Handling Outliers and Missing Data in Regression Models Using R: Simulation Examples

Mohamed Reda Abonazel
Department of Applied Statistics and Econometrics, Faculty of Graduate Studies for Statistical Research, Cairo University, Egypt
Email: mabonazel@hotmail.com

Abstract
This paper has reviewed two important problems in regression analysis (outliers and missing data), as well as some handling methods for these problems. Moreover, two applications have been introduced to understand and study these methods by R-codes. Practical evidence was provided to researchers to deal with those problems in regression modeling with R. Finally, we created a Monte Carlo simulation study to compare different handling methods of missing data in the regression model. Simulation results indicate that, under our simulation factors, the k-nearest neighbors method is the best method to estimate the missing values in regression models.

Keywords: Missing data; Monte Carlo simulation; Multiple imputation methods; R-software; Robust regression estimators.

1. Introduction
Consider the following linear regression model:

\[ Y = X\beta + u, \]

(1)

where \( Y \) is a \( n \times 1 \) vector of dependent variables, \( \beta \) is a \( p \times 1 \) unknown parameters vector, \( X \) is a \( n \times p \) regression matrix, and \( u \) is a \( n \times 1 \) error vector. The classical assumptions for this model are:

A1: \( u \sim N(0, \sigma^2 I_n) \).
A2: \( X \) is non-stochastic matrix.
A3: \( X \) is full column rank matrix, i.e., \( \text{rank}(X) = p \).

The formula of OLS estimator of the model in Eq. (1) is:

\[ \hat{\beta}_{OLS} = (X'X)^{-1}(X'Y). \]

The OLS estimation is highly sensitive to outliers and missing values in dataset. So many studies provided different methods to handle these problems to get more efficient estimation of \( \beta \).

In this paper, we will review the basics of robust estimators of regression models when the dataset contains outliers, and the common methods to handle the missing data in regression models. Moreover, we provide R-codes to handling these problems in the dataset (outliers and missing data problems). Also, we will investigate the efficiency of some methods to handle the missing data in the regression by conducting simulation study.

The rest of the paper is organized as follows: Section 2 provides the background and the basics of the robust regression. Section 3 presents some different methods to handle the missing data in regression models. Section 4 presents two applications using R-codes. Section 5 displays the Monte Carlo simulation study. Section 6 involves the concluding remarks.

2. Robust Regression Estimators
There are two categories of outliers; first the outliers in \( Y \)-dimension (response variable), second the outliers in \( X \)-dimension (explanatory variable).

---

1 The content of this paper was presented in a workshop entitled “Advanced statistical techniques using R: Outliers and missing data” in annual conference on statistics, computer sciences and operations research, in Cairo University, Egypt, vol. 54. 2019. See Abonazel [1].
Detecting or diagnosing outliers is a very important process in regression analysis, so some methods concerning the detection of outliers will be illustrated, and are statistics that focus attention on observations having an influence on OLS estimator, see Barnett and Lewis [2]. Robust estimation provides an alternative to the OLS estimation when classical assumptions are unfulfilled, see Alma [3].

Generally, the goal of robust regression is to develop methods that are resistant to the possibility that one or several unknown outliers may occur anywhere in the data. Robust regression can be used in any situation where OLS regression can be applied. It generally gives better accuracies over OLS because it uses a weighting mechanism to weigh down the influential observations. It is particularly resourceful when there are no compelling reasons to exclude outliers in the dataset.

Robust estimator term refers to the estimator that can protect it against the violations of the classical linear regression model assumptions, see Andersen [4], Gervini and Yohai [5], and Abonazel and Rabie [6].

Robust estimation method is designed to circumvent some limitations of traditional parametric and non-parametric estimation methods in case of outliers in the data. So the robust methods are resistant to the influence of outliers. Therefore, the robust estimation method is discussed by many papers in several regression models other than linear model, such as count regression model [7], semiparametric partially linear model [8, 9], and other. In the following, we will review some of these methods.

