Telecom Company Client Status Missing Data

Joan Saurina

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

June 25, 2023

Contents

1 Preprocessing

1.1 Missing Data Treatment

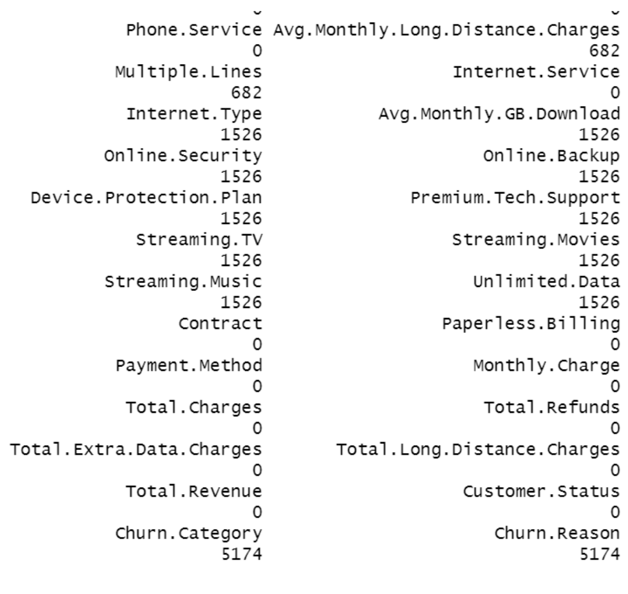
Imputation of missing values is a critical step in data preprocessing, especially in profiling and clustering processes. Missing data can introduce bias, reduce accuracy, and hinder meaningful analysis of the dataset. By imputing the missing values, we can reduce the impact of missing data and improve the quality of profiling and clustering results. Imputation methods fill in the missing values with estimated values based on the available data, which allows us to maintain the overall structure of the dataset while preserving the relationships between variables. This helps to identify patterns and relationships between variables and can provide valuable insights into the data. Overall, accurate imputation of missing values is essential for reliable and effective profiling and clustering processes.

1.1.1 Exploration and First Imputation

First, a look is taken to the total number of NAs in the dataset. *sum(is.na(data))* is performed and the following result is obtained: 26972 NA values.

Those are quite a lot, so the next step is to look deeper and try figure out the origin of these missing values.

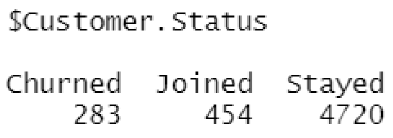
When the number of NAs for each column is checked this is the output:



```
                   ˇ                                 ˇ
       Phone.Service Avg.Monthly.Long.Distance.Charges
                   0                               682
      Multiple.Lines                  Internet.Service
                 682                                 0
       Internet.Type             Avg.Monthly.GB.Download
                1526                              1526
     Online.Security                     Online.Backup
                1526                              1526
Device.Protection.Plan            Premium.Tech.Support
                1526                              1526
        Streaming.TV                  Streaming.Movies
                1526                              1526
     Streaming.Music                    Unlimited.Data
                1526                              1526
            Contract                  Paperless.Billing
                   0                                 0
      Payment.Method                    Monthly.Charge
                   0                                 0
       Total.Charges                     Total.Refunds
                   0                                 0
Total.Extra.Data.Charges     Total.Long.Distance.Charges
                   0                                 0
       Total.Revenue                   Customer.Status
                   0                                 0
      Churn.Category                      Churn.Reason
                5174                              5174
```

Figure 1: NA count per variable on raw data.

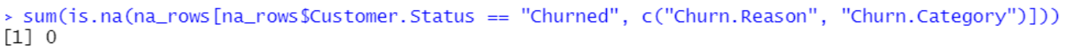
It appears that the columns which contain NAs either have 682, 1526, or 5174 missing values. This is probably not a random phenomenon. In order to look closer, an extraction of the rows with NAs is made and the number of values for each class per column is checked, and, with a little exploration, this is found:



```
$Customer.Status

Churned  Joined  Stayed
    283     454    4720
```

Figure 2: Class counts in variable Customer.Status

The sum of the values corresponding to the “Joined” and “Stayed” classes is equal to 5174, which is the exact number of NAs in the columns related to this information. It makes no sense to specify the churn category nor the churn reason if there is not a churn. To make sure if these is happening with a relation of causality we check the number of NAs in these columns for the rows which have “Churned” on this “Customer.Status” variable.

