

Reproducibility Study: "A Framework for Analyzing the Impact of Missing Data in Predictive Models"

Adrian Schrenk
e11735055@student.tuwien.ac.at
Vienna University of Technology
Austria

Alexander Stefitz
alexander.stefitz@student.tuwien.ac.at
Vienna University of Technology
Austria

Ivona Stojanovic
e11810834@student.tuwien.ac.at
Vienna University of Technology
Austria

Abstract

The aim of this paper is to investigate the reproducibility of the paper "A Framework for Analyzing the Impact of Missing Data in Predictive Models" [1]. Due to the need of a lot of computational power and long run times, the number of considered settings was reduced. After reproducing the experiment, the outcome was compared with the original paper. We were able to replicate and confirm the main finding of the original paper, which states that a larger sample size increases the accuracy, while a higher number of variables has a negative impact. However, some other effects turned out to deviate massively from the originally presented ones.

CCS Concepts: • Information systems → Incomplete data.

Keywords: Reproducibility, Data Simulation, Missing Data, Predictive Model

1 Introduction

As part of the course "Experiment Design for Data Science", we, three students at the Technical University of Vienna studying for a Master's degree, had the task of investigating the reproducibility of the paper "A Framework for Analyzing the Impact of Missing Data in Predictive Models" [1]. To determine how reproducible the original paper is, we tried to repeat the experiment described therein and document our steps, results and encountered difficulties.

2 Outline of the original paper

The original paper takes a look at the impact of missing data in predictive models. To be exact, logistic regression models are considered. A stochastic framework is developed which yields full control of several aspects of the design matrix X and response variable y . It consists of four steps:

The first step is to generate the regressor matrix X . When doing this, the number of rows and columns is defined beforehand. This means that the sample size and number of input variables is determined. Additionally, the correlation between the columns, i.e. the variables, can be specified. Moreover, the data type of the input variables can be selected. Binary, integer and continuous values can be chosen. For creating those, an underlying distribution has to be specified: A Bernoulli distribution with success probability 0.5 produces the binary variables, a Poisson distribution with

parameter 10 provides the integers and a standard Gaussian distribution constructs the continuous ones.

The second step involves simulating the missing data. This again, is done in various different settings. First of all, the amount of missing data can be chosen. There could either be 10% or 30% of data in X missing. But also the type of missing data can be specified: A distinction is made between Missing Completely at Random (MCAR), Missing at Random (MAR) and Missing Not at Random (MNAR). When generating the missing data, a Bernoulli distribution is used. For MCAR, multiple independent Bernoulli distributions were sufficient. But for the others, a multivariate Bernoulli distribution has to be considered. Therefore, for constructing the missing values according to the MAR and MNAR concept, covariance matrices have to be defined. For MAR, they were specified in a certain way to get a correlation of once 0.5 and then 0.8. Lastly for MNAR a diagonal block matrix is used as covariance so that each block was equal in size and one yields 0.5 and the other 0.8 correlation.

The third step is to simulate the target variable y and fit it with a logistic regression model afterwards. The construction of y itself is also based on a logistic regression model and tuned in a way to get 90% accuracy. The predictive power of the input variables can be specified: Since the creation of y is done using a regression, an estimator β has to be computed. This is done with a geometric distribution where its parameter takes one of the values 0.2, 0.5 and 0.8. In addition, the geometric distribution is multiplied with a parameter, the so-called effect size, which allows control of the resulting accuracy achieved by the logistic regression with X . After having constructed all necessary data, the logistic regression model regressing y on X is fitted. Then, the accuracy of the model fit is computed. For each of the simulation scenarios, 150 data sets are fitted: First X is generated 10 times for each chosen setting from the first step. After that, y is simulated 15 times for every regressor matrix X as described in the third step.

As the fourth and final step, the impact of these factors is then judged based on the influence they all have on the accuracy of the logistic regression model. To quantify this influence of the varying settings and make them comparable, a linear regression model regressing the accuracy of the logistic regression on the modifiable aspects is built.

