Using the missForestPredict package

Elena Albu

2022-02-09

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 to be imputed.

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 the median/mode (or mean/mode) of that variable derived on complete cases or a custom imputation scheme. 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 a weighted average is used, that can be controlled by the `OOB_weights` parameter. By default the weights are set to the proportion of missing values of each variable.

The normalized mean squared 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) (Brier and others 1950).

More information on convergence criteria and error monitoring is provided in a 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. Currently matrix input is not supported.

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/', dependencies = TRUE)
```

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 (Anderson 1935).

Diamonds data The diamonds dataset from `ggplot2` R package contains seven continuous variables and three categorical variables for N = 53940 diamonds (Wickham 2016).

Imputation of training and test set

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

```
library(missForestPredict)
```

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)
```

```r
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.

```r
set.seed(2022)
iris_train_miss <- produce_NA(iris_train, proportion = 0.1)
iris_test_miss <- produce_NA(iris_test, proportion = 0.1)

head(iris_train_miss)
#>    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
#> 2           4.9          NA          1.4         0.2    <NA>
#> 4           4.6         3.1          1.5         0.2  setosa
#> 5           5.0         3.6          1.4          NA    <NA>
#> 8           5.0         3.4          1.5         0.2  setosa
#> 9           4.4         2.9          1.4         0.2  setosa
#> 10           NA         3.1          1.5         0.1  setosa
head(iris_test_miss)
#>     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
#> 55           6.5         2.8          4.6         1.5 versicolor
#> 75           6.4         2.9           NA         1.3 versicolor
#> 6            5.4         3.9          1.7         0.4     setosa
#> 123          7.7         2.8          6.7         2.0  virginica
#> 14           4.3          NA          1.1          NA     setosa
#> 7            4.6         3.4          1.4         0.3       <NA>
```

We will impute the training set and learn the random forest imputation models at the same time using the function `missForest`. By default feedback on the number of iterations and the error monitoring is provided. You can set `verbose = FALSE` to silence this output. More information on error monitoring is provided in a separate vignette.

```r
set.seed(2022)
iris_train_imp_object <- missForestPredict::missForest(iris_train_miss)
#>   missForest iteration 1 in progress...done!
#>     OOB errors MSE:             0.184386757911946, 0.125508832271481, 0.190885886904061, 0.047429395
#>     OOB errors NMSE:            0.265052345259937, 0.671430214917994, 0.0632865445207545, 0.0789602
#>     (weigthed) difference NMSE: 0.756916857370376
#>     time:                       0.0799999999999983 seconds
#>
#>   missForest iteration 2 in progress...done!
#>     OOB errors MSE:             0.144011391454906, 0.104539650777098, 0.0875392899227555, 0.04320849
#>     OOB errors NMSE:            0.207013548486481, 0.559252117308309, 0.0290228851323939, 0.0719333
#>     (weigthed) difference NMSE: 0.0454072457888052
#>     time:                       0.100000000000001 seconds
#>
#>   missForest iteration 3 in progress...done!
#>     OOB errors MSE:             0.14405950262543, 0.101993081556839, 0.0846137888551251, 0.04379972
#>     OOB errors NMSE:            0.207082707349759, 0.545628824923885, 0.028052960867353, 0.07291761
#>     (weigthed) difference NMSE: 0.000296087285244218
```

```
#>     time:                        0.0700000000000003 seconds
#>
#>   missForest iteration 4 in progress...done!
#>     OOB errors MSE:              0.142166307243506, 0.102869771851575, 0.0836663522458388, 0.04271833
#>     OOB errors NMSE:             0.20436127614886, 0.550318824363425, 0.0277388465547305, 0.07111732
#>     (weigthed) difference NMSE: 0.0032691712273593 4
#>     time:                        0.0799999999999983 seconds
#>
#>   missForest iteration 5 in progress...done!
#>     OOB errors MSE:              0.137293319624421, 0.101606569777187, 0.089526189540059, 0.04254208
#>     OOB errors NMSE:             0.197356452095942, 0.543561116360402, 0.0296816243044 1, 0.07082389
#>     (weigthed) difference NMSE: -0.000158890129691391
#>     time:                        0.100000000000001 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      4.900000    3.311402          1.4   0.2000000  setosa
#> 4      4.600000    3.100000          1.5   0.2000000  setosa
#> 5      5.000000    3.600000          1.4   0.2612063  setosa
#> 8      5.000000    3.400000          1.5   0.2000000  setosa
#> 9      4.400000    2.900000          1.4   0.2000000  setosa
#> 10     4.786647    3.100000          1.5   0.1000000  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 (median/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           6.5     2.80000     4.600000   1.5000000 versicolor
#> 75           6.4     2.90000     4.433403   1.3000000 versicolor
#> 6            5.4     3.90000     1.700000   0.4000000     setosa
#> 123          7.7     2.80000     6.700000   2.0000000  virginica
#> 14           4.3     3.17691     1.100000   0.2101767     setosa
#> 7            4.6     3.40000     1.400000   0.3000000     setosa
```

Imputation of a single new observation

`missForestPredict` can impute a new observation with missing values. The new observation has to be provided as a dataframe with one row and with named columns (the column names have to correspond to the column names of the training set).

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

print(single_observation)
#>    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
#> 55          6.5          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          6.5    3.008043          4.6         1.5 versicolor
```