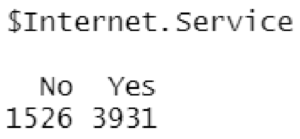


```
> sum(is.na(na_rows[na_rows$Customer.Status == "Churned", c("Churn.Reason", "Churn.Category")]))
[1] 0
```

Figure 3: Sum of the churn related NAs on rows of Churned Clients

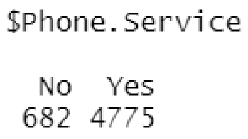
The hypothesis is refuted, and all those NA are not really unknown they are just redundant, so all of them are replace by “Stayed in the company”.

Checking further in the classes per column in the rows with NAs, the following phenomenon appears:



```
$Internet.Service

  No  Yes
1526 3931
```

Figure 4: Class counts on the variable Internet.Service



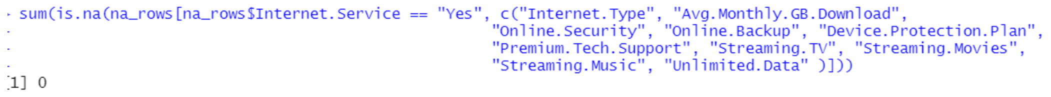
```
$Phone.Service

 No  Yes
682 4775
```

Figure 5: Class counts on the variable Phone.Service

The variable Internet Service has exactly 1526 “No” appearances, which make sense with the features with the variables with this number of NAs that can not be positive if there is no Internet Service contracted. Variables as Online.Secuirty, Internet.Type or Online.Backup are probably experiencing a similar episode as the churn related ones.

Let’s check the sum of those:

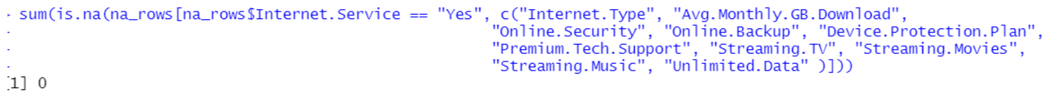
None of those variables make sense if no internet service is contracted, so, again, we have structural missing. Those can be replaced now by a “No” in the categorical features and by a 0 in the numerical ones.



```
· sum(is.na(na_rows[na_rows$Internet.Service == "Yes", c("Internet.Type", "Avg.Monthly.GB.Download",
·                                                        "Online.Security", "Online.Backup", "Device.Protection.Plan",
·                                                        "Premium.Tech.Support", "Streaming.TV", "Streaming.Movies",
·                                                        "Streaming.Music", "Unlimited.Data" )]))
:1] 0
```

Figure 6: Sum of NA values on the internet related features in the clients with internet service contracted

Furthermore, the values for the column Phone Service has exactly 682 rows with “No” as value, which again matches with 2 variables that do not make sense specifying if there is no phone service contracted. The process is repeated:



```
· sum(is.na(na_rows[na_rows$Internet.Service == "Yes", c("Internet.Type", "Avg.Monthly.GB.Download",
·                                                        "Online.Security", "Online.Backup", "Device.Protection.Plan",
·                                                        "Premium.Tech.Support", "Streaming.TV", "Streaming.Movies",
·                                                        "Streaming.Music", "Unlimited.Data" )]))
:1] 0
```

Figure 7: Sum of NA values on the phone service related features in the clients with phone service contracted

Again, it is concluded that the NAs of the numerical features must be changed for a 0, and the ones in the categorical features for “No”. When this is performed there are no missing values left in the database.