2.1. M-Estimation

Huber [10] introduced the M-estimation method which is now the most common robust regression method, see Fox and Robust [11]. The M-estimation method is a generalization to maximum likelihood estimation in context of location models. That is nearly as efficient as OLS. Rather than minimizing the sum of squared errors, as the objective, M-estimation method principle is minimizing the residual function, see Huber [12]. The likelihood function for \( \beta \) and \( \sigma \) is

\[
L(\beta, \sigma) = \frac{1}{\sigma^n} \prod_{i=1}^{n} f \left( \frac{y_i - x_i^T \beta}{\sigma} \right),
\]

where \( x_i = \left(1, x_{i1}, ..., x_{ip-1}\right)^T \). By replacing the OLS criterion with a robust criterion, M-estimator of \( \beta \) is

\[
\hat{\beta}_M = \min_\beta \sum_{i=1}^{n} \rho \left( \frac{y_i - x_i^T \beta}{\sigma} \right), \quad \hat{\sigma}_M = \frac{\text{median} |e_i|}{0.6745},
\]

where \( e_i \) denotes the \( i \)th residual. We obtain the following normal equations:

\[
\sum_{i=1}^{n} x_{ij} \psi \left( \frac{y_i - x_i^T \hat{\beta}_M}{\hat{\sigma}_M} \right) = 0; \text{ for } j = 0, 1, ..., p - 1,
\]

where \( x_{ij} = 0 \) and \( \psi(\cdot) \) is the first derivative function of \( \rho(\cdot) \) and is called the influence function. Iteratively reweighted least squares (IRLS) method used to solve the M-estimates nonlinear normal equations. The following iterative algorithm summarizes this (see Ruckstuhl [13]):

1. Start with the OLS estimate as an initial estimate of \( \hat{\beta}_M \) and then estimate \( \hat{\sigma}_M \).
2. Calculate the weights, say \( w_i \).
3. Calculate a new estimate of \( \hat{\beta} \) using Eq. (3).
4. Repeat steps 2 and 3 until the algorithm converges.

In the end, the formula of M-estimator is

\[
\hat{\beta}_M = (X'WX)^{-1}X'WY; W = diag(w_i)
\]

Table 1 displays the different objective and weight functions which common used in robust regression.

<table>
<thead>
<tr>
<th>Method</th>
<th>Objective Function</th>
<th>Weight Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>OLS</td>
<td>( u^2 )</td>
<td>1</td>
</tr>
<tr>
<td>Huber</td>
<td>( \begin{cases} \frac{1}{2}u^2 &amp; \text{for }</td>
<td>u</td>
</tr>
<tr>
<td>Bisquare</td>
<td>( \begin{cases} \frac{1}{6}k^2 \left( 1 - \left[ \frac{</td>
<td>u</td>
</tr>
</tbody>
</table>

Source: Fox and Weisberg [14]

2.2. S-Estimation

The S-estimator (“S” for “scale statistic”) is a member of the class of high breakdown-point (BDP) estimators introduced by Rousseeuw and Yohai [15].

S-estimation is based on residual scale of M-estimation. The weakness of M-estimation method is the lack of consideration on the data distribution and not a function of the overall data because only using the median as the weighted value.

S-estimation uses the residual standard deviation to overcome the weaknesses of median; the S-estimator is defined by \( \hat{\beta}_S = \min_\beta \hat{\sigma}(e_1, ..., e_n) \), with determining minimum robust scale estimator \( \hat{\sigma}_S \). We obtain the estimating equations for S-estimator:
\[
\sum_{i=1}^{n} x_i \psi \left( \frac{y_i - \hat{\mu}}{\hat{\sigma}} \right) = 0.
\] 

Pitselis [16], showed that S-estimator is more robustly than the M-estimator.

### 2.3. MM-Estimation

Yohai [17], introduced another robust estimation which has high BDP and high efficiency is MM-estimation, by combining S-estimation with M-estimation. Also, Yohai [17] showed that MM-estimators are highly efficient, and not sensitive to leverage points compared to an M-estimators. Recently, Almetwally and Almongy [18] studied the efficiency of some robust estimators by a simulation study, and they conclude that the best robust estimator is MM-estimator.

### 3. Missing Data in Regression Models

The missing data is a common and important topic in statistics. There are many methods proposed to handle the missing data. But before jumping to these methods, we have to understand the reason why data goes missing.

