Using the missForestPredict package

Elena Albu

2022-01-25

Introduction

What is this document?

The goal of this document is to highlight the functionality implemented in the package missForestPredict and to provide guidance for the usage of this package.

Package information

The package missForestPredict implements the missing data imputation algorithm used in the R package missForest (Stekhoven and Bühlmann 2012) with adaptations for prediction settings. The function missForest is used to impute a (training) dataset with missing values and to learn imputations models that can be later used for imputing new observations. The function missForestPredict is used to impute one or multiple new observations (test set) using the models learned on the training data. The word "Predict" in the function name should not misguide the user. The function does not perform prediction of an outcome and is agnostic on whether the desired outcome for a prediction model is part of the training data or not; it will treat all columns of the provided data as variables.

Package functionality

Fast implementation

The imputation algorithm is based on random forests (Breiman 2001) as implemented in the ranger R package (Wright and Ziegler 2017). Ranger provides a fast implementation of random forests suitable for large datasets as well as high dimensional data.

Saved models and initialization

The missing data in each column is initialized with a mean/mode imputation or a custom imputation scheme (TODO). Each variable is then imputed using the iterative algorithm of missForest (Stekhoven and Bühlmann 2012) until a stopping criteria is met. The algorithm supports all variable types (continuous and categorical with two or more levels) and uses a common stopping criteria for a variables. The initialization used for the training data and the random forest models for each iteration are saved and can be later used to impute new observations.

Imputation of new observations

The models are applied iteratively to "predict" the missing values of each variable for the new observation, using the same number of iterations as used in the training data. Imputation initialization and models are "learned" also for variables with no missing values in the original (training) data. This allows for unfortunate situations in which new observations have different missing patterns than the one encountered in the training data (for example, because of accidental registration errors or because of unfortunate train / test split in which all missing values of a variable with low missingness fall in the test set).

Convergence criteria

At each iteration the out-of-bag (OOB) error is calculated for each variable separately. To obtain a global error the OOB errors for all variables are averaged (TODO: implement weighted average)

The normalized mean square error is used for both continuous and categorical variables. For continuous variables, it is equivalent to $1 - R^2$. For categorical variables, it is equivalent to $1 - BSS$ (Brier Skill Score).

Continuous variables:

$NMSE = \frac{\sum_{i=1}^{N}(x_i - \hat{x_i})^2}{\sum_{i=1}^{N}(x_i - \bar{x})^2} = 1 - R^2$, $i = 1, 2, ...N$

$\bar{x}$ = the mean value of variable x

$\hat{x_i}$ = prediction (imputation) for observation i

N = number of observations

Categorical variables:

$NMSE = \frac{BS}{BSref} = 1 - BSS$

$BS = \frac{1}{N}\sum_{j=1}^{R}\sum_{i=1}^{N}(p_{ij} - x_{ij})^2$, $i = 1, 2, ...N$, $j = 1, 2, ...R$

$BSref = \frac{1}{N}\sum_{j=1}^{R}\sum_{i=1}^{N}(p_j - x_{ij})^2 = 1 - \sum_{j=1}^{R} p_j^2$

p_{ij} = prediction (probability) for observation i and class j

p_j = proportion of the event in class j

N = number of observations

R = number of classes

The Brier Score (BS) is calculated as the sum of squared distances between the predictions (as probabilities) and the true values (0 or 1) for each class. The reference Brier Score ($BSref$) is calculated as the Brier Score of a predictor that predicts the proportion of the event in each class (Brier and others 1950).

TODO: Ordinal variables are treated as categorical? TODO: read paper

Imputation error monitoring

TODO - separate vignette

Extended options for binary variables

TODO. This might prove useful in imputation of sparse binary variables.

Support for dataframe and tibble

Both dataframe (data.frame class) and tibble (tbl_df class) are supported as input for the package functions.

How to install

The R package missforestpredict is for the moment only available on the KU Leuven gitlab. Only KU Leuven gitlab users can install the package.

```
library(devtools)
devtools::install_git('https://gitlab.kuleuven.be/u0143313/missforestpredict/')
```

How to use the package

Data used for demonstration

Iris data The iris dataset contains in R base contains 4 continuous variables and one categorical variable with three categories for N = 150 flowers.

