Artificial Neural Networks to Impute Rounded Zeros in Compositional Data

Matthias Templ

Abstract Methods of deep learning have become increasingly popular in recent years, but they have not arrived in compositional data analysis. Imputation methods for compositional data are typically applied on additive, centered or isometric log-ratio representations of the data. Generally, methods for compositional data analysis can only be applied to observed positive entries in a data matrix. Therefore one tries to impute missing values or measurements that were below a detection limit. In this paper, a new method for imputing rounded zeros based on artificial neural networks is shown and compared with conventional methods. We are also interested in the question whether for ANNs, a representation of the data in log-ratios for imputation purposes, is relevant. It can be shown, that ANNs are competitive or even performing better when imputing rounded zeros of data sets with moderate size. They deliver better results when data sets are big. Also, we can see that log-ratio transformations within the artificial neural network imputation procedure nevertheless help to improve the results. This proves that the theory of compositional data analysis and the fulfillment of all properties of compositional data analysis is still very important in the age of deep learning.

1 Introduction

Compositional data and rounded zeros: Compositional data are quantitative descriptions of the parts of some whole, whereby the important information is not the absolute values but relative information. Measurements involving proportions, percentages, probabilities, concentrations are compositional data. An example of compositional data is the 24 hours day categorized into sleeping time, working time, lunch break, and free time. There exists inner dependencies: if one sleeps longer at

Matthias Templ
Institute of Data Analysis and Process Design, Zurich University of Applied Sciences, Rosenstrasse 3, CH-8401 Winterthur, Switzerland e-mail:

a day, automatically less time for other things is left. Also not all values are possible (e.g. a day is restricted to 24h) and the compositions are represented in the simplex instead of an Euclidean geometry. As for the previous example, the sleeping time plus working time cannot be larger than 24 hours, for example. Statistical methods are mostly designed to be applied in the Euclidean geometry. A log-ratio presentation of the data is a way out to shift the original measurements from the simplex to the Euclidean geometry. However, zeros or rounded zeros are problematic when computing log-ratios. Either in the denominator, the ratio is not defined or in the numerator the log is not defined. Zeros are true values where the measurement reports zero. A rounded zero occurs whenever a measurement unit cannot measure a too small amount in a sample. For example, a measurement unit can only detect ppm's of a certain chemical element higher than a certain threshold - the detection limit - and reports zeros whenever there are fewer ppm's in the sample. The value is thus rounded to zero, but only because the measurement unit could not detect anything even though we know that there must be a certain contribution of this chemical element in the sample. Values below the detection limit are thus known under the name rounded zeros [Aitchison, 1986, Martín-Fernández et al., 2003] and represent left censored data.

It is fully meaningful to impute rounded zeros with a reasonable small value and continue with the processing of a complete data set [Filzmoser et al., 2018]. To guarantee imputation by a considerable small value, an upper bound should be considered - typically the detection limit of this variable. Any imputation method should then impute with values below this upper bound, but larger than zero.

To give an example as outlined also similarly in Filzmoser et al. [2018], a rounded zero value present for a variable, say *Arsenic* and the detection limit for *Aresenic* is 0.1 mg/kg. An imputed value should be thus in the interval 0 0 1 . The imputation should consider the information about all other compositional parts, i.e. a multivariate approach is preferable. Furthermore, the log-ratios between other parts should remain unchanged after the imputation of the rounded zeros.

The imputation of rounded zeros can be considered as a very difficult problem with different solutions proposed in the literature [Martín-Fernández et al., 2003, Palarea-Albaladejo et al., 2007, Palarea-Albaladejo and Martín-Fernández, 2008, Martín-Fernández et al., 2012, Palarea-Albaladejo and Martín-Fernández, 2013, Palarea-Albaladejo et al., 2014, Martín-Fernández et al., 2015, van-den Boogaart et al., 2015, Templ et al., 2016, Chen et al., 2017]. More on this in Section 2.

Artificial deep neural networks: A neural network is just a non-linear statistical model [Hastie et al., 2009], which is based on weighted linear combinations of the sample values. Within an ANN, the aim is to find millions of weights to get the best possible output from input data and multiple layers. The weights are updated iteratively. In the first iteration, the weights are random and the result is accordingly bad. For each iteration, we go in the "optimum direction", whereby a gradient method is used for this. It requires backpropagation with an optimizer (e.g. Adam) and a loss function (e.g. mean squared error). The quality of the predictions is evaluated based on a selected metric (e.g. mean absolute error) on validation and training data.

Often missing values are just initialized before the neural network is fitted [Smieja et al., 2018]. Also, missing values may be incorperated in a classification problem without imputing them. The trick is to add an indicator matrix determining the position of missing values to the data matrix and then using a generative additive network (GAN). For more details, see Li et al. [2019], Yoon et al. [2018], Mattei and Frellsen [2018]. Note that this is applied within several image inpainting methods [see, e.g., Xie et al., 2012]. However, our aim is different. Rounded zeros are imputed with fixed values so that the imputed data set can be used by any researcher or practitioner for any analysis that needs complete data.

ANNs for missing values: Artificial neural networks have been rarely used for the imputation of missing values purposes. Maiti et al. [2008] used neural networks for an agriculture survey (and compared it with hot-deck imputation). Various other contributions took place around 2007/2008. Afterwards, Jerez et al. [2010] used a multi-layer perceptron method for the imputation of breast cancer data. McCoy et al. [2018] used variational autoencoders for imputation. Choudhury and Pal [2019] and Silva-Ramírez et al. [2015] used artificial neural networks. Another approach is to not explicitly impute missing values, but to add the missingness matrix. Hereby, a position of the original data matrix with missing value is denoted with a 1, and non-missing values are denoted with 0. This matrix is then integrated into the whole model. Li et al. [2019], [Yoon et al., 2018] and [Mattei and Frellsen, 2018] used generative artificial networks for classification purposes under missing values. For special applications, these were also used by Lim [2019] and Arisdakessian et al. [2019] (RNA data). However, no ANN was used in the context of compositional data analysis and in the context of the imputation of rounded zeros in compositional data.

Outline: In Section, 2 methods for the imputation of rounded zeros are listed and Table 1 summarizes them. Section 3 introduces the specific type of artificial neural networks that is relevant in the context of imputation for compositional data. The parameters of our imputation procedure that have been made available in software are described in Section 4 for the imputation of rounded zeros in one variable. The focus of Section 5 is the adaptation to impute rounded zeros as an Expectation-Maximum (EM) method. This allows also to impute not only one variable at a time but sequentially all compositional parts with rounded zeros in a data set. Two different approaches are presented to replace rounded zeros in compositional data, one using a log-ratio transformation and the other without using a log-ratio representation. The outcome and comparison of the methods on real-world data sets are shown and described in Section 6. Section 7 concludes and summarizes the major findings.

2 General imputation methods and imputation methods for rounded zeros

Note that the simplest method is univariate and imputes zeros with 65% of the detection limit ( ). This minimizes the distortion of the covariance structure [Martín-Fernández et al., 2003, Martín-Fernández et al., 2011]. However, this kind of mean imputation leads to an underestimation of the compositional variability. The variability can be increased by drawing random numbers from a uniform distribution in 0 , with DL, the detection limit of a variable, but then the precision of the imputations lowers. Alternatively, Palarea-Albaladejo and Martín-Fernández [2013] and Palarea-Albaladejo et al. [2014] used a lognormal distribution, truncated by the threshold. In any case, all these methods represent univariate approaches to a multivariate compositional problem. Note that the methods from Palarea-Albaladejo and Martín-Fernández [2013] and Palarea-Albaladejo et al. [2014] needs at least one fully observed variable. In our application, we compare only multiple and multivariate methods and leave out univariate methods.