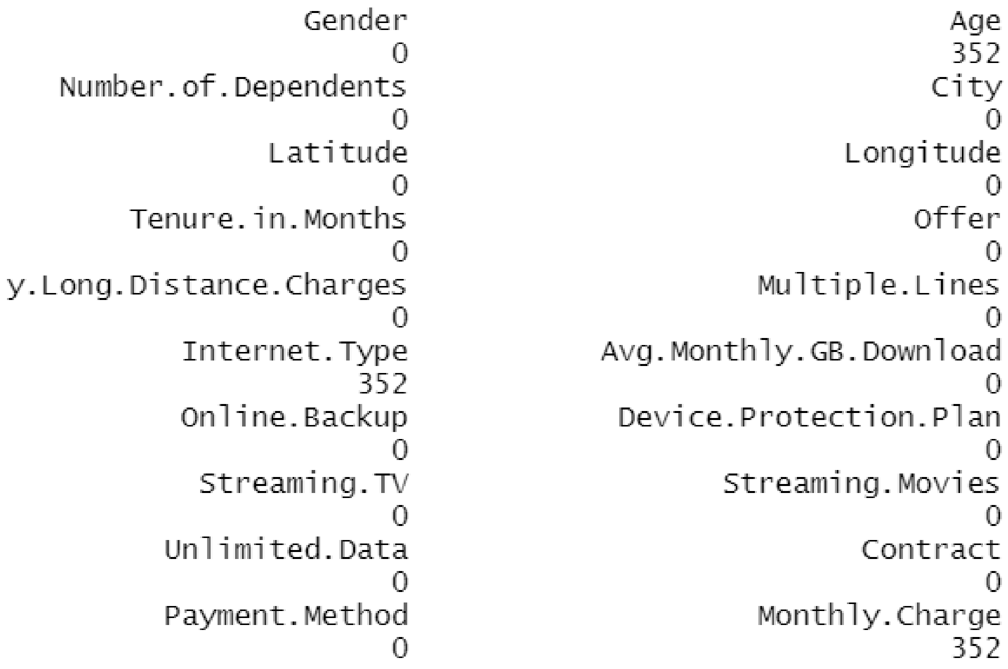
The script saves this dataset as *data_real_imputation.RData*, to be used in the future without having to impute it all again.

1.1.2 Artificial Missing Data

Since there was only structural missing data in the database our team would be missing the whole NA imputation phase, which is important for the learning process. It is needed to learn the methods designed to deal with those and it was decided to create some artificial ones in a few columns.

In order to generate missing values (NAs) in three columns of the data frame the ”missForest” package is used to create 5% missing values at random in the ”Internet.Type”, ”Monthly.Charge”, and ”Age” columns. From this moment on 352 out of 7043 rows in these three columns have NA’s.

It is a simulation to represent a scenario in which missing values are present in a dataset, which can help test how well different statistical methods and machine learning algorithms perform in handling missing data.



```
                  Gender                          Age
                       0                          352
   Number.of.Dependents                          City
                       0                            0
                Latitude                    Longitude
                       0                            0
       Tenure.in.Months                        Offer
                       0                            0
y.Long.Distance.Charges               Multiple.Lines
                       0                            0
           Internet.Type        Avg.Monthly.GB.Download
                     352                            0
           Online.Backup       Device.Protection.Plan
                       0                            0
            Streaming.TV             Streaming.Movies
                       0                            0
          Unlimited.Data                     Contract
                       0                            0
          Payment.Method               Monthly.Charge
                       0                          352
```

Figure 8: Table of the NA counts per feature once missForest is applied

1.1.3 Experiment: Imputation Methods

The availability of original data has facilitated an experiment comparing the performance of the MIMMI method and the MICE method. The utilization of the original data has enabled an accurate simulation of missing data scenarios and allowed for the evaluation of the effectiveness of each method in imputing missing values. This experiment has provided valuable insights into the strengths and limitations of each method, enabling informed decisions to be made about which method to use in future data analysis. The use of original data in this experiment has demonstrated the importance of data quality and the impact it can have on research outcomes.

MIMMI

To impute with the MIMMI method it is needed to specify the number of clusters where the dendrogram will be "cut", those resulting in the final clusters of the imputation. Since we are dealing with a huge volume of data, this number must be quite high, and it is difficult to guess the best one without any testing. This is why the method will be performed with multiple clusters and, benefiting from the fact that there is access to the true values of those NA, the error committed will be measured in order to keep the best result possible and also learn which values work best for our type and volume of data.