Diamonds data The diamonds dataset from ggplot2 R package contains seven continuous variables and three categorical variables for N = 53940 diamonds

Imputation of training and test set

After installing the package you can load it in your R sessions with:

```
library(missForestPredict)
#> Loading required package: ranger
```

We will load the iris dataset and split it in a training set (100 observations) and a test set (50 observations).

```
data(iris)

N <- nrow(iris)
n_test <- floor(N/3)

set.seed(2022)
id_test <- sample(1:N, n_test)

iris_train <- iris[-id_test,]
iris_test <- iris[id_test,]
```

We produce 10% random missing values on each column in both the training and the test set.

```
set.seed(2022)
iris_train_miss <- prodNA(iris_train, noNA = 0.1)
iris_test_miss <- prodNA(iris_test, noNA = 0.1)

head(iris_train_miss)
#>    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
#> 2            NA         3.0          1.4         0.2  setosa
#> 4           4.6         3.1          1.5         0.2  setosa
#> 5            NA         3.6          1.4         0.2  setosa
#> 8           5.0         3.4          1.5         0.2  setosa
#> 9           4.4         2.9          1.4         0.2  setosa
#> 10          4.9         3.1           NA         0.1  setosa
head(iris_test_miss)
#>     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
#> 55            NA         2.8          4.6         1.5 versicolor
#> 75           6.4         2.9          4.3         1.3       <NA>
#> 6             NA         3.9          1.7         0.4     setosa
#> 123          7.7         2.8          6.7         2.0  virginica
#> 14           4.3         3.0           NA         0.1     setosa
#> 7            4.6          NA          1.4         0.3     setosa
```

We will impute the training set and learn the random forest imputation models at the same time using the function *missForest*. Observe that we set *verbose = TRUE* to monitor the number of iterations and the errors. More information on error monitoring is provided in a separate vignette. (TODO)

```
set.seed(2022)
iris_train_imp_object <- missForestPredict::missForest(iris_train_miss, verbose = TRUE)
#>   missForest iteration 1 in progress...done!
#>     OOB error(s) MSE (ranger):     0.1828988 0.1031638 0.09714786 0.07252696 0.04587486
#>     OOB error(s) MSE (corrected):  0.1828988 0.1031638 0.09714786 0.07252696 0.03054371
#>     OOB error(s) NMSE:             0.2638364 0.5572062 0.03101873 0.1241864 0.1380252
#>     difference(s):                 Inf Inf Inf Inf Inf
#>     difference(s) total:           Inf
#>     time: 0.07 seconds
#>
#>   missForest iteration 2 in progress...done!
#>     OOB error(s) MSE (ranger):     0.1212292 0.1032027 0.07512364 0.0374316 0.0456604
#>     OOB error(s) MSE (corrected):  0.1212292 0.1032027 0.07512364 0.0374316 0.03039744
#>     OOB error(s) NMSE:             0.1748763 0.5574163 0.02398652 0.06409335 0.1373643
#>     difference(s):                 0.08896014 -0.0002100449 0.007032203 0.06009304 0.0006609837
#>     difference(s) total:           0.1565363
#>     time: 0.07 seconds
#>
#>   missForest iteration 3 in progress...done!
#>     OOB error(s) MSE (ranger):     0.1192184 0.09571288 0.06872544 0.03591295 0.04663554
#>     OOB error(s) MSE (corrected):  0.1192184 0.09571288 0.06872544 0.03591295 0.03102257
#>     OOB error(s) NMSE:             0.1719757 0.5169622 0.02194362 0.06149299 0.1401891
#>     difference(s):                 0.002900608 0.04045404 0.002042904 0.00260036 -0.002824893
#>     difference(s) total:           0.04517302
#>     time: 0.07 seconds
#>
#>   missForest iteration 4 in progress...done!
#>     OOB error(s) MSE (ranger):     0.1153429 0.09688349 0.07231865 0.0360973 0.04712426
#>     OOB error(s) MSE (corrected):  0.1153429 0.09688349 0.07231865 0.0360973 0.03137361
#>     OOB error(s) NMSE:             0.1663852 0.5232849 0.02309091 0.06180865 0.1417755
#>     difference(s):                 0.005590505 -0.006322669 -0.001147287 -0.0003156575 -0.001586343
#>     difference(s) total:           -0.003781452
#>     time: 0.07 seconds
```