EM-based model-based methods offer multivariate approaches for the imputation of rounded zeros and by using censored regression it can be guaranteed that imputed values do not exceed a defined threshold. A modified EM algorithm [Palarea-Albaladejo et al., 2007, Palarea-Albaladejo and Martín-Fernández, 2008] in additive log-ratio coordinates [Aitchison, 1986] was reformulated by using pivot log-ratio coordinates [Filzmoser et al., 2018] in Martín-Fernández et al. [2012]. Both algorithms uses censored multiple regression for each variable in an EM-based manner until the imputations stabilize. The algorithm is complex because detection limits must also be represented in isometric log-ratio coordinates and inverse isometric log-ratio transformation must happen in each step of the algorithm together with an adjustment of values to not lose the absolute information of the original data. This is necessary when we still want to know, for example, the exact household expenditures in expenditure data and not only the ratios between them. A detailed description of this complex algorithm can be found in Martín-Fernández et al. [2012], Filzmoser et al. [2018]. The method itself needs also an initialization of rounded zeros (for example using the 65% method) because otherwise log-ratios cannot be calculated from zero values. Instead of ordinary least squares regression, a robust estimator can be plugged in.

For high-dimensional data, two other methods to replace rounded zeros should be mentioned. Templ et al. [2016] uses partial least squares regression for imputing all variables sequentially and repeatedly in an EM-based algorithm. To decrease computation time, Chen et al. [2017] uses a clustering approach to select variables within the algorithm.

In addition to methods suitable to handle rounded zeros by imputing below a given detection limit, we also want to compare popular imputation methods that do not take these limits into account. A major point of criticism of this contribution could be that methods for rounded zeros are compared with methods that do not take into account the special problem of rounded zeros. The motivation for this comparison

comes from practice and is not scientifically based, but is based on the experience as package maintainer of VIM [Templ et al., 2012, Kowarik and Templ, 2016], and robCompositions [Templ et al., 2011, Filzmoser et al., 2018] - both containing imputation methods. This experience says that readers need to see a comparison to understand that only (compositional) methods that take the problem of rounded zeros into account successfully impute rounded zeros. Up to our knowledge, articles on rounded zero imputation have only compared specific methods for rounded zeros, with the result that users without the obvious comparison still resort to unsuitable methods, because they are just trendy to use for more general imputation problems. Many users are also confused about whether they should use imputation methods for missing values in compositional data or methods for replacing rounded zeros. We show what happens when the wrong methods are selected and what the consequences are. Therefore, methods for rounded zeros of compositional data are also compared to very popular imputation methods: imputation using random forests implemented in R package [Stekhoven and Bühlmann, 2011] or the faster implementation in [Mayer, 2019], predictive mean matching using R package [van Buuren and Groothuis-Oudshoorn, 2011], -nearest neighbor imputation of R package [Kowarik and Templ, 2016], missing value imputation using pivot coordinates as implemented in R package in function [Hron et al., 2010] and from the same package a -nearest neighbor method for missing values using Aitchison distances in function [Hron et al., 2010].

3 Artificial neural networks

In this section the general construction of a neural network is explained, details about the neural network used for the imputation are given in the next section.

Neurons hold some information on each variable and a set of observations. Each neuron has an activation, depending on how the input information looks like. A layer in a deep neural network is a collection of neurons in one step. There are three types of layers: the input layer, the hidden layers, and the output layer. A neuron $_1$ in the input layer, for example, takes the observations of a data set as activations and the hidden layers represent non-linear weighted observations.

The activations **a** should take values between zero and one. The (default) activation function reLU (rectified Linear Unit) [He et al., 2015] takes the value zero if the activation is negative. Most activation functions use a non-linear function mapping between the inputs and the response.

In the first hidden layer with 2 1 with each neuron with 1 represents very detailed information, while passed through the layers, the information gets generalized and in the end summed up to either one or multiple neurons depending on the type of variable. represents the number of layers, including the input and the output layer, and the number of neurons within the layer . Figure 1 shows a simplified illustration of a network with two hidden layers and few neurons.

Table 1 Methods compared in this contribution. Univariate methods were not considered. We distinguish between methods for the imputation of missing values (not designed for detection limit problems) and rounded zeros (considering detection limites) and methods that do or do not take the compositional nature of compositional data into account.

Method	Reference	Description
Non-compositional approaches not designed for detection limit problems (non-CoDa, non-DL)		
deepImp	Templ [2020]	ANNs in an EM-based implementation (serves as a benchmark for deepImp-dl)
kNN	Kowarik and Templ [2016]	nearest neighbor imputation
mice	van Buuren and Groothuis-Oudshoorn [2011]	Predictive mean matching in an EM-based implementation
missForest (rf)	Stekhoven and Bühlmann [2011]	Imputation using random forests in an EM-based implementation
Non-compositional approaches considering detection limits (non-CoDa, DL)		
deepImp-dl	Templ [2020]	ANNs in an EM-based implementation considering detection limits (serves as a benchmark for deepImpCoDa-dl)
Compositional methods ignoring not designed for detection limit problems (CoDa, non-DL)		
deepImpCoDa	this article	ANNs in an EM-based implementation considering compositinal data (serves as a benchmark for deepImpCoDa-dl)
aknn	Hron et al. [2010]	NN imputatation for compositional data
impCoDa	Hron et al. [2010]	EM-based regression imputation using pivot coordinates.
Compositional methods considering detection limits (CoDa, DL)		
deepImpCoDa-dl	this article	ANNs in an EM-based implementation considering detection limits
imputeBDL-lm	Templ et al. [2016]	EM-based regression imputation using pivot coordinates and considering detection limits.
imputeBDL-pls	Templ et al. [2016]	EM-based partial least-squares regression imputation using pivot coordinates and considering detection limits.
imputeBDL-rob	Templ et al. [2016]	EM-based robust (MM) regression imputation using pivot coordinates and considering detection limits.
impRZilr	Martín-Fernández et al. [2012]	EM-based partial least squares regression imputation using pivot coordinates and considering detection limits.

The goal of training a network is to find the optimal weights $_1$ for each connection between the neurons of two layers (see also Figure 2). Then the activations of the neurons from the first layer are taken and the weighted sum according to these weights is computed.

Let $\mathbf{\Omega}$ be the matrix of weights in each layer. The weights of the -th layer are defined as $\boldsymbol{\omega}$. The activation values of the neurons in one layer are $\boldsymbol{\nu}$ $_1$ while the biases within one layer are $\boldsymbol{\beta}$ $_1$. The formula for the transition of one layer to the other is then defined by $\mathbf{\Omega}$ $\boldsymbol{\beta}$.

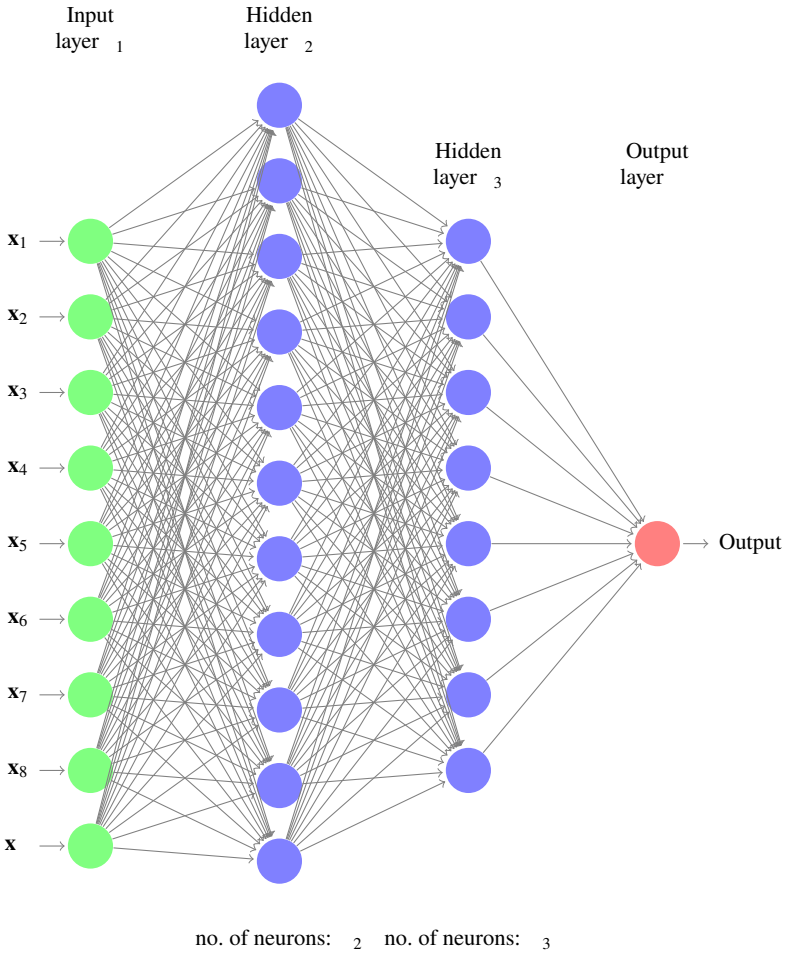

Fig. 1 Simplified schematic overview of a neural network with a few observations, two hidden layer with 13 and 9 neurons. Note that the default deep artificial neural network consists of observations, 10 layers, 1000 neurons in the first hidden layer, 900 in the second, ..., 100 in the last hidden layer and the output depends on the scaling of the target variable. In total 3325601 parameters (mostly weights) are trained in the default network.