The main results of this analysis are that a larger sample size has a positive effect on accuracy and a higher number of variables a negative one.

3 Our approach

As a first step to reproduce the results, we were investigating which resources were available and what had to be done by ourselves. The original authors of the paper thankfully published parts of the code on GitHub¹, which was also noted in the original paper [1]. The GitHub repository is quite small and contains only three files, one of which (README.md) contains a few lines of text on how the program code is split among the two other files. The code of this paper is written in the statistical programming language R.

The file `effect_size.R` loads the packages which are required to run the code and contains different functions that are needed to calculate the effect size (see section 2). In this file it is also defined, which combinations of features will be considered. This is done by creating one vector for each feature with different settings and then crossing them, to create one big data frame. There each row represents one unique combination of the feature levels and each combination is present exactly once. After this, the calculation of the effect sizes for each of the combinations is performed and written to the file `teste.txt`.

The last file is called `simulation.R` and as the name implies, here the actual simulation is done. The file again starts with importing some necessary packages, then functions are defined. Some of these functions are the identical to the ones in the previous file. At the end of the file, the simulation is performed. We noticed that this file does not contain any code to read in the file `teste.txt`. Instead, there is a variable `experimentos` which never gets introduced or declared. It seems like this variable should contain a data frame with the values of `teste.txt` with a different column order, but it's not quite clear at this moment.

Also interesting to note is that the authors of the original paper decided to use the function `foreach()` from the package `doParallel` to perform the calculations on multiple CPU cores at once, since R usually runs on only one CPU core, according to [2]. This already gave us the impression that the task we want to reproduce is quite computationally laborious.

3.1 Idea

To approach the problem in a structured way, we decided to create a R Markdown file in which we copied each of the functions. This made it easier to run different components for debugging purposes and gave us more possibilities to add comments.

Before being able to run the given code, we have to modify or expand it by creating a connection between the two

given files. This was easily achieved by reading in the file `teste.txt` with a built-in function. We decided to store the newly created data frame in the variable `experimentos`, so we had to change as few of the original code as possible. It resulted in the following line of code:

```
experimentos
  <- read.csv("teste.txt", sep="\t",
              header=FALSE)

names(experimentos)
  <- c("id", "acc", "n", "j",
        "i", "dist", "p", "ef")
```

The file does not contain a header, so we have to set this parameter and give column names manually. The names have the following meaning, for which we oriented ourselves towards the names used in the code.

- `id` is a technical column which is created when writing the file, it is not needed in our process.
- `acc` stands for the desired accuracy, in this framework it's always 0.9.
- `n` represents the number of observations.
- `j` describes the number of variables.
- `i` stands for the predictive power of the input variables.
- `dist` represents the used distribution.
- `p` is the correlation between the variables
- `ef` stands for the effect size.

While all of the functions required for the simulation seemed to be available and working without any problems, there were some changes needed in the function call of the simulation. It turned out that the file `experimentos` should have a different and unclear column order, and the columns are being accessed by column numbers, not by name. Since we thought that this is a more intuitive and less error-prone way, we changed the code as follows to access the columns by name:

```
solve_simulation(
  j = experimentos[w, "j"],
  n = experimentos[w, "n"],
  dist = experimentos[w, "dist"],
  p = experimentos[w, "p"],
  i = experimentos[w, "i"],
  ef = experimentos[w, "ef"])
```

Another small change in the original code was necessary: In the code from the original paper, the results (achieved accuracy) from each simulation scenario are saved in an `.RData` file locally on the computer. However, we tried this with some test data and despite the program finishing, no file was saved. Therefore we changed this line so that the results get saved as an `.csv` file:

¹https://github.com/fsantore/missing_data_analysis

```
name <- paste0("j=", j, "-n=", n,
               "-", dist, "-rho=", p,
               "-i=", i,
               "-ef=", ef, ".csv")
write.csv(resultsFinal, file = name)
```