The error will be measured with the quadratic error for the numerical features and the sum of errors for the categorical ones. This is the result:

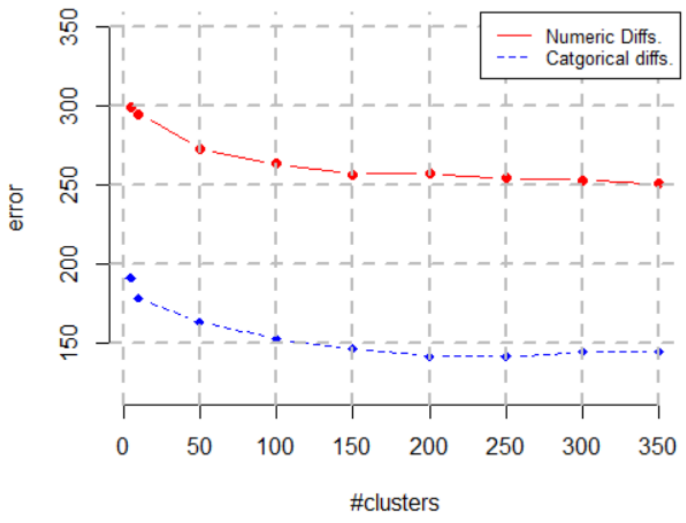


Figure 9: Categorical and numerical error along the different cluster cut with MIMMI. To be able to see both results in the same plot, the quadratic error is being squared and divided by 2.

While the categorical error seemed converged, the numerical error seems to continue decreasing as more clusters were added, although with the MIMMI function that was given, it was not possible to increase the number of clusters without getting errors. The final number of clusters decided is 350. The difference between the categorical error in this number of clusters and the lowest found is not very significant, and the numeric one seems more critical, much more if the operation realized on those values is considered.

MICE

MICE stands for ”Multiple Imputation by Chained Equations” and is a widely used approach for handling missing data in datasets. In MICE, missing values are imputed (i.e., filled in) multiple times to generate several complete datasets. Each of these datasets is then analyzed separately, and the results are combined to produce a final estimate of the effect of interest.

MICE works by assuming that the data are missing at random (MAR), meaning that the probability of missingness depends only on observed variables and not on unobserved variables. In MICE, each missing value is imputed based on its own distribution and conditional on other variables in the dataset.

First of all, in order to visually evaluate the missing data the md.pattern function shows the pattern of missingness, while the aggr_plot function creates a plot that displays the histogram of missing data, the pattern of missing data, and the labels of the columns in the subsetted data. The plot helps to identify the extent and pattern of missingness in the data, which can guide the imputation process.

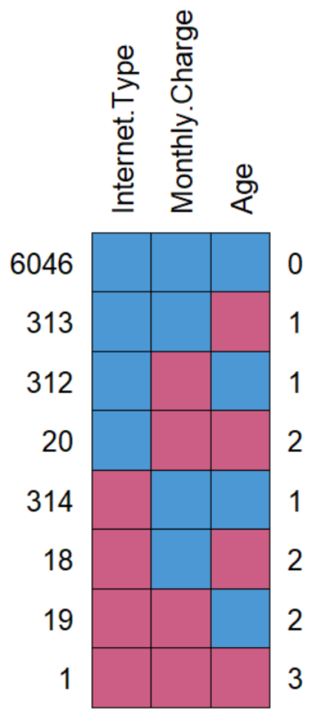


Figure 10: Intersection of the missing values on the features to be imputed.

This graphic shows how in blue if there are not missing values in the variables and in red if there are.

The first step involves creating a copy of the original dataset and initializing a list to store different combinations of imputation methods. Different imputation methods for each variable are going to be tested so the best combination can be found so a list is created to store the different combinations to be tested.

For numerical variables, the following imputation methods are used: predictive mean matching (pmm), classification and regression trees (cart), multiple imputation using distance-based imputation (midastouch), random forest (rf), and normal regression imputation (norm).

For categorical variables, the code uses the following imputation methods: pmm, cart, polynomial regression (polyreg), and linear discriminant analysis (lda). By using different imputation methods for different variables, MICE can more accurately capture the distribution of the data and provide better imputed values for the missing data.

Once this list is created, the misclassification rate and RMSE matrices are then initialized to store the evaluation metrics for different imputation methods.

The next step involves creating a loop to perform multiple imputation using the ”mice” function with different imputation methods.