After setting initially all weights randomly, a loss function of the network is defined. The output of the loss function is a single number judging the quality of the neural network. To lower the value of the loss function, an adaptive moment estimation called Adam [Kingma and Ba, 2014, Ruder, 2016] is used (other methods can be selected), which is a stochastic gradient descend method that uses adaptive learning rates for each parameter of the algorithm. With this gradient descend optimization all the weights $\boldsymbol{\Omega}$ are optimized to reach the next local minimum $\boldsymbol{\Omega}$ resulting in the most rapid decrease. This is causing the most rapid decrease in our loss func-

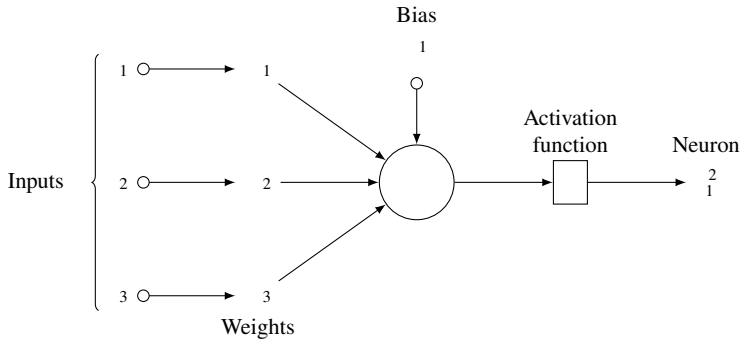

Fig. 2 This schematic illustration tells the story behind the calculation of one single neuron with respect to the first hidden layer. This neuron serves then as input for the next layer.

tion. The gradient used to adjust these weights is computed with back propagation, whereby the weights gets updated.

The weighted sum, which defines the activation of a neuron is defined by z ω ν ^{1} β . The activation of a neuron ν in layer , is then defined with ν z . The loss is therefore computed by comparison of z and ν . The sensitivity of the loss function with regard to changes in the weights ω is calculated with the chain rule

$$\frac{}{\omega} \quad \frac{z}{\omega}\frac{\nu}{z}\frac{}{\nu}$$

This means that the amount of this nudge influencing the last layer, depends on the strength of the previous neuron, $\frac{z}{\omega}$ ν ^{1}.

This is done over all training observations, $\frac{}{\omega}$ $\frac{1}{}$ $\frac{1}{0}\frac{}{\omega}$. The gradient vector contains all the partial derivatives from the loss function with respect to all weights and biases. The sensitivity of the loss function to changes in the bias is computed equivalent with

$$\frac{}{\beta} \quad \frac{\nu}{z}\frac{}{\nu}$$

The above described approach is then applied and computed through the complete neural network. The weights will be adjusted and the computation is iterated and repeated many times (also called the number of epochs, see below). More details about how neural networks are computed can be found, for example, in Nielsen [2015].

4 Artificial neural networks to impute rounded zeros - parameter settings

In the following an artificial neural network is described and adapted for imputation purposes, implemented in function *impNNetCoDa* in R package *deepImp*. This deep learning method works with a neural network based on *keras* [Allaire and Chollet, 2019, Chollet et al., 2015] and *tensorflow* [Abadi et al., 2018, 2015].

Before describing the whole EM-based algorithm that automatically imputes all rounded zeros in all variables of a data set, the parameter choices available in our implementation are outlined.

Initialization of rounded zeros: In principle, any other method from before can be used to initialize the rounded zeros and several choices are available in our implementation. The default approach is to use a nearest neighbor approach that uses Aitchison distances and robust means [Hron et al., 2010].

Iterations: Initialized missing values are updated step by step with imputations. Whenever all variables with rounded zeros are updated once, one iteration has passed. This represents an outer loop of the algorithm. If the EM-based sequential algorithm does not converge after a specified number of , then the algorithm is stopped and a warning is supplied.

Threshold for convergence: Given the -th iteration, a change in the imputed values compared to the previous iteration is measured. More precisely, the algorithm stops as soon as the sum of squared relative distances between the imputed values in the -th step and the 1-th step is below the specified threshold.

Number of epochs: The data set is not only sent once through the deep neural network but many times (number of epochs). This is needed to optimize the learning, which is an iterative process. Updating the weights in one pass (i.e. one epoch) is too little. Generally, it applies that too few epochs often leads to an underfit while too many epochs can lead to an overfit. The optimum number of epochs is data set dependent. In all our experiments, a choice of about 200-300 epochs gave the most promising results, when either looking at and comparing validation and training errors or when evaluating it directly on the validation criteria defined in Section 6.2. If the validation error is higher than the training error, this is an indication of overfitting. Figure 3 corresponds to the imputation of rounded zeros in the variable Ag in the Moss data set (see Section 6.1) using our artificial deep learning network with 10 layers and different number of epochs. No overfit is visible. Note that in our implementation 20% of the observations used as the validation set by default (for each epoch).

The number of epochs is one parameter of our algorithm with a sensible default. However, it is better to set the number of epochs to a high value instead of too small value, because in any case, the next parameter provides a stopping rule to stop the training of the deep neural network earlier.

For any given data set, the training and validation errors can be used to evaluate the correct number of epochs, but also to evaluate the number of layers and the number of neurons in the layers. If training and validation errors are quite different, too few layers, neurons, and/or epochs were selected. In Figure 3 we see a perfect picture where both errors in the deep neural network, using the evaluation criterion *mean_absolute_error*, are about the same size. The optimal number of epochs is about 350, after that there is no real improvement.

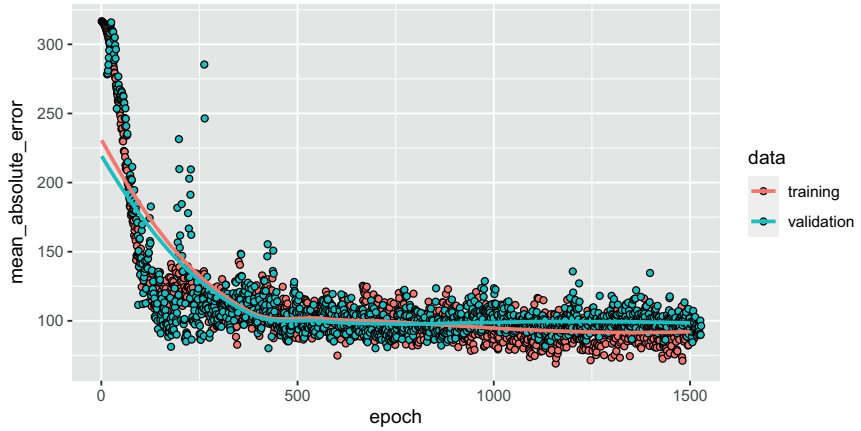

Fig. 3 Training and validation error for the imputation of variable Ag in the Moss data.

Regarding imputation, it is also interesting to look at an imputation error, as long as you know the true values (as in the numerical examples in Section 6). The imputation error over the validation measure CED (see Section 6.2) is lowest at about 200-300 epochs for all later used data.

