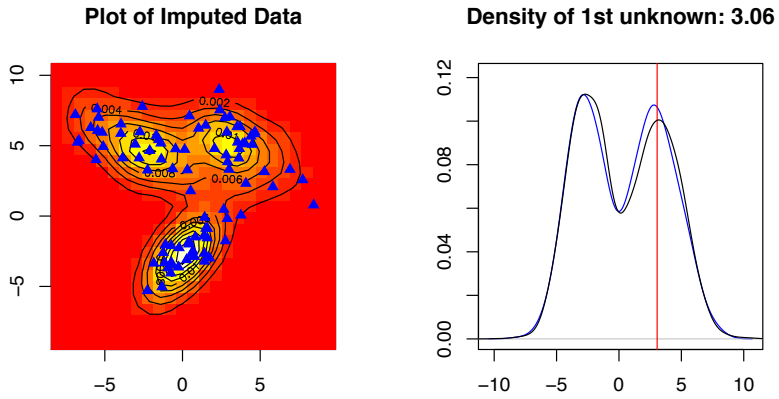
$$\Omega_k \sim \text{LKJcorr}\left(1\right)$$

The experiments used a simulated gaussian mixture dataset (with correlated variables), a simulated non-gaussian dataset, and on the continuous portion of the iris dataset. 10% of the data points for are randomly corrupted, erasing a random data entry in the row, or more specifically changed to a number that should not be produced (999999999). This is because stan will only accept numbers as input. A maximum of one entry is corrupted for each row. If the MCMC algorithm correctly converges, then the densities for the returned values for imputation should be very similar to the conditional distribution for that variable, given the records known values.

3.1 Correlated Gaussian Mixture

The first experiment was a simulated data that came from a mixture of 2-dimensional gaussians with correlated variables. The simulated dataset uses an equal proportion of the multivariate normals below with 900 data points (90 data entries are corrupted). The MCMC algorithm was told to assume a mixture of 3 gaussians.

$$MVN\left(\begin{bmatrix}0\\-3\end{bmatrix},\begin{bmatrix}2 & 2\\2 & 3\end{bmatrix}\right), MVN\left(\begin{bmatrix}-3\\5\end{bmatrix},\begin{bmatrix}5 & -3\\-3 & 1/2\end{bmatrix}\right), MVN\left(\begin{bmatrix}3\\5\end{bmatrix},\begin{bmatrix}5 & -2\\-2 & 2.5\end{bmatrix}\right)$$



The imputed missing entries are are plotted as ▲ in the left plot (along with it's corresponding known pair).
The right plot shows the distribution approximation of the imputed data entry (the black line) and should correspond to the conditional distribution given it's known pair (the blue line). The true value is shown in red.
To approximate the true conditional distribution we used a high density simulation, filtered out the points not corresponding to the missing values paired value $x \pm 0.01$, and plotted it's corresponding density.

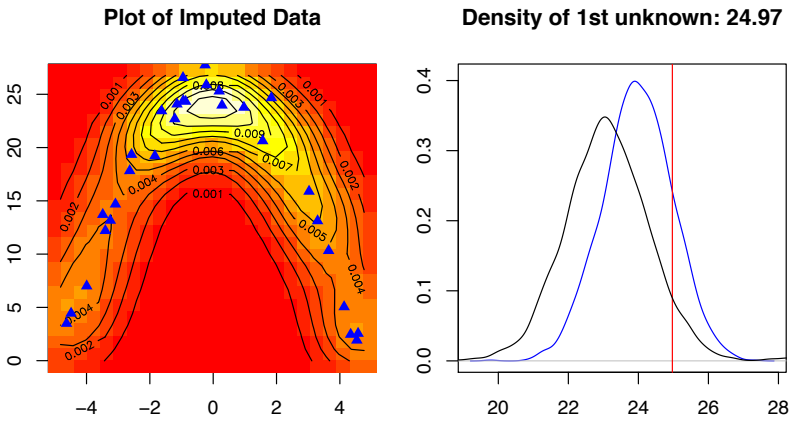
Figure 3: Correlated Gaussian Mixture

The plot of the distribution of imputed values in the left plot of Figure 3 is representative of all 90 imputed values. The distribution of imputed values follows the approximation of the conditional distribution of true distribution given it's corresponding known values. The true value, shown in red, looks like it could of been sampled from the imputed values.