The imputed training set can be found by extracting ximp dataframe from the object.

```
iris_train_imp <- iris_train_imp_object$ximp

head(iris_train_imp)
#>    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
#> 2      4.691556         3.0     1.400000         0.2  setosa
#> 4      4.600000         3.1     1.500000         0.2  setosa
#> 5      5.082323         3.6     1.400000         0.2  setosa
#> 8      5.000000         3.4     1.500000         0.2  setosa
#> 9      4.400000         2.9     1.400000         0.2  setosa
#> 10     4.900000         3.1     1.435052         0.1  setosa
```

We will further impute the test set using the learned imputation models. The function *missForestPredict* will:

- initialize the missing values in each variable with the initialization "learned" from the training set (e.g: mean / mode)
- imperatively predict the missing values of each variable using the learned random forest models for each iteration

```
iris_test_imp <- missForestPredict::missForestPredict(iris_train_imp_object,
                                                       newdata = iris_test_miss)

head(iris_test_imp)
#>     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
#> 55      6.329693    2.800000      4.60000         1.5 versicolor
#> 75      6.400000    2.900000      4.30000         1.3 versicolor
#> 6       5.209457    3.900000      1.70000         0.4     setosa
#> 123     7.700000    2.800000      6.70000         2.0  virginica
#> 14      4.300000    3.000000      1.36519         0.1     setosa
#> 7       4.600000    3.004022      1.40000         0.3     setosa
```

Imputation of a single new observation

missForestPredict can impute a new observation with missing values.

```
single_observation <- iris_test_miss[1,]
single_observation[1,2] <- NA

print(single_observation)
#>    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
#> 55           NA          NA          4.6         1.5 versicolor

single_observation_imp <- missForestPredict::missForestPredict(iris_train_imp_object,
                                                                newdata = single_observation)

print(single_observation_imp)
#>    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
#> 55     6.277947    2.773944          4.6         1.5 versicolor
```

Imputation can be done even when all variables are missing for a single new observation. This is of course, not an ideal situation and probably not realistic, but it can be technically handled by the *missForestPredict* function. The observation will be initialized with the "mean observation" and the random forest models will impute iteratively.

missForestPredict package can impute observations with new missingness patterns not present in the training data as well as ariables with no missingness (complete) in training data. The iniialization and the random forest imputation models are "learned" for all variables in the dataset, regardless of the amount of missingness. The only condition is that the new observations follow the same data types as the ones in the training set.

```
single_observation <- iris_test_miss[2,]
single_observation[,1:4] <- NA_real_

print(single_observation)
#>    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
#> 75           NA          NA           NA          NA    <NA>
```

```
single_observation_imp <- missForestPredict::missForestPredict(iris_train_imp_object,
                                                               newdata = single_observation)

print(single_observation_imp)
#>    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
#> 75     6.325027     2.84886     4.694795    1.460062 versicolor
```

Predict without storing the imputed matrix

The function *missForest* returns both the imputed training dataframe as well as the imputation models.

```
str(iris_train_imp_object, max.level = 1)
#> List of 8
#>  $ ximp              :'data.frame':   100 obs. of  5 variables:
#>  $ OOBerror          : Named num [1:5] 0.1192 0.0957 0.0687 0.0359 0.0466
#>   ..- attr(*, "names")= chr [1:5] "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width" ...
#>  $ err_OOB_corrected: Named num [1:5] 0.1192 0.0957 0.0687 0.0359 0.031
#>   ..- attr(*, "names")= chr [1:5] "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width" ...
#>  $ init              :List of 5
#>  $ models            :List of 4
#>  $ impute_sequence  : chr [1:5] "Sepal.Length" "Petal.Width" "Sepal.Width" "Petal.Length" ...
#>  $ iter              : num 4
#>  $ maxiter           : num 10
#>  - attr(*, "class")= chr "missForest"
```