Patient value: The aim is to stop at those number of epochs without any significant change of weights in the last epochs (patient value). Setting this value appropriately overcomes overfitting when specifying the number of epochs too high. For example, when the weights do not change for about 25 epochs (default), the training of the weights in the deep neural network would stop.

Drop out: This is the rate of input units in each layer, which are excluded in the neural network. The main purpose of using drop-out is to protect against over-matching, but also to achieve different results when imputing again if the random number generator is not set to the same value. This is also the only way to go in the direction of multiple imputation. Although proper multiple imputation (with correct coverage rates) is not possible to achieve in general using drop-out.

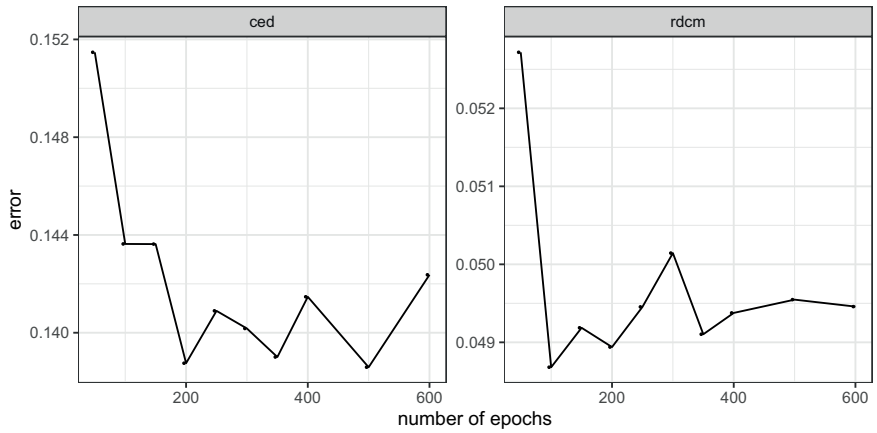

Fig. 4 Imputation error for different numbers of epochs for the imputation of variable Ag in the Moss data set. For our choice of the number of epochs for the numerical results section is based on the CED criteria (see Section 6.2).

Number and density of layers in the deep network: The optimal number of layers is data-dependent. In general, too many layers can cause an overfit, while too few layers can cause an underfit. Generally, it is better to add too many layers than too few. This is the reason why by default the number of layers in the deep neural network is set to 10.

The user can evaluate the optimal number of layers using cross-validation. The validation error is not very different from training terror, both errors are obtained from cross-validation.

Number of neurons in each layer: The optimal number of neurons in each layer depends on your data particularities. By default, the first hidden layers consist of 1000 neurons and then decrease by 100 in each following layer so that the 10th layer has 100 neurons. Note that the user can overwrite this default choice. Again cross-validation using different choices of neurons can give some hints about the optimal choice of the number of neurons.

Optimizer: The choice of the optimizer is crucial. It defines the gradient method used to iteratively update the weights in the neural network. Our default choice is Adam [Kingma and Ba, 2014], a first-order stochastic gradient-based optimization based on adaptive estimates of lower-order moments.

Other optimizers selectable are SGD (Stochastic gradient descent optimizer), RMSprop, Adagrad, Adadelta, Adamax, and Nadam (Nesterov Adam optimizer). Note that all methods have sensible defaults, e.g. on the learning rate.

Loss function: The choice of the loss function depends on the scale type of the response variable (the variable to be imputed). For nominal variables to be imputed, typically a cross-entropy is used, while for continuous variables a mean squared error is used. Note that we did not replace this with a compositional counterpart. This could be future work which would need careful implementation in software. Please also note that no robust loss function can be integrated into Keras, because no quantiles or any robust measure can be used to construct a robust loss measure. Also, the deep neural networks are computationally expensive, thus replacing a mean squared error with robust measures, the computation time would increase heavily. As a consequence, all deep neural networks are non-robust and sensitive to outliers.

Evaluation metrics: An evaluation metric is used to evaluate the performance of your model. A metric function is thus similar to a loss function. The latter is used for training the model while the evaluation metrics are used to judge the performance of the model. One can use any of the loss functions as a metric function. In our software implementation, the default evaluation metric is the mean absolute error.

Activation: Especially for compositional data, a good choice of an activation function is important, because the weights for neurons get transformed by the activation function. For example, when using the activation *linear*, the outcome of the neural network for a 2-dimensional continuous data set in a regression context is approximately the same as for least squares regression. To allow for non-linear combination of neurons, we choose by default the activation *reLU*, see Vedaldi and Lenc [2015]. One positive aspect of the reLU function is that it accelerates the convergence of stochastic gradient descent compared to other activation functions and it is computationally cheap [Krizhevsky et al., 2012]. Other choices are, e.g., softmax, elu, selu, or sigmoid, to just name a few. For the last layer (that actually do the imputation) a different choice of activation must be taken, namely linear.

5 Imputation of rounded zeros using artificial neural networks in an Expectation-Maximum style

The proposed approach uses one artificial network per variable to be imputed. Note that the structure of the output layer, the activation function, the loss, and evaluation function for the output layer depend on the distribution of a variable to be imputed. Naturally, this is a log-ratio, but in principle also external variables that are non-compositional could be integrated. Generally, for continuous measurements and log-ratios, the activation function relu [He et al., 2015] is used concerning all layers (default), the mean absolute error is used as a loss function (default) and the output layer consists of a single neuron and linear activation.

As already mentioned, the initialization of rounded zeros is necessary whenever artificial neural networks are applied to log-ratio representations instead of raw original values.

The algorithm works in a chain, i.e. it iterates sequentially applied to all variables until convergence (EM-based method).

All networks have 10 layers (by default), which in experimental pre-studies showed to be sufficient to reach a convergent result. It uses fully connected layers with a dropout rate of 10% after each layer to overcome overfitting and to allow for a kind of multiple imputation. In the first hidden layer 2 there are $_2$ 1000 neurons, as well as in the others except for the output layer. Note that this default choice can be overwritten by the user.

The layers are applied one after another, while the activations **a** of neurons in a first layer determine the activations of the second layer. In a final output layer, the desired imputation of missing values takes place.

Let **X** $\mathbf{x}_1$ $\mathbf{x}_2$ **x** be a -dimensional data matrix. **x** denotes the observed values of the -th variable, while **x** represents the missing values. We define two algorithms, while the first one serves as a benchmark when the compositional nature of compositional data is ignored.

Algorithm 1: Imputation of rounded zeros with ANNs (without using log-ratios)

Result: The imputed data matrix **X** in original scale
d vector of detection limits (one entry for each variable) ;
k vector of indices with colums in **X** that contains missing values ;
threshold for convergence with sufficient small value, e.g. set to 1 ;
number of rounded zeros in **X** ;
0 index for iterations and the maximum number of iterations ;
Initialization using aKNN (default) or similar methods ;
while *and* **do**
1 ;
X store previously imputed data matrix;
for *in* **k do**
Fit the deep neural net: **x** $\mathbf{x}_1$ $\mathbf{x}_2$ **x** **x** and predict **x**
X update imputed data matrix with **x** ;
Imputed values above are set to ;
end
update $\frac{1}{}$ **X** **X**
end

The second algorithm makes use of pivot log-ratio transformations. One particular choice of an orthogonal basis using one original composition $_1$ is

$$\frac{}{\frac{}{1}} \ln \frac{}{\frac{}{1}} \quad \text{for} \quad 1 \quad 1 \qquad (1)$$

These coordinates will be referred to as *pivot* (log-ratio) *coordinates* [Filzmoser et al., 2018].

As mentioned above, the pivot coordinates have the feature that part $\mathbf{x}_1$ only appears in coordinate $\mathbf{z}_1$. This is not the case for other parts; $\mathbf{x}_2$, for example, appears in $\mathbf{z}_1$ and in $\mathbf{z}_2$. Isolating one part into one coordinate is appealing, since $\mathbf{z}_1$ summarizes now all relative information (log ratios) about $\mathbf{x}_1$.