#### 3.1. Missing Data Types (Mechanisms)

It is helpful to know why they are missing. There are three general missingness mechanisms, moving from the simplest to the most general (see Rubin [19]):

- **3.1.1. Missing Completely at Random (MCAR)**
  - When the missing data are independent both of observable data and of unobservable data

- **3.1.2. Missing at Random (MAR)**
  - When the missing data are not related to the missing data, but it is related to some of the observed data

- **3.1.3. Missing not at Random (MNAR)**
  - When the missing data are related to the reason it's missing. MNAR is called “non-ignorable” because the missing data mechanism itself has to be modeled as you deal with the missing data. You have to include a model for why the data are missing.

#### 3.2. Missing Data Patterns

The missingness pattern is very important because it affects the choice of how to deal with missing values, see Van Buuren [20]. Figure 1 shows various data patterns in multivariate data.

#### 3.3. Handling Missing Data

Note that the methods for handling missing data differ depending on the type of data (variable), and therefore we cannot use any of them for any data. Many references discuss these methods such as Carpenter and Kenward [21], Berglund and Heeringa [22], Raghunathan, et al. [23], El-Sheikh, et al. [24], and Abonazel and Ibrahim [25]. Figure 2 summarizes some of the methods for handling missing data.

![Figure 1: Some missing data patterns; Blue is observed and red is missing](Source: Van Buuren [20])

<table>
<thead>
<tr>
<th>Univariate</th>
<th>Monotone</th>
<th>File matching</th>
<th>General</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="Image" alt="Univariate Pattern" /></td>
<td><img src="Image" alt="Monotone Pattern" /></td>
<td><img src="Image" alt="File matching Pattern" /></td>
<td><img src="Image" alt="General Pattern" /></td>
</tr>
</tbody>
</table>
4. R-Applications

In this section, we will provide two applications using R-codes. The first application displays full steps of the regression analysis when the dataset includes outliers. Similarly, the second application displays different methods to estimate the missing values and make a comparison study between these methods and then select the best estimation method of them. We can consider that these applications are practical guides for researchers to handle these problems (outliers and missing data) in regression using R.

4.1. Application I: Robust Estimators in R

*rlm()* function (in MASS package) is the main function of robust regression. In *rlm()* function, the outliers can be weighted based on three weight functions: *psi.huber*, *psi.hampel*, and *psi.bisquare*, specified by the *psi* argument.

```r
#----- Prepare the R console
rm(list = ls(all = TRUE)) # Remove all objects in R console
set.seed(09061982)# Set the seed for reproducible results
n =100
out.per=.20
#======================
### generate the model
#======================
x=rnorm(n)
error1=rnorm(n- (n*out.per))
error2=rnorm(n*out.per.max(error1)*8,1)
error=sample(c(error1,error2))
#error=c(error1,error2)
y=1+3*x+error
#----------------------------------
# saving the dataset
#---------------------
#dataset_outliers=data.frame(y,x)
#write.csv(dataset_outliers,"dataset_outliers.csv")
#----------------------------------
# 1- OLS Estimation
#---------------------
ols=lm(y~x)
summary(ols)
```

Figure-2. Some methods for handling missing data
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##    Min     1Q Median     3Q    Max
##-7.386 -3.697 -3.027 -1.619 16.148
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.2230     0.6767   6.241  1.11e-08 ***
##            3.2985     0.6632   4.973  2.80e-06 ***
## ---
## Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
##
## Residual standard error: 6.766 on 98 degrees of freedom
## Multiple R-squared:  0.2015, Adjusted R-squared:  0.1934
## F-statistic: 24.74 on 1 and 98 DF,  p-value: 2.804e-06

windows()
boxplot(y)

plot(y~x)
abline(ols)

# Four Diagnostics
### a. Normality of Residuals

```r
windows()
par(mfcol=c(2,2))
plot(ols)
```

![Residuals vs Fitted](image1)
![Scale-Location](image2)
![Normal Q-Q](image3)
![Residuals vs Leverage](image4)

```r
shapiro.test(ols$residuals)
## Shapiro-Wilk normality test
```

```
## data:  ols$residuals
## W = 0.64596, p-value = 3.51e-14
```

### b. Linearity test

```r
library(lmtest)
harvtest(ols)
```

```
## Harvey-Collier test
```

```
## data:  ols
## HC = 0.16906, df = 97, p-value = 0.8661
```