The imputed training data is though not necessary for imputing the test set. To avoid storing further these data in the object, *ximp* can be set to NULL.

```
iris_train_imp <- iris_train_imp_object$ximp
iris_train_imp_object$ximp <- NULL

iris_test_imp <- missForestPredict::missForestPredict(iris_train_imp_object,
                                                      newdata = iris_test_miss)

head(iris_test_imp)
#>     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
#> 55      6.329693    2.800000      4.60000         1.5 versicolor
#> 75      6.400000    2.900000      4.30000         1.3 versicolor
#> 6       5.209457    3.900000      1.70000         0.4     setosa
#> 123     7.700000    2.800000      6.70000         2.0  virginica
#> 14      4.300000    3.000000      1.36519         0.1     setosa
#> 7       4.600000    3.004022      1.40000         0.3     setosa
```

Impute larger datasets by adapting *num.trees* or *max.iter*

Although *missForestPredict* benefits of the improved computation time of *ranger* package, larger dataset can still prove time consuming to impute.

We will load the diamonds dataset, which contains more than 50000 observations and produce 30% missing values on each variable.

```r
library(ggplot2)

data(diamonds)

class(diamonds)
#> [1] "tbl_df"     "tbl"        "data.frame"

N <- nrow(diamonds)
n_test <- floor(N/3)

set.seed(2022)
id_test <- sample(1:N, n_test)

diamonds_train <- diamonds[-id_test,]
diamonds_test <- diamonds[id_test,]

diamonds_train_miss <- prodNA(diamonds_train, noNA = 0.1)
diamonds_test_miss <- prodNA(diamonds_test, noNA = 0.1)

head(diamonds_train_miss)
#> # A tibble: 6 x 10
#>   carat cut       color clarity depth table price     x     y     z
#>   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
#> 1  0.23 Ideal     E     <NA>     61.5    55   326  3.95  3.98  2.43
#> 2  0.21 Premium   E     SI1      59.8    61   326  3.89  3.84 NA
#> 3  0.23 Good      E     VS1      56.9    65   327  4.05  4.07  2.31
#> 4  0.29 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
#> 5  0.24 Very Good <NA>  VVS2     NA      57   336  3.94  3.96  2.48
#> 6  0.24 Very Good I     VVS1     62.3    57   336 NA     3.98  2.47
head(diamonds_test_miss)
#> # A tibble: 6 x 10
#>   carat cut     color clarity depth table price     x     y     z
#>   <dbl> <ord>   <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
#> 1  0.33 Ideal   E     VVS1     61.9  57    1312  4.43  4.46  2.75
#> 2  0.3  Ideal   H     SI1      62.6  53.1   475  4.27  4.3   2.69
#> 3  0.4  Ideal   I     <NA>     62    56    1240  4.74  4.77 NA
#> 4  0.73 Ideal   F     SI1      61.7  55    3249  5.79  5.82  3.58
#> 5  0.3  Premium E     SI2      61.9  58     540  4.31  4.28  2.66
#> 6  1.01 Fair    F     VS2      64.8  NA    4791  6.3   6.25  4.07
```