It is possible to come back to the original parts (up to a scaling factor) by the inverse pivot (log-ratio) transformation [see also Filzmoser et al., 2018]

$$
\begin{aligned}
&{}_1 \quad \exp \frac{\overline{}}{\overline{}} {}_1 \\
&\exp \quad {}_1^1 \frac{1}{\overline{}} \quad \frac{\overline{}}{\overline{}} \quad 2 \quad 1 \\
&\exp \quad {}_1^1 \frac{1}{\overline{}} \qquad (2)
\end{aligned}
$$

The scaling factor is different for each composition in the imputation context. For more details on this factor, we refer to Templ et al. [2016].

Algorithm 2: Imputation of rounded zeros with ANNs (with log-ratios)

Result: The imputed data matrix **X**
Use the same parameters as described in Algorithm 1 ;
Initialization using aKNN (default) or similar methods ;
while *and* **do**
  for *in* **k do**
    Sort the data matrix so that **x** is the first column ;
    Z apply a pivot log-ratio transformation on the sorted **X** ;
    Z store previously imputed data matrix;
    represent the detection limit regarding $\mathbf{z}_1$ in pivot log-ratio coordinates ;
    Fit the deep neural net: $\mathbf{z}_1$ $\mathbf{z}_2$ $\mathbf{z}_3$ **z** $_1$ and predict $\mathbf{z}_1$
    Z update imputed data matrix with $\mathbf{z}_1$;
    Imputed values above are set to ;
    Apply the inverse pivot log-ratio transformation ;
    Backsort to the original order of variables ;
    Adjust values so that absolute values are preserved ;
  end
  update
end

6 Numerical results

The numerical results are all based on real data. To evaluate the imputation of rounded zeros on real data, detection limits for compositional parts are artificially chosen. This way the imputations can be compared with the real observed values. Note that in principle simulation studies could also be done with artificially generated data. However, it is more realistic to work with real data - where detection limit problems occur in principle - and to set detection limits there. Furthermore, deep learning methods require a lot of computing time (see Figure 10) and therefore it is unrealistic to perform large simulation studies with many replications or varying an extensive grid of parameters for one real data set.

6.1 Data

Kola C-horizon and Moss data: The *Kola Ecogeochemistry Project* was a geochemical survey of the Barents region whose aim was to reveal the environmental conditions in the European arctic. Soil samples were taken at different levels and are linked to spatial coordinates. In this paper, the C-horizon and the Moss data [Reimann et al., 2008] are used. The C-horizon data represent soil samples at approximately 1 1/2 meters depth under the surface, the Moss data represent soil samples from the mosses on the surface. In the Moss data, one can see the environmental influence (e.g. pollution by nickel smelters) while in the C-horizon the type of rock has a decisive influence on the soil sample. The raw data sets, in which zeros are the result of element concentrations below the detection limit, are included in the R package [Templ et al., 2012, Kowarik and Templ, 2016]. For this study the main elements (Al, Ca, Fe, K, Mg, Mn, Na, P, Si, Ti) are selected for the C-horizon data and for the Moss data we selected the elements Ag, Al, Ca, Cu, Fe, Mg, Na, Ni, Pb, and Si. These elements did not include any values below the detection limit, thus by artificially setting a detection limit, we can compare the true value with the imputed ones. The number of observations is 606 (C-horizon) and 594 (Moss data).

Gemas data: The GEMAS (Geochemical Mapping of Agricultural and Grazing land Soil) project provides geochemical data on agricultural and grazing land soil. It documents the concentration of almost 60 chemical elements at the scale of Europe.

Element concentrations of 2108 samples of agricultural soil (in 0–20 cm depth) and grazing land soil (0–10 cm) are available in package robCompositions [Templ et al., 2011, Filzmoser et al., 2018]. The average sample density was 1 site per 2500 km^2. For our study we used all chemical elements of the data set: Al, Ba, Ca, Cr, Fe, K, Mg, Mn, Na, P, Si, Sr, Ti, V, Y, Zn, Zr.

TrondelagO data: A regional-scale geochemical survey of O-horizon samples in Nord-Trondelag including the concentrations of nine different plant types. The data set includes 754 observations and we used the chemical concentrations on Ag, Al,

As, Ca, Cr, Cu, Fe, Mg, Na, Ni, Pb, and S, that represents those chemical elements that are often used for analysis [see, e.g., Templ et al., 2008, Filzmoser et al., 2018].

For all data sets, an artificial detection limit was introduced. For each variable in the data set the 0.05 quantile represents the detection limit. This means that for each variable 5% of the smallest values are set to zero and are subsequently imputed with different methods.

6.2 Validation criteria

The validation criteria defined in the following [see also Martín-Fernández et al., 2012, Templ et al., 2016] assume that the complete data information is available. The symbol is used for imputed data or estimators thereof.

Relative difference between covariance matrices (RDCM): Let $\mathbf{S}$ be the sample covariance matrix of the original observations in pivot log-ratio coordinates from Equation (1) (or any other isometric log-ratio coordinates), and let $\mathbf{S}$ denote the sample covariance matrix computed with the same isometric log-ratio coordinates where all the rounded zeros have been imputed. A measure of the relative difference between both covariance matrices [Hron et al., 2010], based on the Frobenius matrix norm , is

$$\frac{\sqrt{\frac{1}{1^2}\sum_{1}^{1}\sum_{1}^{1}\quad{}^2}}{\frac{1}{1}\frac{\mathbf{S}\quad\mathbf{S}}{\mathbf{S}}} \tag{3}$$

Compositional error deviation (CED): Let denotes the number of samples $\mathbf{x}$ containing at least one rounded zero, the index set referring to such samples, and $\mathbf{x}$ the corresponding observation with imputed values. Then the criterion

$$\frac{\frac{1}{}\quad\mathbf{x}\quad\mathbf{x}}{\max_{\mathbf{x}\ \mathbf{x}\ \mathbf{X}}\quad\mathbf{x}\quad\mathbf{x}} \tag{4}$$

measures the differences by using Aitchison distances (defined below). stands for the Aitchison distance, which is defined for two compositions $\mathbf{x}$ and $\tilde{\mathbf{x}}$ as $\mathbf{x}\ \tilde{\mathbf{x}}\quad\frac{1}{}\ {}_1^1\quad{}_1\ \log\frac{}{}\quad\log\frac{\tilde{}}{\tilde{}}\quad{}^{2\ 1\ 2}$ [Aitchison et al., 2000]. This criterion represents a generalization of the measure defined in Hron et al. [2010] and it was used in Martín-Fernández et al. [2012] and Templ et al. [2016]. Note that the denominator is the maximum distance in the original data set.

Curious Imputations: The number of imputations below 0 and above the detection limit represents the imputation of rounded zeros that are not valid.

6.3 Results

Looking at the result for the imputation of the C-horizon data in Figure 5, it can be seen that, except for `deepImp-dl`, the best-judged methods are those which take the compositional nature of the data and the special problems for rounded zeros into account. The very best methods for the imputation of rounded zeros for the C-horizon data are `impRZilr`, `imputeBDL_pls` and `deepImpCoDa-dl`. Since `impRZilr` and `imputeBDL_pls` differ algorithmically only slightly and both methods represent an EM-algorithm with partial least squares regression, it is not surprising that almost identical results are obtained. Slightly better than `deepImp-dl` is `deepImpCoDa-dl`. This shows that a pivot log-ratio transformation does bring a slight improvement even when using ANNs. The two methods are also best for the evaluation criterion CED , which expresses normalized Aitchison distances between observed and imputed values. Comparatively bad are methods that represent pure imputation methods without taking the problem of rounded zeros into account. The benchmarks `deepImpCoDa` and `deepImp` are comparably worse than their rounded zeros counterparts. Surprisingly, `knn` scores slightly better than `aknn`.