### c. Heteroskedastic test

```r
library(lmtest)
gqtest(ols)
```

```
## perform Goldfeld-Quandt test
```

```
## data:  ols
## GQ = 1.3782, df1 = 48, df2 = 48, p-value = 0.135
```

## alternative hypothesis: variance increases from segment 1 to 2
#=========================  
## 2 - Robust estimators
#=========================

library(MASS)
Robust_1 <- rlm(y ~ x, method="M",psi = psi.bisquare)
summary(Robust_1)
## Call: rlm(formula = y ~ x, psi = psi.bisquare, method = "M")
## Residuals:
##       Min     1Q   Median     3Q    Max
## -4.0599 -0.4089  0.1719  1.6026 19.4509
##
## Coefficients:
##      Value   Std. Error     t value
## (Intercept)  0.9508  0.1112     8.5472
##         x    2.9964  0.1090    27.4803
##
## Residual standard error: 1.254 on 98 degrees of freedom

windows(500,500)
plot(y~x)
abline(ols)
abline(Robust_1, col=2)

Robust_2 <- rlm(y ~ x, psi= psi.huber)
summary(Robust_2)
## Call: rlm(formula = y ~ x, psi = psi.huber)
## Residuals:
##       Min     1Q   Median     3Q    Max
## -4.5693 -0.9175 -0.3343  1.1055 18.9445
##
## Coefficients:
##      Value   Std. Error     t value
## (Intercept)  1.4536  0.1689     8.6053
##         x    3.0338  0.1656    18.3246
##
## Residual standard error: 1.399 on 98 degrees of freedom

abline(Robust_2, col=3)

Robust_S<-lqs(y ~ x,method ="S")
abline(Robust_S, col=4)

Robust_MM <- rlm(y ~ x, method = "MM",psi= psi.huber)
summary(Robust_MM)
## Call: rlm(formula = y ~ x, psi = psi.huber, method = "MM")
## Residuals:
##       Min     1Q   Median     3Q    Max
## -4.0575 -0.4068  0.1736  1.6043 19.4532
##
## Coefficients:
##      Value   Std. Error     t value
## (Intercept)  0.9487  0.1114     8.5189
##         x    2.9954  0.1092    27.4424
##
## Residual standard error: 1.299 on 98 degrees of freedom

abline(Robust_MM, col=5)

methods=c("OLS", "M-bisquare","M-huber","S","MM")
legend("topleft",legend=methods,col=1:5,lwd = .5,horiz=F)
### 3- Remove the outliers --> use OLS

```r
bp = boxplot(y)

lower = median(y) - 1.5 * IQR(y)
upper = median(y) + 1.5 * IQR(y)

new.y = y
new.y[new.y > upper] = NA
new.y[new.y < lower] = NA

new.ols = lm(new.y ~ x)
summary(new.ols)
```

---

lower = \textbf{median} (y) - 1.5 \times \textbf{IQR}(y)
upper = \textbf{median} (y) + 1.5 \times \textbf{IQR}(y)

```r
new.y = y
new.y[new.y > upper] = NA
new.y[new.y < lower] = NA

new.ols = \textbf{lm}(\texttt{new.y} ~ x)
\textbf{summary}(\texttt{new.ols})
```

---

## Call:
```
\textbf{lm}(\texttt{formula = new.y} \sim x)
```

## Residuals:
```
\textbf{Min} \quad \textbf{1Q} \quad \textbf{Median} \quad \textbf{3Q} \quad \textbf{Max}
```
## -4.0188 -0.4928 -0.0038 0.6820 2.0884
##
## Coefficients:
## Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.9129 0.1176 7.761 2.73e-11 ***
## x 2.9789 0.1118 26.640 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.051 on 78 degrees of freedom
## (20 observations deleted due to missingness)
## Multiple R-squared: 0.901, Adjusted R-squared: 0.8997
## F-statistic: 709.7 on 1 and 78 DF,  p-value: < 2.2e-16
##
## compare between two models
par(mfcol=c(1,3))

r2=summary(ols)$r.squared
plot(y~x, main=paste("R^2=",round(r2,2)))
abline(ols,col=2)

new.data=new.ols$model
new.r2=summary(new.ols)$r.squared
plot(new.data$new.y~new.data$x,
     main=paste("R^2=",round(new.r2,2)))
abline(new.ols)