Now all preparations were done to start trying to reproduce the results of the original paper. We planned to keep all values of the features as they were, and run the experiments locally on one of our available computers. Available computers were:

1. Lenovo ThinkPad X390, Intel i5-8265U, 16GB RAM, Windows 11
2. Apple MacBook Pro (13-inch, 2019), Intel i5-8279U, 8GB RAM, macOS 12
3. Lenovo ThinkPad E595, AMD Ryzen 7 3700U, 16GB RAM, Windows 10

On all of the computers R version 4.1.2 was installed, the required packages had the following versions: SimCorMultRes 1.8.0, caret 6.0-90, bindata 0.9-20 and tidyverse 1.3.1.

We planned on using device 1 as our main device to perform the experiments and switch to the other ones if more computational power is needed. We also created a Google Colab Free account and a shared notebook to be able to run code online, if needed.

The gained results would then be used to create a linear regression model which uses all framework features to predict the achieved accuracy. Since all features only take at most three different values and to make the coefficients comparable with the ones of the original paper, we had to convert them to be factor variables.

The code which creates the linear model then looks as follows, where the variable `df` is a data frame containing all results. This was created by appending all `.csv` files described above. There are two further things to note: Firstly, one notices that in this code, different names are used for the variables. This also originates from the original paper, although being confusing, it assigned the variables correctly and we decided not to change it to stay as close to the original workflow as possible. Secondly, we multiply the calculated accuracy by 100 so that the obtained coefficients have the same magnitude as in the original paper. This procedure was not documented directly in the original paper, but the results presented left no doubt that this step had been done.

```
df$Accuracy <- df$Accuracy * 100
model <- lm(Accuracy~col+row+
            corr+power+
            distribution+
            type+amount,
            data=df)
```

3.2 Running the code and encountered difficulties

After having done all necessary preparations to reproduce the original paper [1] in [subsection 3.1](#), we started running the code. Like explained in [section 2](#) and [3](#), we first had to calculate the effect size. Since we tried to reproduce all given combinations of the variables `j`, `n`, `dist`, `rho`, `i` and every one of these variables takes 3 different values, the effect size has to be calculated for $3^5 = 243$ different settings.

This is the first time problems with the run time of the code occurred. Since the authors of the original paper did not make any statement on how they performed their calculations, we assumed that they could be done in a reasonable amount of time without any issues. After letting R run for a few hours, we checked the progress: The scenarios with a small number of observations and variables had been finished relatively fast, but those with a higher number have not been finished yet.

We decided to let it run in the background for several hours and if finished after more than 36 hours. Now we had the required effect size values to move on to the main part of the study, trying to measure the impact of the missing data.

Using the code we wrote and presented in [subsection 3.1](#), we read in the effect sizes and started the simulation study. After being surprised by the scope of the run time for calculating the effect size, we were expecting that this would not be finished in only a few hours. However, it was more severe than expected: In the given setting, one simulation of the smallest setting ($n = 500$, $j = 10$) took about 2 hours, while any increase, no matter if we increased the number of observations or the number of variables, made it impossible to finish the simulation in less than 12 hours. After this period, we interrupted R because even if it would have finished closely afterwards, this had to be repeated several times and the time for this reproducibility study was limited.

As a first step to bring the scope of this experiment to a reasonable size, we decided to reduce the number of iterations: In the original paper, like explained in [section 2](#), for every setting of features, the input variables are created 10 times independently, and for each of those variables, the target variable gets simulated 15 times. So in total, we simulate each scenario 150 times.

In order to reproduce the results, we had to massively reduce this number: We reduced both numbers to 2, so we repeat the simulation $2 \cdot 2 = 4$ instead of 150 times. Although this reduction seems very severe, this seemed like the only possibility to simulate also scenarios with larger variable and observation sizes.