`missForestPredict` package can impute observations with new missingness patterns not present in the training data as well as variables with no missingness (complete) in training data. The initialization and the random forest imputation models are "learned" for all variables in the dataset, regardless of the amount of missingness.

Predict without storing the imputed matrix

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

```r
str(iris_train_imp_object, max.level = 1)
#> List of 8
#>  $ ximp           :'data.frame': 100 obs. of  5 variables:
#>  $ init           :List of 5
#>  $ initialization : chr "median/mode"
#>  $ impute_sequence: chr [1:5] "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width" ...
#>  $ maxiter        : num 10
#>  $ models         :List of 5
#>  $ err_MSE        :'data.frame': 5 obs. of  5 variables:
#>  $ err_NMSE       :'data.frame': 5 obs. of  5 variables:
#>  - 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.

```r
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           6.5     2.80000     4.600000   1.5000000 versicolor
#> 75           6.4     2.90000     4.433403   1.3000000 versicolor
#> 6            5.4     3.90000     1.700000   0.4000000     setosa
#> 123          7.7     2.80000     6.700000   2.0000000  virginica
#> 14           4.3     3.17691     1.100000   0.2101767     setosa
#> 7            4.6     3.40000     1.400000   0.3000000     setosa
```

In `ximp` is set to NULL by mistake, and the training set imputation is lost, it can be recovered without rerunning the algorithm. Imputing the training set with the function `missForestPredict` will give the same results as the initial results of `missForest` function because the same models are applied to the variables in the same order and for the same number of iterations.

```
set.seed(2022)
iris_train_imp_object <- missForestPredict::missForest(iris_train_miss)
#>   missForest iteration 1 in progress...done!
#>     OOB errors MSE:             0.184386757911946, 0.125508832271481, 0.190885886904061, 0.047429395
#>     OOB errors NMSE:            0.265052345259937, 0.671430214917994, 0.0632865445207545, 0.0789602
#>     (weigthed) difference NMSE: 0.756916857370376
#>     time:                       0.0700000000000003 seconds
#>
#>   missForest iteration 2 in progress...done!
#>     OOB errors MSE:             0.144011391454906, 0.104539650777098, 0.0875392899227555, 0.0432084
#>     OOB errors NMSE:            0.207013548486481, 0.559252117308309, 0.0290228851323939, 0.0719333
#>     (weigthed) difference NMSE: 0.0454072457888052
#>     time:                       0.0999999999999979 seconds
#>
#>   missForest iteration 3 in progress...done!
#>     OOB errors MSE:             0.14405950262543, 0.101993081556839, 0.0846137888551251, 0.0437997
#>     OOB errors NMSE:            0.207082707349759, 0.545628824923885, 0.028052960867353, 0.0729176
#>     (weigthed) difference NMSE: 0.000296087285244218
#>     time:                       0.109999999999999 seconds
#>
#>   missForest iteration 4 in progress...done!
#>     OOB errors MSE:             0.142166307243506, 0.102869771851575, 0.0836663522458388, 0.0427183
#>     OOB errors NMSE:            0.20436127614886, 0.550318824363425, 0.0277388465547305, 0.0711173
#>     (weigthed) difference NMSE: 0.00326917122735934
#>     time:                       0.23 seconds
#>
#>   missForest iteration 5 in progress...done!
#>     OOB errors MSE:             0.137293319624421, 0.101606569777187, 0.089526189540059, 0.0425420
#>     OOB errors NMSE:            0.197356452095942, 0.543561116360402, 0.02968162430441, 0.0708238968
#>     (weigthed) difference NMSE: -0.000158890129691391
#>     time:                       0.0600000000000023 seconds

# store imputed dataframe
iris_train_imp <- iris_train_imp_object$ximp
iris_train_imp_object$ximp <- NULL

# re-impute the same dataframe using missForestPredict
iris_train_imp_2 <-  missForestPredict::missForestPredict(iris_train_imp_object,
                                                          newdata = iris_train_miss)

identical(iris_train_imp, iris_train_imp_2)
#> [1] TRUE
```

Impute larger datasets by adapting `num.trees`

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.