#=======================================================
## 4. Remove the outliers --> Imputate them --> use OLS
#=======================================================
new.y[new.y<mean(new.y, na.rm = T)] <- mean(new.y, na.rm = T) # not run
new2.ols=lm(new.y~x )
s
## Residuals:
## Min 1Q Median 3Q Max
## -4.0499 -0.5792 0.0519 1.0684 3.0819
##
## Coefficients:
## Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.8309 0.1444 5.754 9.97e-08 ***
## x 2.5304 0.1415 17.877 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.444 on 98 degrees of freedom
## Multiple R-squared: 0.7653, Adjusted R-squared: 0.7629
## F-statistic: 319.6 on 1 and 98 DF,  p-value: < 2.2e-16

new2.r2=summary(new2.ols)$r.squared
plot(new.y~x, main=paste("R^2=",round(new2.r2,2)))
abline(new2.ols)
4.2. Application II: Handling Missing Data in R

#------ Prepare the R console
rm(list = ls(all = TRUE))  # Remove all objects in R console
set.seed(09061982)  # Set the seed for reproducible results

#========
n = 100

#=================
# generate the model
#=================
x1 = runif(n)
x2 = rnorm(n)
y = 2 + 2*x1 + 2*x2 + rnorm(n)
mydata = data.frame(y = y, x1 = x1, x2 = x2)

#=================================
# OLS Estimation in complete case
#================================
ols = lm(y ~ ., data = mydata)
summary(ols)

##
## Call:
## lm(formula = y ~ ., data = mydata)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
##-4.3743 -0.6803 -0.0398  0.8327  3.0980
##
## Coefficients:
## Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2.3010  0.2173 10.588  < 2e-16 ***
## x1          1.4987  0.3742  4.005  0.000121 ***
## x2          1.8068  0.1338 13.506  < 2e-16 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.185 on 97 degrees of freedom
## Multiple R-squared: 0.6875, Adjusted R-squared: 0.681
## F-statistic: 106.7 on 2 and 97 DF, p-value: < 2.2e-16

### Generate missing values

```r
library(mice)
X.m = ampute(mydata, mech="MAR", prop = .3)
mis.data = X.m$amp
md.pattern(mis.data)
```

```
##    x2 y x1
## 75  1 1  1  0
## 11  1 1  0  1
##  9  1 0  1  1
##  5  0 1  1  1
##     5 9 11 25
```

### Handle the missing values (by imputation)

#### 1- Imputation with mean / median / mode

```r
library("Hmisc") #install.packages("Hmisc")
impute(mis.data$y, mean) # replace with mean
```

```
##    1  2  3  4  5  6
## 4.25895971 4.13612565 2.51248406 3.02811279* 4.78862371 4.29797486
##  7  8  9 10 11 12
## -2.40211563 6.66297317 2.53558573 6.40344155 2.50706075 0.82731992
## 13 14 15 16 17 18
##  0.12160708 7.90683033 3.99172620 5.23821466 3.75585737 4.13542821
## 19 20 21 22 23 24
##  2.01817748 4.14212435 4.61391631 6.35521494 2.34265227 4.80397411
```