We also asked ourselves whether it would be possible to use different simulation numbers for each of the settings, but discarded this idea because it could introduce a bias through an uneven sample size.

When testing these changes, it turned out that our approach was successful: We were able to finish the simulation

Feature	Our framework	Original paper
Observations n	200, 1000	500, 1000, 10000
Variables j	10, 50	10, 50, 200
Predictive power i	0.2, 0.5, 0.8	0.2, 0.5, 0.8
Correlation between variables p	0, 0.5, 0.8	0, 0.5, 0.8
Distribution dist	"normal", "binomial", "poisson"	"normal", "binomial", "poisson"

Table 1. Simulation Settings

with 1000 observations and 50 variables, and therefore also all smaller settings, in less than 12 hours each.

3.3 Actual Realization

In [subsection 3.2](#) we explained our process how we found out which simulations are feasible for us, in this section the final choice of considered feature values will be presented. Like already explained in the previous subsection, through reducing the number of repetitions, we are able to simulate relatively large models.

[Table 1](#) shows the considered values in our framework, compared with the original paper. Regarding the predictive power, the correlation and the distribution we decided to keep all levels identical to the original paper, since the values of these features do not have an impact on the run time of the according scenario. Regarding the number of variables, we decided to drop the highest level (200 variables) due to the long run time, but keep the other two levels (10 and 50 variables) identical to the original paper.

Regarding the observations, we decided to perform a more drastic change: Beside dropping the highest level of 10000 observations, we decreased the smallest level of observations from 500 to 200. This is necessary to achieve a larger difference between the two considered levels. In the original paper, the main comparison was done between the smallest and the largest level, but this is not possible for us. Since the long run time defines the level of 1000 observations as upper bound, we had to decrease the value of the lower level. We have chosen the value 200 so that we have the same ratio between the observation levels as for the variable levels (where we had 10 and 50).

We also decided against using Google Colab because the free version has some quite significant restrictions: One can not run code in the background, and the execution gets interrupted after a few hours. Especially for the larger scenarios, it often terminated before finishing.

It also turned out that the device 2 (MacBook Pro) was able to perform the simulations significantly faster than the other devices. We think this is because the packages used to run R multi-threaded are optimized for Unix-based operating systems. Therefore, we simulated the largest scenarios on device 2 and the others on devices 1 and 3. Of course we performed some tests if the devices behave in the same way, which was the case.

4 Results

After conducting the above described experiment, we finally had the values we needed to compare them to the results from the paper. Based on the differences we can then judge how reproducible the study from the paper is.

[Figure 1](#) is made in the same way as the plot in the original paper: We show the first and third quantile of all accuracies we get from fitting the logistic regression models per sample size, number and type of variables, predictive power and missing data mechanism. Just as in the paper, we also only show the cases where the correlation between the input variables is 0.8. We presented the missing data mechanism MAR with correlation 0.5 and only considered the settings where we simulated 30% of missing data likewise.

Except for the MAR case, we can confirm that the resulting accuracies seem to be less affected by missing data, if the sample size is higher. The accuracies also get more precise for a higher number of observations as described in the paper. This can be seen in the narrower bars. We can also support the thesis that a higher number of variables yields a decrease in accuracy levels.

We can also see that the combination of more variables and less observations is the worst case scenario, but for us the underlying missing data mechanism doesn't seem to be as determining in this case for the impact on the accuracy as it was in the paper. It looks like all of the accuracies are affected more or less in the same way regardless of this mechanism. When it comes to data types and the predictive power of the input variables, the outcomes also start to differ.

While there is a clear trend of achieving the best accuracies with continuous variables (Gaussian) and the worst accuracies with binary ones (Bernoulli) in the paper, we cannot clearly derive such a trend with our results. In our case, the data type doesn't seem to play a big role. Furthermore, the predictive power of the input variable appeared to be a quite determining factor in the paper. We on the other hand, cannot attribute a definite influence of this feature. This of course could stem from the fact that we reduced the number of repetitions and therefore also the amount of accuracies plotted considerably.