```
library(ggplot2)

data(diamonds)

# split train / test
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 <- produce_NA(diamonds_train, proportion = 0.1)
diamonds_test_miss <- produce_NA(diamonds_test, proportion = 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     <NA>  SI2      61.5    55   326  3.95  3.98  2.43
#> 2  0.21 Premium   E     <NA>     59.8    61   326  3.89  3.84  2.31
#> 3  0.23 Good      E     VS1      56.9    65   327  4.05  4.07  2.31
#> 4  0.29 Premium   I     <NA>     62.4    58   334  4.2   4.23  2.63
#> 5  0.24 Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
#> 6  0.24 Very Good I     VVS1     NA      57   336  3.95  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   <NA>  <NA>     61.9  57    1312  4.43  4.46  2.75
#> 2  0.3  Ideal   H     SI1      62.6  53.1   475 NA     4.3   2.69
#> 3  0.4  Ideal   I     VVS2     62    56    1240  4.74  4.77  2.95
#> 4  0.73 Ideal   F     SI1      61.7  55    3249  5.79  5.82  3.58
#> 5  0.3  Premium E     SI2      NA    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 parameters to be passed to the `ranger` function. By default, the default values of `ranger` are used (e.g. the number of trees in the forest is 500). This can be overridden by passing `num.trees = 100`. Using less trees will prove to be computationally more efficient.

```
set.seed(2022)
diamonds_train_imp_object <- missForestPredict::missForest(diamonds_train_miss,
                                                           num.trees = 100)
#>   missForest iteration 1 in progress...done!
#>     OOB errors MSE:             0.000743496780825689, 0.0769294112146851, 0.0985155159051316, 0.0685
#>     OOB errors NMSE:            0.00332395655622506, 0.537195051969248, 0.821679469708538, 0.6666016
#>     (weigthed) difference NMSE: 0.724955596593445
#>     time:                       12.47 seconds
#>
```

```
#>   missForest iteration 2 in progress...done!
#>     OOB errors MSE:             0.000374899545481199, 0.0637328743064622, 0.0888911676747777, 0.06064
#>     OOB errors NMSE:            0.00167606617037956, 0.445044153914882, 0.741406537292253, 0.5897427
#>     (weigthed) difference NMSE: 0.0363922446315141
#>     time:                       14.28 seconds
#>
#>   missForest iteration 3 in progress...done!
#>     OOB errors MSE:             0.000371365907017232, 0.0632891252379298, 0.0882261112208388, 0.0602
#>     OOB errors NMSE:            0.0016602683067673, 0.441945471627215, 0.735859560966976, 0.5861116
#>     (weigthed) difference NMSE: 0.00138580949967954
#>     time:                       13.75 seconds
#>
#>   missForest iteration 4 in progress...done!
#>     OOB errors MSE:             0.000377293410348567, 0.0633237952425312, 0.0882291990219114, 0.0600
#>     OOB errors NMSE:            0.00168676843974482, 0.442187570905365, 0.735885315110621, 0.5843248
#>     (weigthed) difference NMSE: 0.000250803981161146
#>     time:                       13.67 seconds
#>
#>   missForest iteration 5 in progress...done!
#>     OOB errors MSE:             0.000365854087567819, 0.0633076314814127, 0.0881345407494727, 0.0603
#>     OOB errors NMSE:            0.0016356265747947, 0.442074699997387, 0.735095807403268, 0.5873034
#>     (weigthed) difference NMSE: -0.000373738294787429
#>     time:                       15.08 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  4.29  4.3   2.69
#> 3  0.4  Ideal   I     VVS2     62    56    1240  4.74  4.77  2.95
#> 4  0.73 Ideal   F     SI1      61.7  55    3249  5.79  5.82  3.58
#> 5  0.3  Premium E     SI2      62.0  58     540  4.31  4.28  2.66
#> 6  1.01 Fair    F     VS2      64.8  57.0  4791  6.3   6.25  4.07
```

Alternatively, the `maxiter` parameter can be set to a lower number (the default is 10) or other `ranger` parameters can be adapted. Note that not all parameters to ranger function are supported. Parameters that should be adapted in function of the imputed variable (outcome for the ranger function) are not supported. Generic parameters that can be applied to all variables are supported (like: `num.trees, mtry, min.node.size, max.depth, replace,` ...), as well as `class.weights` for factor variables.