# 25 26 27 28 29 30
## -0.39479251 -0.67064877 3.02811279* 4.4529580 2.13682751 2.33210465
## 31 32 33 34 35 36
##  1.41112495 5.62521709 3.02811279* 6.58397883 3.23972542 1.95696792
## 37 38 39 40 41 42
##  1.96671307 2.61884521 3.02811279* 5.69990554 4.23095395 2.83844396
## 43 44 45 46 47 48
##  5.68721542 1.48302466 3.48162128 3.03504925 1.58205434 1.31988225
## 49 50 51 52 53 54
##  3.02811279* 4.71365184 6.09065793 4.25338900 1.80330014 2.63764575
## 55 56 57 58 59 60
##  1.11465402 2.60232124 2.79087145 6.25713161 2.15909267 2.12889666
## 61 62 63 64 65 66
##  2.63823649 4.34302100 -1.43241773 3.02811279* 5.33434646 3.02811279*
## 67 68 69 70 71 72
```
# 4.30411477   5.02586996   4.72682231   5.64803131 0.96975789 -1.04791948
## 73 74 75 76 77 78
# 3.02811279* 1.77375584   0.52301394   0.93211377   3.70730405   5.54738352
## 79 80 81 82 83 84
# 3.15402724   1.29331211   1.85495048   4.62787165   4.50157667   2.10319755
## 85 86 87 88 89 90
# 2.23215777   5.34761510   3.39623562 -1.64866455 -1.03138494   2.08054016
## 91 92 93 94 95 96
# 3.80960314   0.04550609   3.75700330   3.54720085 3.02811279* 1.48533186
## 97 98 99 100
# -0.52877259   4.16344831   4.22027048   3.28332195
impute(miss.data$x1, mean)
impute(miss.data$x2, mean)
# or if you want to impute manually use this code
#miss.data$y[is.na(miss.data$y)] <- mean(miss.data$y,na.rm = T)

# Compute the accuracy when it is imputed with mean
library(DMwR)
im_mean= regr.eval(y, impute(miss.data$y, mean))
im_median= regr.eval(y, impute(miss.data$y, median))

cbind(im_mean,im_median)
## im_mean im_median
## mae 0.10942473 0.1094941
## mse 0.18412838 0.1841866
## rmse 0.42910183 0.4291696
## mape 0.06506366 0.0652356
#==============================================
# 2- KNN (k-nearest neighbors)Imputation
#==============================================
KNN_data= knnImputation(miss.data)
summary(KNN_data)
## y  x1  x2
## Min. :-2.402 Min. :0.004143 Min. :2.364494
## 1st Qu.: 1.842 1st Qu.:0.208550 1st Qu.: -0.451906
## Median : 3.115 Median :0.465875 Median : 0.026660
## Mean : 3.088 Mean :0.496652 Mean : 0.026660
## 3rd Qu.: 4.424 3rd Qu.:0.808414 3rd Qu.: 0.679296
## Max. : 7.907 Max. :0.983236 Max. : 2.032580
im_KNN= regr.eval(y, KNN_data$y)

cbind(im_mean,im_median,im_KNN)
## im_mean im_median im_KNN
## mae 0.10942473 0.1094941 0.06547363
## mse 0.18412838 0.1841866 0.26803215
## rmse 0.42910183 0.4291696 0.26803215
## mape 0.06506366 0.0652356 0.03803944
#==============================================
# 3- Multiple imputation methods
#==============================================
library("mice")
library("randomForest")
# 1- perform mice imputation, based on Predictive mean matching (PMM)
PMM <- mice(miss.data, m = 10, method = "pmm",printFlag=F )
PMM_data <- complete(PMM) # generate the completed data.
anyNA(PMM_data)
anyNA(PMM_data)
## [1] FALSE
#2- ... based on Classification and regression trees
CRT_data <- complete(miss.data, m = 10, method = "cart",printFlag=F )
#3- ... based on Random forest
RF_data <- complete(miss.data, m = 10, method = "rf",printFlag=F )
#4- ... based on Bayesian linear regression
Bayes_data <- complete(miss.data, m = 10, method = "norm",printFlag=F )
# compute the accuracy of all methods
im_PMM= regr.eval(y, PMM_data$y)

#==============================================
im_CRT= \texttt{regr.eval}(y, \texttt{CRT\_data}\$y)
im_RF= \texttt{regr.eval}(y, \texttt{RF\_data}\$y)
im_Bayes= \texttt{regr.eval}(y, \texttt{Bayes\_data}\$y)

\texttt{cbind}\,(\texttt{im\_mean,im\_median,im\_KNN,}
\texttt{im\_PMM, im\_CRT, im\_RF, im\_Bayes})

## im\_mean im\_median im\_KNN im\_PMM im\_CRT im\_RF
## mae 0.10942473 0.1094941 0.06547363 0.11019595 0.11053526 0.13208236
## mse 0.18412838 0.1841866 0.07184123 0.16251412 0.17604581 0.28277237
## rmse 0.42910183 0.4291696 0.26803215 0.40313040 0.41957813 0.53176345
## mape 0.06506366 0.0652356 0.03803944 0.05158428 0.05699919 0.05414655

## im\_Bayes
## mae 0.16779583
## mse 0.48765064
## rmse 0.69831987
## mape 0.06598063

\texttt{compare=cbind}(\texttt{im\_mean,im\_median,im\_KNN,}