To showcase the overall effectiveness of this imputation the missing entries are plotted against the true distribution in the left plot of Figure 3. The missing entries are randomly selected from the imputed values that the MCMC algorithm returns. This incorporates the variability of the imputation in the plot and allows a more accurate imputation when it's density is not unimodal. Overall the plotted points look like they could of been sampled from the original distribution, indicating the imputation was done well.

3.2 Mixture of Gaussians for Approximating an Arc

To test the flexibility of gaussian mixtures to approximate various non-gaussian continuous data distribution shapes, we simulated an arc that can be seen in Figure 4 using 300 data points (30 data entries are corrupted). The MCMC algorithm was told to assume a mixture of 5 gaussians.



$$x \sim Unif\,(-5,5) \quad y = 25 - x^2 + \epsilon \quad \epsilon \sim N\,(0,1)$$

Figure 4: Imputation on Arc: 5 Gaussian Mixtures

Similar to the first experiment, the distribution of the imputed values is close to the conditional distribution of the true distribution (as can be seen in Figures 4 and 5). The true value also looks like it could of been sampled from this distribution. However, we can also see evidence that the gaussian mixture cannot fully fit the given dataset with (just 5 gaussian mixtures) as some of the imputed distributions are shifted slightly from where they should be. This is likely due to having only 5 gaussian mixtures to approximate the arc. Much like approximating a curve with a set of straight lines, a gaussian mixture can only do so much to approximate the dataset. Approximation accuracy can be increased with increasing the number of gaussian mixtures but at the expense of increased computational cost.

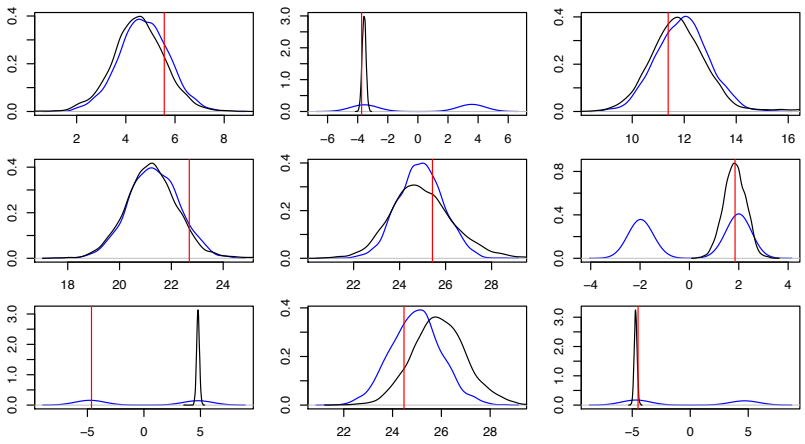


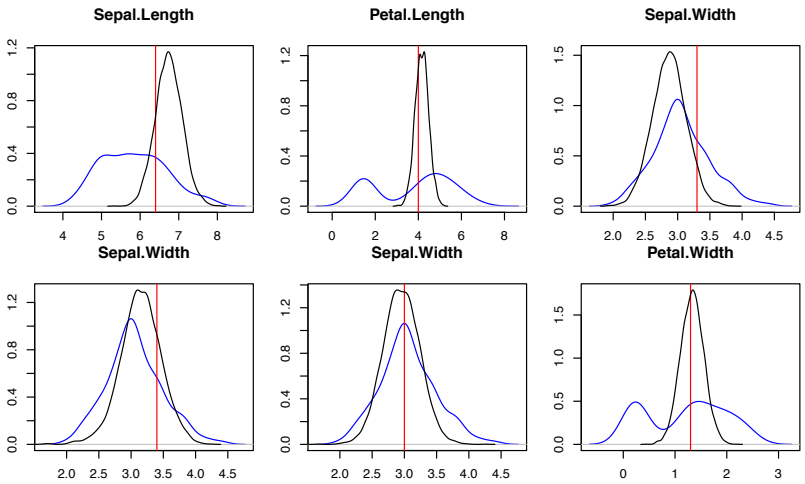
Figure 5: Imputation on Arc: Various Imputed Distributions

Another important consideration is that with HMC stan searches the parameter space along where it initially start searching. This means that if stan starts off imputing missing values in a pocket of parameter values with little probability of getting to the other pockets of parameter values only the first pocket will likely be explored. This can be seen in some of the plots in Figure 5, the imputed values are only explored for one portion of the true distribution for 4 of the plots. These are the data points where y is known with a low value (and x is imputed) as shown in the plots in Figure 5. One solution for this is to dramatically increase the number of iterations to increase the probability of exploring the other pockets, much like buying billions of lottery tickets to win the lottery. Another,

more practical solution is to increase the number of chains for the MCMC algorithm. Each chain will have a different initial starting point increasing the probability that more pockets will be explored.