Inside the loop the code is using the MICE package to impute the missing values in the dataset, and the following parameters are used:

1. **data_mice**: This is the dataset with missing values that will be imputed using MICE.
2. **m = 5**: This parameter specifies the number of imputed datasets to generate. In this case, five datasets will be generated.
3. **method = methods**[[**i**]]:This parameter specifies the imputation method to be used for each variable. The ”methods” object is a list of different imputation methods, and the [[i]] notation specifies which method to use for the current iteration of the loop.
4. **maxit = 20**: This parameter specifies the maximum number of iterations allowed for each imputation method. This is the maximum number of times the algorithm will loop over the dataset to fill in the missing values.
5. **nblocks = 38**: This parameter specifies the number of blocks used in the imputation. A block is a set of variables that are imputed together. In this case, there are 38 blocks, which means that the missing values for each variable will be imputed separately from the other variables. This can help improve the accuracy of the imputations by allowing each variable to be imputed using a method that is best suited for its own distribution.

Convergence diagnostics are then plotted for each of the imputed datasets so a visual judgment can be done. Afterward, a numerical evaluation is going to be done.

The best imputation methods for this dataset **(cart,cart)** which are later explained, draw the following convergence diagnostics:

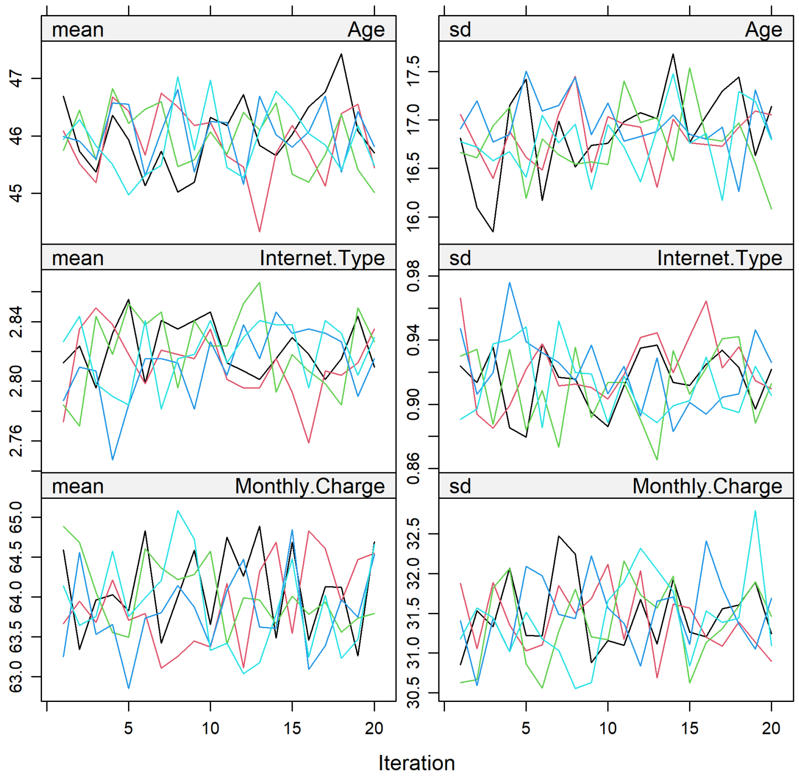


Figure 11: Convergence diagnosis of the different features

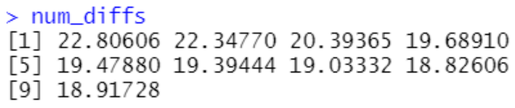
As it can be observed from the graphic, except from one imputed dataset in the variable age (light green line) , both standard deviation and mean show convergence in the 3 variables. This was not happening as well as in this combination in other ones.

The ”complete” function is used to obtain the completed dataset from each of the imputed datasets, and the misclassification rate and RMSE are calculated for the imputed values of the ”Internet.Type,” ”Age,” and ”Monthly.Charge” columns. The calculated misclassification rates and RMSEs are then stored in the matrices initialized earlier.

Later one, we are going to compare it with the MIMMI algorithm. If the results are favorable we will choose the best MICE imputation method.

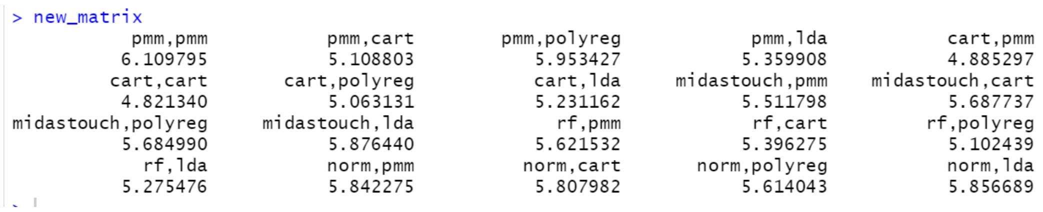
Method Comparison When checking the results of both implementations, the MICE method appears to be clearly superior to the MIMMI imputation, although is also much more computationally demanding.