Custom initialization

The parameter `initialization` supports three values: `mean/mode`, `median/mode` and `custom`. The default is median/mode, which will initialize each variable with the median (for continuous variables) or mode (for categorical variables) calculated based on the complete observations on that variable.

When `custom` is used, a complete dataframe is expected in `x_init`. For example, the imputations of another imputation method can be used as initialization. When `custom` is used in `missForest` function, an initialization dataframe has to be passed to the `missForestPredict` function later on too.

We will exemplify this with an initialization using linear models on the `iris` dataset. Let's assume that variables `Petal.Width` and `Species` are known to be never missing. We will regress the other variables on these two variables and save the linear models to be applied on the test set afterwards.

```
data(iris)

# split train / test
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,]

# produce missing values
set.seed(2022)
iris_train_miss <- produce_NA(iris_train, proportion = c(0.2, 0.2, 0.2, 0, 0))
iris_test_miss <- produce_NA(iris_test, proportion = c(0.2, 0.2, 0.2, 0, 0))

# build linear models for Sepal.Length, Sepal.Width, Petal.Length using complete cases for each variabl
fit_1 <- lm(Sepal.Length ~ ., data = iris_train_miss[!is.na(iris_train_miss$Sepal.Length),
                                                     c("Sepal.Length", "Petal.Width", "Species")])
fit_2 <- lm(Sepal.Width ~ ., data = iris_train_miss[!is.na(iris_train_miss$Sepal.Width),
                                                    c("Sepal.Width", "Petal.Width", "Species")])
fit_3 <- lm(Petal.Length ~ ., data = iris_train_miss[!is.na(iris_train_miss$Petal.Length),
                                                     c("Petal.Length", "Petal.Width", "Species")])

# impute training with predictions of linear model
iris_train_init <- iris_train_miss
iris_train_init$Sepal.Length[is.na(iris_train_init$Sepal.Length)] <-
  predict(fit_1, iris_train_init[is.na(iris_train_init$Sepal.Length), c("Petal.Width", "Species")])
iris_train_init$Sepal.Width[is.na(iris_train_init$Sepal.Width)] <-
  predict(fit_2, iris_train_init[is.na(iris_train_init$Sepal.Width), c("Petal.Width", "Species")])
iris_train_init$Petal.Length[is.na(iris_train_init$Petal.Length)] <-
  predict(fit_3, iris_train_init[is.na(iris_train_init$Petal.Length), c("Petal.Width", "Species")])

# impute the training set using this initialization
set.seed(2022)
iris_train_imp_obj <- missForest(iris_train_miss,
                                 OOB_weights = c(1,1,1,0,0),
                                 initialization = "custom",
                                 x_init = iris_train_init)
#>   missForest iteration 1 in progress...done!
#>     OOB errors MSE:             0.163933654954688, 0.0882494745289653, 0.118292105872923, 0.03065459
#>     OOB errors NMSE:            0.252490812240121, 0.510752551784412, 0.0385064060474997, 0.05147436
#>     (weigthed) difference NMSE: 0.732750076642656
#>     time:                       0.0600000000000023 seconds
#>
#>   missForest iteration 2 in progress...done!
```

```
#>     OOB errors MSE:             0.137096628439939, 0.0832399892320051, 0.0957429358514928, 0.0323552
#>     OOB errors NMSE:            0.211156391771722, 0.481759660753553, 0.0311662078958829, 0.0543300
#>     (weigthed) difference NMSE: 0.0258891698836252
#>     time:                       0.0600000000000023 seconds
#>
#>   missForest iteration 3 in progress...done!
#>     OOB errors MSE:             0.128854552836604, 0.0801770674958631, 0.095563114533803, 0.0339263
#>     OOB errors NMSE:            0.198461937029004, 0.464032698627144, 0.0311076725217439, 0.0569681
#>     (weigthed) difference NMSE: 0.0101599840810886
#>     time:                       0.0699999999999932 seconds
#>
#>   missForest iteration 4 in progress...done!
#>     OOB errors MSE:             0.135958463726782, 0.0830331384835583, 0.0955956970689075, 0.0341691
#>     OOB errors NMSE:            0.209403389113619, 0.480562491612641, 0.0311182787774828, 0.0573759
#>     (weigthed) difference NMSE: -0.00916061710861724
#>     time:                       0.0499999999999972 seconds

# build test set initialization using the linear models learned on training
iris_test_init <- iris_test_miss
iris_test_init$Sepal.Length[is.na(iris_test_init$Sepal.Length)] <-
  predict(fit_1, iris_test_init[is.na(iris_test_init$Sepal.Length), c("Petal.Width", "Species")])
iris_test_init$Sepal.Width[is.na(iris_test_init$Sepal.Width)] <-
  predict(fit_2, iris_test_init[is.na(iris_test_init$Sepal.Width), c("Petal.Width", "Species")])
iris_test_init$Petal.Length[is.na(iris_test_init$Petal.Length)] <-
  predict(fit_3, iris_test_init[is.na(iris_test_init$Petal.Length), c("Petal.Width", "Species")])

# impute test set
iris_test_imp <- missForestPredict(iris_train_imp_obj, newdata = iris_test_miss,
                                   x_init = iris_test_init)

evaluate_imputation_error(iris_test_imp, iris_test_miss, iris_test)
#>       variable       MSE       NMSE MER
#> 1 Sepal.Length 0.1422054 0.29725212  NA
#> 2  Sepal.Width 0.1611004 0.72404672  NA
#> 3 Petal.Length 0.1107817 0.04607456  NA
#> 4  Petal.Width 0.0000000 0.00000000  NA
#> 5      Species        NA         NA   0
evaluate_imputation_error(iris_test_init, iris_test_miss, iris_test)
#>       variable       MSE       NMSE MER
#> 1 Sepal.Length 0.1362000 0.28469896  NA
#> 2  Sepal.Width 0.1331878 0.59859694  NA
#> 3 Petal.Length 0.1630601 0.06781738  NA
#> 4  Petal.Width 0.0000000 0.00000000  NA
#> 5      Species        NA         NA   0
```