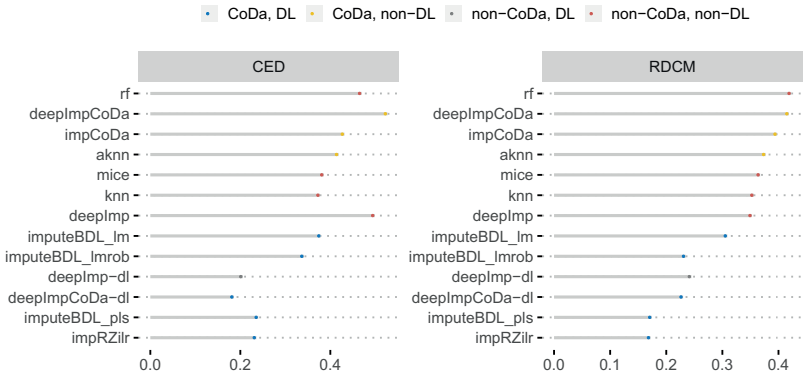

Fig. 5 Comparison of methods for the C-horizon data. Methods that take the compositional nature of the data and detection limits into account are shown in blue. The legend expresses the nature of the method. For example, *CoDa, DL* means the use of a compositional method that is designed to deal with detection limits. See Table 1 for details.

The results of the Moss data are shown in Figure 6. The two new methods, `deepImpCoDa-dl` (ced: 0.1539, RDCM: 0.0516) and `deepImp-dl` (ced: 0.1584, RDCM: 0.0479) are by far the best, followed by `aknn`. (ced: 0.4053, RDCM: 0.0856). So there is no difference whether algorithm 1 or algorithm 2 is used with respect to ANNs, i.e. the deep neural network learns the restrictions of the simplex indirectly and also maps those possibly non-linear relationships which may become linear or better represented by a log-ratio transformation. The other methods for rounded zeros

also perform relatively poorly, for example `imputeBDL_pls` (ced: 0.4136, RDCM: 0.1545). The benchmark methods `deepImp` and `deepImpCoDa` work very poorly, neither of which addresses the problem of detection limits.

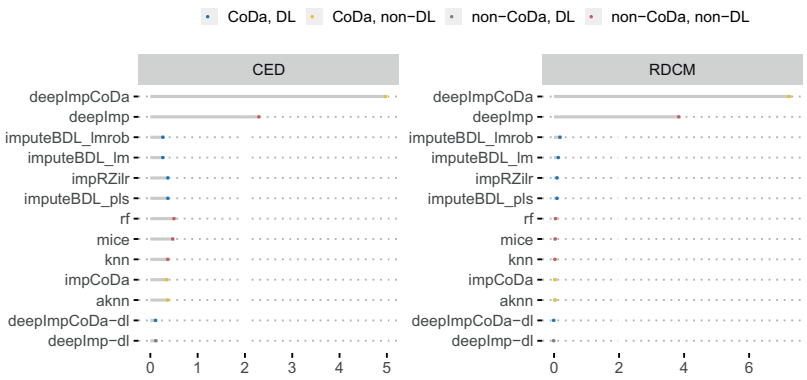

Fig. 6 Comparison of methods for the Moss data. Methods that take the compositional nature of the data and detection limits into account are shown in blue.

A very similar picture can be seen in the results for the TrondelagO data in Figure 8. Again, the two algorithms 1 (Benchmark: `deepImp-dl`, CED: 0.2608, RDCM: 0.1645) and algorithm 2 (CoDa-Adaption: `deepImpCoDa-dl`, CED: 0.2539, RDCM: 0.1642) performed best by far. Again, a representation in pivot log-ratio coordinates is not important and the deep neural network learns the relationships even in simplex. Compared to other methods, the rounded zeros methods also perform relatively well in the CED criterion, for example `imputeBDL_pls` (ced: 0.6508, RDCM: 0.2915) is better than for example mice (CED: 0.9552, RDCM: 0.2708). The two benchmarks `deepImp` and `deepImpCoDa` are comparatively poor. The reason is that a few imputed values were imputed very poorly.

By far the best rounded zero imputation method for the Gemas data is Algorithm 2 (`deepImpCoDa-dl`, CED: 0.1721, RDCM: 0.0971), followed by Benchmark Algorithm 1 (`deepImp-dl`, CED: 0.2538, RDCM: 0.1195). All other methods are comparatively poor, for example `imputeBDL_pls` (CED: 0.5201, RDCM: 0.5172). It is interesting that for the ANNs this time the pivot log-ratio transformations contribute to the improvement of the results. This means in this case that the ANNs did not learn the compositional structure of the data, but an isometric log-ratio transformation is needed to improve the results (compare `deepImpCoDa-dl` with `deepImp-dl` and `deepImpCoDa` with `deepImp`).

In Figure 9 one can see which imputation methods do not take detection limits into account and very often they impute values outside the interval (0, detection limit). While this is not such dramatic with the Moss and TrondelagO data, it is evident with the C-horizon and Gemas data. It is interesting to note that `deepImpCoD` imputes

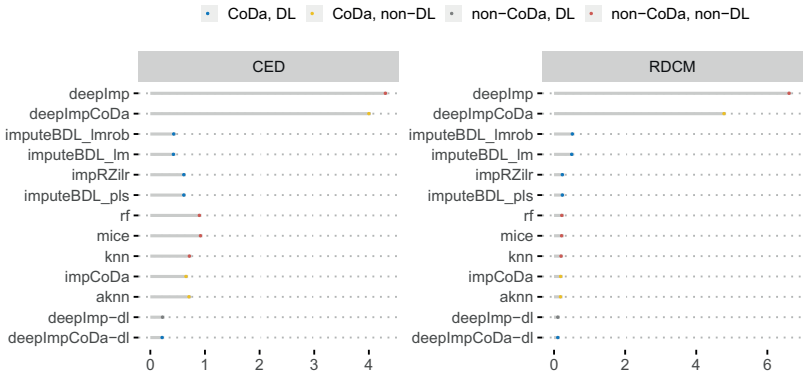

Fig. 7 Comparison of methods for the TrondelagO data. Methods that take the compositional nature of the data and detection limits into account are shown in blue.

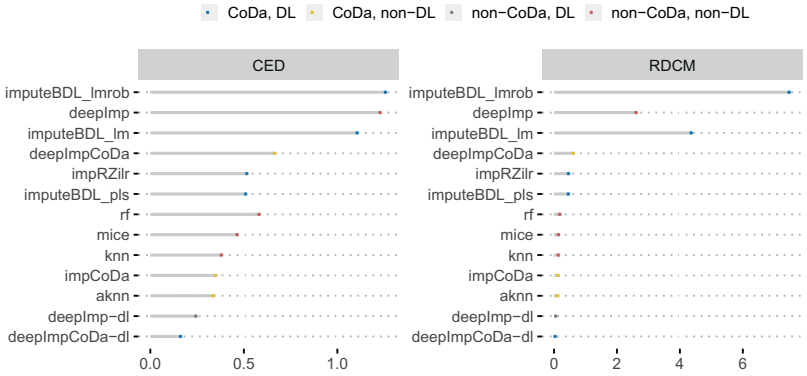

Fig. 8 Comparison of methods for the Gemas data. Methods that take the compositional nature of the data and detection limits into account are shown in blue.

comparatively fewer values outside the intervals than `deepImp`, since no values below 0 can be imputed with this method. The use of methods that do not take detection limits into account is generally critical, since many values outside the definition range, censored with 0 and detection limit, are imputed. Note that values outside the interval (0, DL] can be shrinked to the limits. This means that imputed values above the detection limit are winsorized to the detection limit, and values less than or equal to zero are set to zero + constant, where typically the constant is often chosen very small. The latter implies that outliers may be generated by a log-ratio transformation.