The function *missForest* supports additional arguments to be passed to the *ranger* function. By default, the default values of *ranger* are used. By default, the number of trees in the forest will be 500. This can be overridden by passing *num.trees = 100*, for example. Using less trees will prove to be computationally more efficient.

```r
set.seed(2022)
diamonds_train_imp_object <- missForestPredict::missForest(diamonds_train_miss,
                                                           verbose = TRUE,
                                                           num.trees = 100)
#>   missForest iteration 1 in progress...done!
#>     OOB error(s) MSE (ranger):     0.0003787425 0.2192521 0.5138894 0.3403084 0.2342903 2.292686 466
#>     OOB error(s) MSE (corrected):  0.0003787425 0.07277755 0.09668604 0.06206883 0.2342903 2.292686
#>     OOB error(s) NMSE:             0.001687859 0.5082415 0.8065911 0.6036099 0.1135731 0.4628146 0.0
#>     difference(s):                 Inf Inf Inf Inf Inf Inf Inf Inf Inf Inf
#>     difference(s) total:           Inf
#>     time: 11.14 seconds
#>
#>   missForest iteration 2 in progress...done!
#>     OOB error(s) MSE (ranger):     0.0003431825 0.1861123 0.456954 0.3309383 0.1999283 2.263814 3161
#>     OOB error(s) MSE (corrected):  0.0003431825 0.06344201 0.08840567 0.06051051 0.1999283 2.263814
#>     OOB error(s) NMSE:             0.001529387 0.4430468 0.7375131 0.5884555 0.09691592 0.4569864 0.
#>     difference(s):                 0.0001584725 0.06519469 0.06907799 0.01515441 0.01665713 0.005828
#>     difference(s) total:           0.1854763
#>     time: 11.34 seconds
#>
#>   missForest iteration 3 in progress...done!
#>     OOB error(s) MSE (ranger):     0.0003454045 0.1861467 0.4543205 0.3314173 0.2020122 2.271731 318
#>     OOB error(s) MSE (corrected):  0.0003454045 0.06345301 0.08797196 0.06058643 0.2020122 2.271731
#>     OOB error(s) NMSE:             0.001539289 0.4431237 0.733895 0.5891938 0.09792612 0.4585845 0.0
#>     difference(s):                 -9.902338e-06 -7.685487e-05 0.003618124 -0.000738298 -0.001010194
#>     difference(s) total:           -4.61943e-07
#>     time: 11.47 seconds

# impute test set
diamonds_train_imp_object$ximp <- NULL
diamonds_test_imp <- missForestPredict::missForestPredict(diamonds_train_imp_object,
                                                          newdata = diamonds_test_miss)

head(diamonds_test_imp)
#> # A tibble: 6 x 10
#>   carat cut     color clarity depth table price     x     y     z
#>   <dbl> <ord>   <ord> <ord>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
#> 1  0.33 Ideal   E     VVS1     61.9  57    1312  4.43  4.46  2.75
#> 2  0.3  Ideal   H     SI1      62.6  53.1   475  4.27  4.3   2.69
#> 3  0.4  Ideal   I     IF       62    56    1240  4.74  4.77  2.94
#> 4  0.73 Ideal   F     SI1      61.7  55    3249  5.79  5.82  3.58
#> 5  0.3  Premium E     SI2      61.9  58     540  4.31  4.28  2.66
#> 6  1.01 Fair    F     VS2      64.8  57.5  4791  6.3   6.25  4.07
```

Alternatively, the *maxiter* parameter can be set to a lower number (the default is 10).

```
#set.seed(2022)
#diamonds_train_imp_object <- missForestPredict::missForest(diamonds_train_miss, verbose = TRUE, maxite
#
## impute test set
#diamonds_train_imp_object$ximp <- NULL
#diamonds_test_imp <- missForestPredict::missForestPredict(diamonds_train_imp_object, newdata = diamond
#
#head(diamonds_test_imp)
```

TODO: test myself what seems to be preferable (on diamonds dataset, maxiter or num.trees)

Perform imputation in descending order

TODO: test if it's worth it

References

Breiman, L. 2001. "Random forests." *Machine Learning* 45 (1): 5–32.

Brier, Glenn W, and others. 1950. "Verification of Forecasts Expressed in Terms of Probability." *Monthly Weather Review* 78 (1): 1–3.

Stekhoven, Daniel J, and Peter Bühlmann. 2012. "MissForest—Non-Parametric Missing Value Imputation for Mixed-Type Data." *Bioinformatics* 28 (1): 112–18.

Wright, Marvin N., and Andreas Ziegler. 2017. "Ranger: A Fast Implementation of Random Forests for High Dimensional Data in c++ and r." *Journal of Statistical Software* 77 (1): 1–17. https://doi.org/10.18637/jss.v077.i01.