\texttt{im\_PMM, im\_CRT, im\_RF, im\_Bayes})

\texttt{windows}(400,500)
\texttt{barplot}(t(compare), beside = T,col = \texttt{rainbow}(7),
\texttt{ylim}=c(0,1.3*\texttt{max}(compare)),
\texttt{legend.text} = c("Mean","Median","KNN",
"PMM","CRT","RF","Bayes"),
\texttt{args.legend} = \texttt{list}(x = "topleft",ncol=3))

#==============================================
# comparison these methods in regression
#==============================================
\texttt{ols}\,= \texttt{lm}(y\sim.,\texttt{data} = \texttt{mydata})
\texttt{PMM}\,= \texttt{lm}(y\sim.,\texttt{data} = \texttt{PMM\_data})
\texttt{CRT}\,= \texttt{lm}(y\sim.,\texttt{data} = \texttt{CRT\_data})
\texttt{RF}\,= \texttt{lm}(y\sim.,\texttt{data} = \texttt{RF\_data})
\texttt{Bayes}\,= \texttt{lm}(y\sim.,\texttt{data} = \texttt{Bayes\_data})

\texttt{AIC}\,= \texttt{AIC}(\texttt{ols,PMM,CRT,RF,Bayes})
\texttt{BIC}\,= \texttt{BIC}(\texttt{ols,PMM,CRT,RF,Bayes})
5. A Monte Carlo Simulation Study

We will make a simulation study to examine the efficiency of imputation methods of missing. See Abonazel [26, 27] for information about how to make Monte Carlo simulation studies using R.

![Efficiency of MI methods](image)

5. A Monte Carlo Simulation Study

We will make a simulation study to examine the efficiency of imputation methods of missing. See Abonazel [26, 27] for information about how to make Monte Carlo simulation studies using R.

![Simulation steps](image)
rm(list = ls(all = TRUE))
set.seed(09061982)
library(mice); library(DMwR); library("randomForest")
loop=500
#======================
# 1- Generate the model
#======================
n =100
x1=rnorm(n); x2 = rnorm(n)
sum_select=0
for(i in 1:loop){
y = 1 + x1 + x2 + rnorm(n)
mydata = data.frame(y=y, x1=x1, x2=x2)

#======================
# 2- Generate missing values
#======================
X.m=ampute(mydata, mech="MAR", prop =.3)
miss.data=X.m$amp

#======================
# 3- Imputation methods
#======================
KNN_data=knnImputation(miss.data)
PMM_data <- complete(mice(miss.data, method = "pmm", printFlag=F))
CRT_data <- complete(mice(miss.data, method = "cart", printFlag=F))
RF_data <- complete(mice(miss.data, method = "rf", printFlag=F))
Bayes_data <- complete(mice(miss.data, method = "norm", printFlag=F))

#======================
# 4- Comparison these methods by AIC and BIC
#======================
KNN=lm(y~., KNN_data)
PMM=lm(y~., PMM_data)
CRT=lm(y~., CRT_data)
RF=lm(y~., RF_data)
Bayes=lm(y~.,Bayes_data)

AIC=AIC(KNN, PMM, CRT, RF, Bayes)
BIC=BIC(KNN, PMM, CRT, RF, Bayes)

select=as.matrix(cbind(AIC, BIC)[-c(1,3)])
simulation_results = sum_select/loop
simulation_results

windows()
barplot((simulation_results),beside = T,main = "Simulation Results",
ylim = c(0,1.2*max(select)), col = rainbow(5),
args.legend = list(x = "topleft",
ncol=5), legend.text = c ("KNN","PMM", "CRT", "RF", "Bayes") )

201
The simulation results indicate that KNN method is better than the other methods that were used in this study. Note that this conclusion may change if the simulation factors (missing mechanism, variables type, options of the imputation methods, etc.) are changed.

6. Conclusion

In this paper, handling methods of outliers and missing data were studied using R. Practical evidence was provided to researchers to deal with these problems (outliers and missing data) in regression with R. We can conclude that OLS residuals must be examined initially, if they have outliers, a robust estimation method should be used instead of OLS to get an efficient estimation of the regression model. While in the case of missing data, we note that different handling methods of missing data must be examined to determine a good estimation of missing values, because there is no one suitable method for all datasets. According to our simulation study, we find that KNN method is better than the other methods to estimate the missing values in regression models.

References