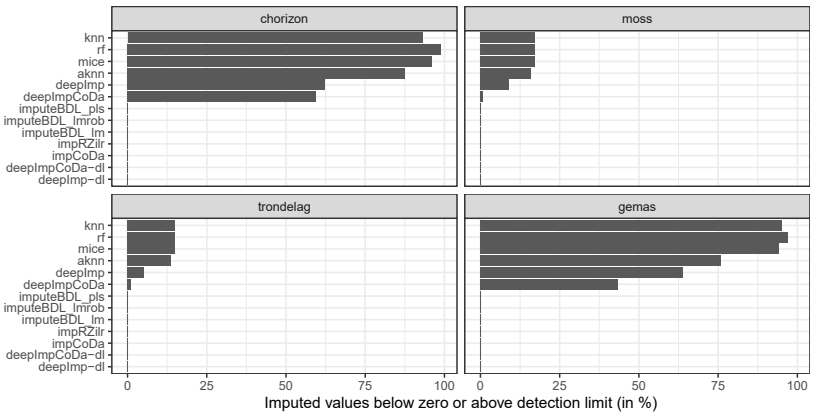

Fig. 9 Percent of imputed values above the detection limit or below 0.

6.4 Computation time

The computational cost of our ANN imputation framework is compared with the runtimes of other imputation procedures using a 2.6 GHz Intel Core i7 single cpu in Table 10. The kNN method implemented in package VIM [Templ et al., 2012, Kowarik and Templ, 2016] is the fastest in general, however, this method does not consider the rounded zero problem adequately. The runtimes of the EM-based methods using pivot coordinates in Martín-Fernández et al. [2012], Templ et al. [2016] are computationally faster than the artificial neural network methods by approximately a factor of 10. Note that should be generally faster than , but in the EM-based approach, the linear model approach takes longer to converge as with the robust method, i.e. the imputations need longer to stabilize and more iterations are necessary. This is also the case with .

The increase of the quality of the imputation using our ANN framework for the imputation of rounded zeros has a price regarding the computation time.

However, if needed, there are several ways to speed up the computation of our ANN framework, namely to use fewer layers, fewer neurons per layer, fewer epochs, and/or fewer iterations.

7 Conclusion

Up to our knowledge and literature research on the date of the submission of this paper, this is the first peer-reviewed contribution in compositional data analysis with ANN's and, more general, using deep learning approaches. The proposed new methods for the imputation of rounded zeros were compared not only with other methods

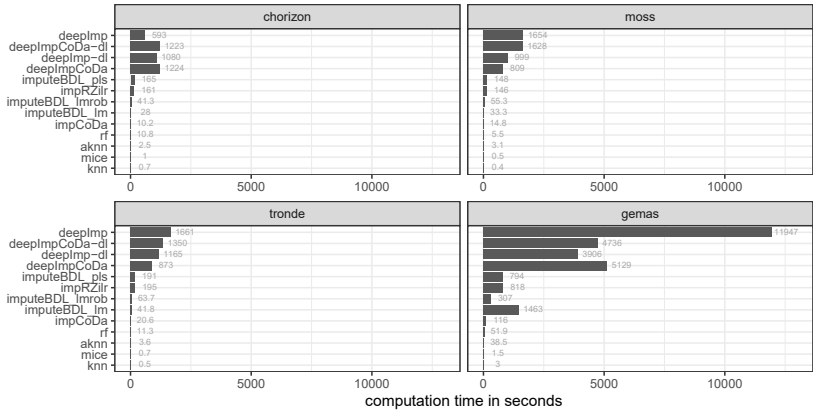

Fig. 10 Computation times on all data sets.

for the imputation of rounded zeros but also with popular conventional imputation methods. As already mentioned, this sounds a bit strange at first glance, since rounded zeros are exclusively censored data. The motivation for this comes from practical and subjective experience as package maintainer of imputation packages and author of many imputation functions. This subjective experience shows that many users and researchers haven't understood that the problem of rounded zeros requires specially designed methods and that users and researchers mostly make the wrong choice of imputation methods, although the articles and function descriptions are clear on the choice of methods. Also, with such a comparison we highlight the compositional data problem to research areas where compositional data are not well known. For this reason, we will compare exemplary imputation methods with general imputation methods to make this more obvious and evident. Note that a practitioner who neglect compositional data analysis methods could criticize that the compositional data methods are probabably only better because of the suggested compositinal evaluation measures. However, compositional data also require compositional validation critera. Furthermore, the results shown in Figure 9 are a strong argument against the use of non-compositional "classical" imputation methods because these methods are often used to impute above the detection limit.

Also, benchmark methods were determined. For example, Algorithm 1 is not adequate from a theoretical point of view, but it is interesting to compare it as a benchmark for Algorithm 2 to answer the question of how far results improve on real data when using log-ratio transformations.

The newly introduced method using ANNs ( ) is not only competitive but consistently gave the best results for all three data sets. The Moss data has better correlations between the variables than the C-horizon data - this gives a boost to the ANNs. The TrondelagO data have a good group structure concerning different plant species, which the ANNs learn better than other methods. The Gemas data are significantly larger than the other data sets, which has an impact on the quality of

the ANNs, which generally improve as more training, testing, and validation data are available. The presented imputation methodology with ANNs beats all other methods by far. Note that for other data sets it is likely to reveal different measures of performance caused by distinct multivariate correlation and dependency structures.

It is interesting to see that a representation in (pivot) log-ratio coordinates is not necessarily needed anymore, but the results hardly deteriorate if the data are not represented in log-ratio coordinates, because the limitations of the simplex and the coherences in the data are well learned in the deep neural network. Theoretically, however, the correct representation of the data also plays a role in deep neural networks, since the distances between observed values should be considered in a compositional way. We also see differences in the C-horizon results and especially in the results for the Gemas data set, where Algorithm 2 (with log-ratio transformations) performs better than the benchmark Algorithm 1 (without log-ratio transformations).

A drop of bitterness is the long computing times because an ANN with enough layers and neurons takes a long time to train. For typical data sets in geochemistry, for example, this is not a problem as long as no parameter optimization on the number of layers, neurons, loss-function, evaluation-function, activation function, epochs, etc. is made. In practice, the imputation is often performed exactly once, and therefore it doesn't matter if an algorithm is ready in a second or if it takes 4 hours. In any case, we provide sensible defaults and stop whenever the evaluation criteria will not improve after a given number of epochs. Multiple imputation is possible in principle and can be controlled by the parameter , which also can avoid overfitting.

The presented algorithms using ANNs are part of a larger R package for deep learning in imputation and compositional data analysis that is work in progress, and it will be published independently of this contribution.

We propose three topics for future research. Firstly, it would be interesting to compare the methods from the R package *zCompositions*. Although these methods are mostly of univariate nature or include a multiple EM-based method based on the alr transformation, in many cases rounded zeros are very well imputed. Secondly, large simulation studies may, under certain circumstances, lead to further knowledge about the quality of the methods. Thirdly, in this contribution we only looked at some well-known evaluation measures such as the CED and RDCM but further analysis is possible. For example the evaluation of classifiers on the imputed data can show which methods are more advantageous and lead to fewer missclassifications, and log-ratio biplots or comparison of results from a regression model can show what the effects of imputing rounded zeros are.

Acknowledgements I would like to thank Peter Filzmoser and Karel Hron for the many collaborations and the fruitful discussion on the topic of compositional data analysis, imputation, and rounded zeros. Furthermore, my thanks go to Eric Grunsky and Peter Filzmoser as well as to one unknown reviewer for their constructive and helpful comments on the initial submission.

References

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL . Software available from tensorflow.org.

M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, and C. Citro. TensorFlow: large-scale machine learning on heterogeneous systems, 2018. URL . Version: 1.10.0, Software available from tensorflow.org.

J. Aitchison. *The Statistical Analysis of Compositional Data*. Chapman & Hall, London, 1986.

J. Aitchison, C. Barceló-Vidal, J.A. Martín-Fernández, and V. Pawlowsky-Glahn. Logratio analysis and compositional distance. *Mathematical Geology*, 32(3): 271–275, 2000.

J.J. Allaire and F. Chollet. *keras: R Interface to 'Keras'*, 2019. URL . R package version 2.2.4.1.9001.

C. Arisdakessian, O. Poirion, B. Yunits, X. Zhu, and L.X. Garmire. Deepimpute: an accurate, fast, and scalable deep neural network method to impute single-cell rna-seq data. *Genome Biology*, 20(1):211, 2019. doi: 10.1186/s13059-019-1837-6.