The biggest visible difference between our outcome and the one from the paper would be the accuracies in the MAR case. For us, the accuracies were always on the lower end for this missing data mechanism regardless of the setting. Only

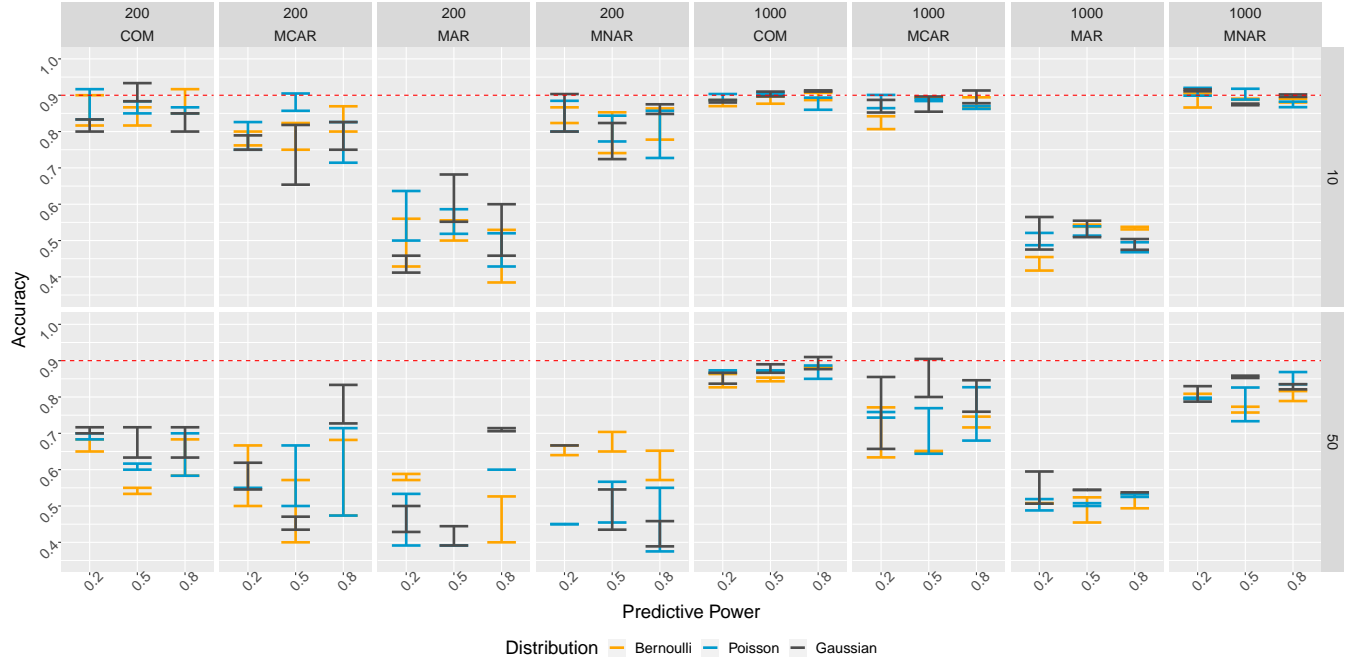


Figure 1. Accuracy quantile intervals by sample size (x-axis top), missing data mechanism (x-axis top), predictive power (x-axis bottom), number of variables (y-axis left) and used data type (color of bars)

the narrower bars, i.e. the more precision in accuracies, for a higher sample size could be reproduced.

Table 2 shows the estimated linear regression coefficients we received in our framework and the ones from the original paper. Before evaluating the results, we have to note again that due to the decrease of one feature level, some coefficients may not be comparable in size. These are marked with a star. However, we can still compare the sign and therefore the

effect of the variable change on the accuracy. Regarding the naming conventions in the table we again oriented ourselves on the original paper.