3.3 Mixture of Gaussians for Approximating Iris Dataset

To test this model on a real dataset we chose the continuous portion iris dataset, as this dataset is well known and large enough to get good missing data results from. Again, we corrupted 10% of the dataset (15 data entries are corrupted). The MCMC algorithm was told to assume a mixture of 3 gaussians.



The imputed distribution estimation is in black, and the true value is in red. For comparison, the blue line is the full distribution of the variable specified.

Figure 6: Imputation on Iris dataset: 3 Correlated Gaussian Mixture

While it is no longer possible to get the conditional distribution (since we don't know the true distribution), we can compare the imputed distribution to an approximate of the unconditional distribution of the variable itself. The imputed distribution should have less variance than the unconditional distribution and the true value should easily be sampled from the imputed distribution. All features that we can see in Figure 6.

4 Limitations of Approach

There are a variety of limitations of imputing missing data using gaussian mixtures. One of the biggest is the necessity of working with only continuous data. This is a major limitation as many datasets have categorical data. Even if imputation isn't needed for the categorical data, this model cannot take advantage of the information that these categorical variables provides. Categorical data can be imputed, just not with this approach.

This approach is also highly reliant on the user to specify the number of gaussian mixtures, meaning the user must search though trial and error to find what is the best mixture of gaussians. Unfortunately, MCMC computational cost increases with respect to the size of the dataset and the model complexity. This means that a user might spend a lot of time searching for the appropriate number of gaussians to use. This is especially problematic for complex non-linear distributions as they are more difficult for gaussian mixtures to approximate, meaning a large number of mixtures is needed.

At present this approach only works with MCAR data, however with some changes to the model (increasing complexity) it should be able to impute MAR data as well. This will be explored in future improvements.

5 Future Improvements

The model can be modified to work with MAR data by first modeling the probability of having a variable given the known variables. This information can then be used to inform the search for imputing missing values.

One modification to take advantage of complete categorical data is to model separate gaussian mixtures for each category. If categories are related in some way, this could be done though a hierarchical model and sharing some parameters between the categories.

Stan has methods to estimate parameters (such as μ, σ^2) while using truncated data, we should be to incorporate this into our model and allow for missing data imputation. However, due to lack of knowledge of what the true distribution is beyond these truncated values any modeling assumptions needs to more highly scrutinized.

As well as the above, our model can still be optimized to allow for faster computational time.

6 Conclusion

Data imputation is imperative issue in modern data analysis. Data is rarely clean or complete, and list-wise deletion is rarely a viable action for a good data analysis. In this paper, we explore the viability of using MCMC with a gaussian mixture model to impute continuous MCAR data. While not perfect, this model gave good results for imputation that is robust to various different continuous data distributions. In conclusion, MCMC with a gaussian mixture model should be considered for imputing continuous MCAR data when a more appropriate model cannot be determined.

References

Rebecca R. Andridge and Roderick J. A. Little. A review of hot deck imputation for survey non-response. *International Statistical Review*, 78(1):40–64, 2010. ISSN 1751-5823. doi: 10.1111/j.1751-5823.2010.00103.x. URL http://dx.doi.org/10.1111/j.1751-5823.2010.00103.x.

Iffat A. Gheyas and Leslie S. Smith. A neural network-based framework for the reconstruction of incomplete data sets. *Neurocomputing*, 73(16-18):3039–3065, 2010. doi: 10.1016/j.neucom.2010.06.021. URL http://dx.doi.org/10.1016/j.neucom.2010.06.021.

Julie Josse, Jérôme Pagès, and François Husson. Multiple imputation in principal component analysis. *Advances in Data Analysis and Classification*, 5(3):231–246, 2011. ISSN 1862-5355. doi: 10.1007/s11634-011-0086-7. URL http://dx.doi.org/10.1007/s11634-011-0086-7.

Stan Development Team. *Stan Modeling Language Users Guide and Reference Manual, Version 2.14.0*, pages 22–28. 2016. URL http://mc-stan.org.

R. M. Valdovinos and J. S. Sánchez. *Combining Multiple Classifiers with Dynamic Weighted Voting*, pages 510–516. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-642-02319-4. doi: 10.1007/978-3-642-02319-4_61. URL http://dx.doi.org/10.1007/978-3-642-02319-4_61.