Note that we chose to set the `OOB_weights` parameter so that the errors of the last 2 variables are not taken into account when calculating the convergence criteria.

Final words

The package `missForestPredict` is based on the `missForest` package (Stekhoven and Bühlmann 2012) with additional functionality for convergence criteria, initialization and imputation of new observations.

Other imputation packages based on random forests are available in the R ecosystem, using iterative algorithms, like `missRanger` (Mayer and Mayer 2019), `mice` (Van Buuren, Oudshoorn, and Jong 2007), `CALIBERrfimpute` (Shah et al. 2021), `miceRanger` (Wilson 2020). These provide various extended functionality, but they do not support the imputation of new observations. Non-iterative algorithms are also available: the function `rfImpute` in the `randomForest` (Liaw and Wiener 2002) package requires presence of the outcome at imputation time, which makes it inadequate for prediction settings; `caret` (Kuhn and others 2008) implements bagged trees without iterations and will impute new observations for most missingness patterns; `imputeMissings` implements random forests non-iteratively but will require a test set for initialization.

The R package `missForestPredict` uses an iterative algorithm and can impute a single new observation in (applied) prediction settings, even when the new observation exhibits a missingness pattern not encountered in the training set. It is therefore suitable for (applied) prediction settings.

References

Anderson, Edgar. 1935. "The Irises of the Gaspe Peninsula." *Bull. Am. Iris Soc.* 59: 2–5.

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.

Kuhn, Max, and others. 2008. "Caret Package." *Journal of Statistical Software* 28 (5): 1–26.

Liaw, Andy, and Matthew Wiener. 2002. "Classification and Regression by randomForest." *R News* 2 (3): 18–22. https://CRAN.R-project.org/doc/Rnews/.

Mayer, Michael, and Maintainer Michael Mayer. 2019. "Package 'missRanger'." *R Package*.

Shah, Anoop, Jonathan Bartlett, Harry Hemingway, Owen Nicholas, Aroon Hingorani, Maintainer Anoop Shah, and TRUE BuildVignettes. 2021. "Package 'CALIBERrfimpute'."

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

Van Buuren, Stef, CGM Oudshoorn, and Maintainer Roel de Jong. 2007. "The MICE Package." *URL Https://Www. Rdocumentation. Org/Packages/Mice/Versions/2.25*.

Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org.

Wilson, S. 2020. "miceRanger: Multiple Imputation by Chained Equations with Random Forests. R Package Version 1.3. 5."

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.