This is the comparison of the standard error for both numerical features combined for each method:



```
> num_diffs
[1] 22.80606 22.34770 20.39365 19.68910
[5] 19.47880 19.39444 19.03332 18.82606
[9] 18.91728
```

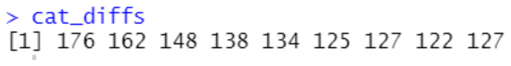
Figure 12: Standard error for the different number of clusters for MIMMI



```
> new_matrix
          pmm,pmm          pmm,cart       pmm,polyreg           pmm,lda          cart,pmm 
         6.109795          5.108803          5.953427          5.359908          4.885297 
        cart,cart      cart,polyreg          cart,lda   midastouch,pmm   midastouch,cart 
         4.821340          5.063131          5.231162          5.511798          5.687737 
midastouch,polyreg    midastouch,lda            rf,pmm           rf,cart        rf,polyreg 
         5.684990          5.876440          5.621532          5.396275          5.102439 
           rf,lda          norm,pmm         norm,cart      norm,polyreg          norm,lda 
         5.275476          5.842275          5.807982          5.614043          5.856689 
```

Figure 13: Standard error along de CV process of MICE

As for the Misclassification error:



```
> cat_diffs
[1] 176 162 148 138 134 125 127 122 127
```

Figure 14: Missclassification error for the different number of clusters for MIMMI

The difference between the methods with the categorical error appears to be less significant but MICE is better in almost all cases.

The MICE imputation result will be the one finally kept. Now it is needed to find the best combination of parameters for it.

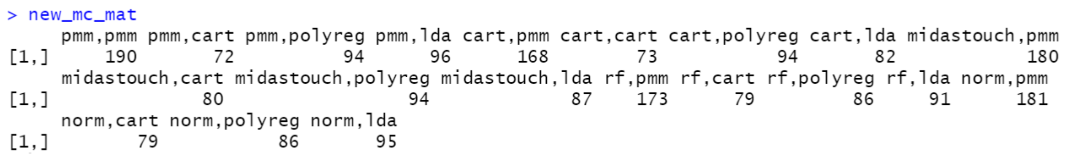
1.1.4 Final Selection

To do this a weighted normalized decision matrix is computed to determine the imputation method that yields the best overall performance based on the misclassification rate and RMSE metrics. The weights for the misclassification rate and RMSE criteria are defined as 0.5 and 0.25, respectively (there are two columns in RMSE).

The misclassification rate and RMSE matrices (previously computed) are then normalized using the ”apply” function and stored in the ”m” and ”l” variables, respectively. The ”decision_mat” variable is then computed as a weighted combination of the normalized misclassification rate and RMSE matrices using the defined weights.

Finally, the ”colnames” function is used to identify the column name (i.e., imputation method) that corresponds to the minimum value in the ”decision_mat” matrix, which represents the best overall performance.

The best imputation method results are found in cart for both numerical and categorical variables despite the fact that there are others whose performance



```
> new_mc_mat
     pmm,pmm pmm,cart pmm,polyreg pmm,lda cart,pmm cart,cart cart,polyreg cart,lda midastouch,pmm
[1,]     190       72          94      96      168        73           94       82            180
     midastouch,cart midastouch,polyreg midastouch,lda rf,pmm rf,cart rf,polyreg rf,lda norm,pmm
[1,]              80                 94             87    173      79         86     91      181
     norm,cart norm,polyreg norm,lda
[1,]        79           86       95
```

Figure 15: MIssclassification error along the CV process of MICE

is notable.

The misclassification error of cart in the categorical variables is approximately 1% and 3.35 of RMSE in the numerical variables.

1.1.5 Imputation Verification

In order to validate and ensure that the imputation is correct and goes along with the rest of the data, it is needed to compare the distributions of the features with imputed values to check if it follows a similar one.

To do this, the script reloads all the data: the original database, in which NAs are again created using the same seed to get the second database, and the final imputed database.