First we take a look at the effect of an increase of the number of observations (in Table 2 denoted by SS) on the accuracy: Like in the original paper, the coefficient is positive. In our case, this means that by increasing the number of observations from 200 to 1000, the accuracy increases by 8.3%.

Also regarding the number of variables in the model (denoted by NINPUT) we can confirm the statement of the original paper: By increasing the number of variables, the accuracy decreases. However, according to our results, the effect is three times larger than described in the original paper.

The original paper claimed that the correlation of the input variables (CINPUT) and the predictive power (PPINPUT) have no significant effect onto the accuracy according to the linear regression. We can confirm this thesis, also all of our obtained coefficients are very close to 0. We can also confirm the effect of the proportion of missing data (PMD): Our coefficient is very close to the one given in the paper.

However, not all statements could be confirmed: In the original paper, there was a significant difference in accuracy when using different types of input data (TINPUT). In our results however, there is no significant difference between

Parameter	Reproduced	Original
Intercept*	85.88	79.61
SS_1000*	8.30	6.31
SS_10000*	NA	14.89
NINPUT_50	-12.41	-4.52
NINPUT_200	NA	-19.53
CINPUT_0.5	0.50	-0.08
CINPUT_0.8	-0.46	-0.23
TINPUT_Poisson	-0.91	4.06
TINPUT_Gauss	-0.11	8.50
PPINPUT_0.5	-0.44	-0.07
PPINPUT_0.8	-0.10	0.36
TMD_MCAR	-3.93	-1.46
TMD_MAR	-35.40	-4.07
TMD_MNAR	-3.92	-4.67
PMD_0.9	2.48	2.27

* Not directly comparable

Table 2. Comparison of linear regression coefficients

the variable types as demonstrated in Figure 1 as well: All coefficients are relatively close to 0, and the difference between the three types is reversed.

Also regarding the type of missing data (TMD) some interesting differences can be noted: While the basic statement that missing data scores worse than complete data can be confirmed, the impact of data which was missing at random (MAR) was measured to be more than 8 times as big as in the original paper. This can also be seen in Figure 1, especially for the larger observation number. This huge difference was surprising at first, but multiple repetitions and also checking the code for mistakes confirmed this outcome.

5 Conclusion and Discussion

Our reproducibility study gives a mixed picture: Through providing the code, the authors laid a good foundation for basic reproducibility. However, to reproduce all of the original settings a massive amount of computational power is needed: Despite reducing the workload massively, there was still a total run time of a few days. Therefore, it can be assumed that the original simulations have been done on a computer of much larger scale, for example a powerful cloud hosted server or a machine at university. However, nothing about their setup was mentioned in the original paper.

The authors of the original paper were also very inconsistent with the naming of variables and parameters. For example, the number of observations was denoted by at least

three different names: `n`, `row`, `SS`. This did not prevent us from reproducing the paper, but made the procedure harder than necessary.

Regarding the results, we were able to reproduce the main statements regarding the impact of the number of observations and variable on the accuracy, but not everything could be confirmed. Especially the measured effects of the input variable type and the type of missing data deviated massively from the results presented in [1].

For potential future reproducibility studies on this paper, we suggest to make sure that sufficient computational power is available.

References

- [1] Fabiola Santore, Eduardo C. de Almeida, Wagner H. Bonat, Eduardo H. M. Pena, and Luiz Eduardo S. de Oliveira. A framework for analyzing the impact of missing data in predictive models. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management, CIKM '20*, page 2209–2212, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450368599. doi: 10.1145/3340531.3412129. URL <https://doi.org/10.1145/3340531.3412129>.
- [2] Aloysius Lim and William Tjhi. R high performance programming, Jan 2015. URL <https://subscription.packtpub.com/book/application-development/9781783989263/1/ch01lvl1sec10/r-is-single-threaded>.

A Digital Object Identifier

The DOI of this reproducibility study is <http://doi.org/10.5281/zenodo.5920221>.