J. Chen, X. Zhang, K. Hron, M. Templ, and S. Li. Regression imputation with q-mode clustering for rounded zero replacement in high-dimensional compositional data. *Journal of Applied Statistics*, 45(11):2067–2080, 2017. doi: 10.1080/02664763.2017.1410524.

F. Chollet et al. Keras. , 2015.

S.J. Choudhury and N.R. Pal. Imputation of missing data with neural networks for classification. *Knowledge-Based Systems*, 182:104838, 2019. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2019.07.009.

P. Filzmoser, K. Hron, and M. Templ. *Applied Compositional Data Analysis*. Springer International Publishing, 2018. ISBN 9783319964225. doi: 10.1007/978-3-319-96422-5.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, New York, 2nd edition, 2009. ISBN 978-0-387-84857-0.

K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification, 2015.

K. Hron, M. Templ, and P. Filzmoser. Imputation of missing values for compositional data using classical and robust methods. *Computational Statistics & Data Analysis*, 54(12):3095–3107, 2010. ISSN 0167-9473. doi: DOI: 10.1016/j.csda.2009.11.023.

J.M. Jerez, I. Molina, P.J. García-Laencina, E. Alba, N. Ribelles, M. Martń, and L. Franco. Missing data imputation using statistical & machine learning methods

in a real breast cancer problem. *Artificial Intelligence in Medicine*, 50(2):105–115, 2010. ISSN 0933-3657. doi: 10.1016/j.artmed.2010.05.002.

D.P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

A. Kowarik and M. Templ. Imputation with the R package VIM. *Journal of Statistical Software*, 74(7):1–16, 2016. doi: 10.18637/jss.v074.i07.

A. Krizhevsky, I. Sutskever, and G.E. Hinton. Imagenet classification with deep convolutional neural networks, 2012. Internet Resource, accessed on: 14.01.2019.

S.C-X. Li, B. Jiang, and B.M. Marlin. Misgan: Learning from incomplete data with generative adversarial networks. *CoRR*, abs/1902.09599, 2019. URL .

Y.C. Lim. Learning generative models from incomplete data. Technical Report CMU-CS-19-120, School of Computer Science, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213, 2019.

T. Maiti, C.P. Miller, and P.K. Mukhopadhyay. Neural network imputation: An experience with the national resources inventory survey. *Journal of Agricultural, Biological, and Environmental Statistics*, 13(3):255–269, 2008. ISSN 10857117.

J. Martín-Fernández, K. Hron, P. Templ, M. Filzmoser, and J. Palarea-Albaladejo. Model-based replacement of rounded zeros in compositional data: classical and robust approaches. *Computational Statistics and Data Analysis*, 56(9):2688–2704, 2012. doi: 10.1016/j.csda.2012.02.012.

J.A. Martín-Fernández, C. Barceló-Vidal, and V. Pawlowsky-Glahn. Dealing with zeros and missing values in compositional data sets using nonparametric imputation. *Mathematical Geology*, 35(3):253–278, 2003.

J.A. Martín-Fernández, J. Palarea-Albaladejo, and R.A. Olea. Dealing with zeros. In V. Pawlowsky-Glahn and A. Buccianti, editors, *Compositional Data Analysis: Theory and Applications*, pages 43–58. Wiley, Chichester, 2011.

J.A. Martín-Fernández, K. Hron, M. Templ, P. Filzmoser, and J. Palarea-Albaladejo. Bayesian-multiplicative treatment of count zeros in compositional data sets. *Statistical Modelling*, 15(2):134–158, 2015.

P-A. Mattei and J. Frellsen. missiwae: Deep generative modelling and imputation of incomplete data. *ArXiv*, abs/1812.02633, 2018.

M. Mayer. *missRanger: Fast Imputation of Missing Values*, 2019. URL . R package version 2.1.0.

J.T. McCoy, S. Kroon, and L. Auret. Variational autoencoders for missing data imputation with application to a simulated milling circuit. *IFAC-PapersOnLine*, 51(21):141 – 146, 2018. ISSN 2405-8963. doi: https://doi.org/10.1016/j.ifacol.2018.09.406. 5th IFAC Workshop on Mining, Mineral and Metal Processing MMM 2018.

M.A. Nielsen. *Neural networks & deep learning*, volume 25. Determination press, USA, 2015.

J. Palarea-Albaladejo and J. A. Martín-Fernández. A modified em alr-algorithm for replacing rounded zeros in compositional data sets. *Computer & Geosciences*, 34 (8):902–917, 2008.

J. Palarea-Albaladejo and J.A. Martín-Fernández. Values below detection limit in compositional chemical data. *Analytica Chimica Acta*, 764:32–43, 2013.

J. Palarea-Albaladejo, J.A. Martín-Fernández, and J. Gómez-García. A parametric approach for dealing with compositional rounded zeros. *Mathematical Geology*, 39(7):625–645, 2007.

J. Palarea-Albaladejo, J.A. Martín-Fernández, and R.A. Olea. A bootstrap estimation scheme for chemical compositional data with nondetects. *Journal of Chemometrics*, 28(7):585–599, 2014.

C. Reimann, P. Filzmoser, R.G. Garrett, and R. Dutter. *Statistical Data Analysis Explained: Applied Environmental Statistics with R*. Wiley, Chichester, 2008.

S. Ruder. An overview of gradient descent optimization algorithms, 2016. URL . arXiv: 1609.04747.

E-L. Silva-Ramírez, R. Pino-Mejías, and M. López-Coello. Single imputation with multilayer perceptron and multiple imputation combining multilayer perceptron and k-nearest neighbours for monotone patterns. *Applied Soft Computing*, 29:65 – 74, 2015. ISSN 1568-4946. doi: https://doi.org/10.1016/j.asoc.2014.09.052.

M. Smieja, U. Struski, J. Tabor, B. Zieliski, and P. Spurek. Processing of missing data by neural networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 2724–2734, Red Hook, NY, USA, 2018. Curran Associates Inc.

D.J. Stekhoven and P. Bühlmann. Missforest—non-parametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118, 10 2011. ISSN 1367-4803. doi: 10.1093/bioinformatics/btr597. URL .

M. Templ. *deepImp: Imputation with deep learning methods*, 2020. URL . R package version 1.0.0.

M. Templ, P. Filzmoser, and C. Reimann. Cluster analysis applied to regional geochemical data: Problems and possibilities. *Applied Geochemistry*, 23(8):2198 – 2213, 2008. ISSN 0883-2927. doi: 10.1016/j.apgeochem.2008.03.004. URL .

M. Templ, K. Hron, and P. Filzmoser. *robCompositions: An R-package for Robust Statistical Analysis of Compositional Data*, pages 341–355. John Wiley & Sons, Ltd, 2011. ISBN 9781119976462. doi: 10.1002/9781119976462.ch25. URL .

M. Templ, A. Alfons, and P. Filzmoser. Exploring incomplete data using visualization techniques. *Advances in Data Analysis and Classification*, 6(1):29–47, 2012. doi: DOI:10.1007/s11634-011-0102-y.

M. Templ, K. Hron, P. Filzmoser, and A. Gardlo. Imputation of rounded zeros for high-dimensional compositional data. *Chemometrics and Intelligent Laboratory Systems*, 155:183–190, 2016. doi: 10.1016/j.chemolab.2016.04.011. URL .

S. van Buuren and K. Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in R. *Journal of Statistical Software*, 45(3):1–67, 2011. URL .

K.G. van-den Boogaart, R. Tolosana-Delgado, and M. Templ. Regression with compositional response having unobserved components or below detection limit values. *Statistical Modelling*, 15(2):191–213, 2015.

A. Vedaldi and K. Lenc. Matconvnet: convolutional neural networks for MATLAB. In *Proceedings of the 23rd ACM international conference on Multimedia*, pages 689–692. ACM, 2015.

J. Xie, L. Xu, and E. Chen. Image denoising and inpainting with deep neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, page 341–349, Red Hook, NY, USA, 2012. Curran Associates Inc.

J. Yoon, J. Jordon, and M. van der Schaar. GAIN: missing data imputation using generative adversarial nets. *CoRR*, abs/1806.02920, 2018. URL

.