Here there are the compaisons between the original dataset after the structural NA imputation, the dataset with the artificial missing values and the final dataset with the MICE imputation:

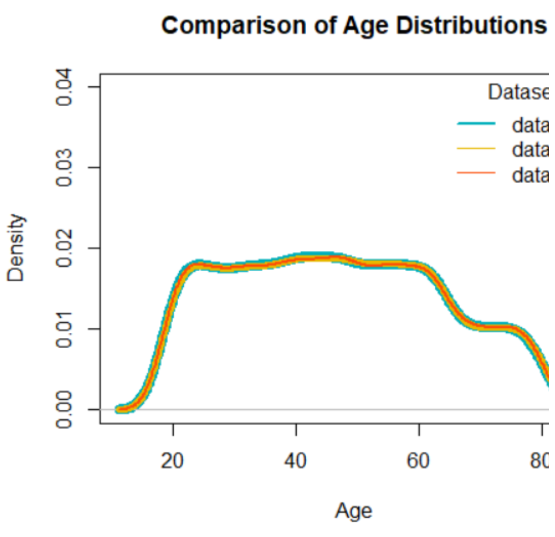


Figure 16: Density distribuition on Age on pre, during and Post imputation

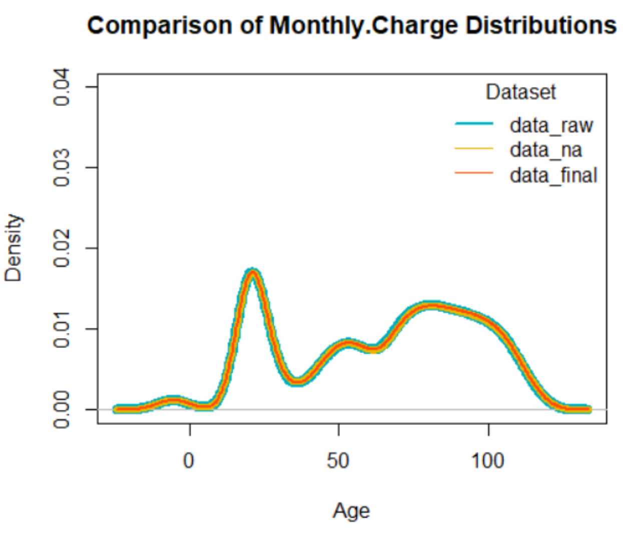


Figure 17: Density distribuition on Monthly.Charge on pre, during and Post imputation

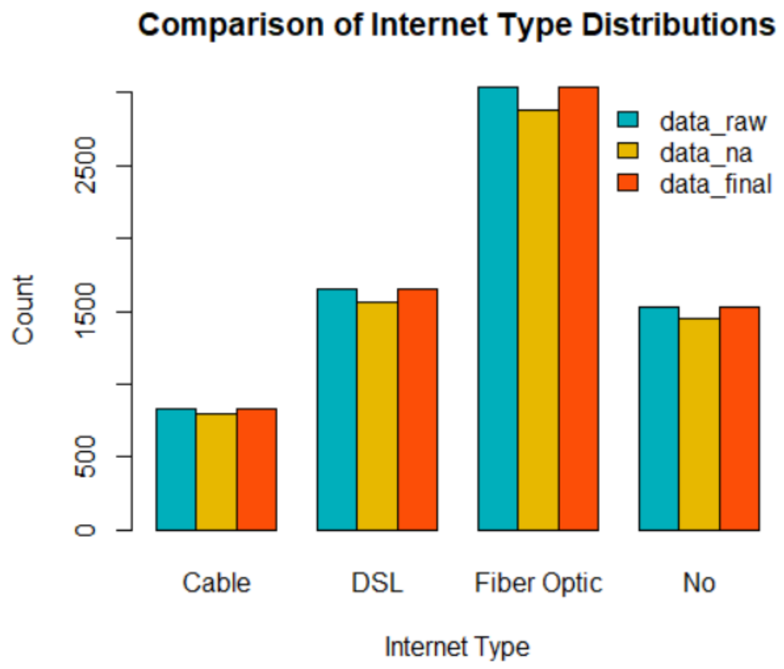


Figure 18: Class counts on Monthly.Charge on pre, during and Post imputation

The distributions are stable in all datasets. The little difference appreciated on the data_na dataset on the Internet.Type variable are due to it has less data. It is not appreciable in the other features since those plots are made density based (which does not take in account the quantity), while this last one is a count.