Handling Missing Values

Detecting, Excluding and Imputing NA’s

Contents

Introduction

- Missing values in data is a common phenomenon in real world problems and can impact the statistical analysis if not treated properly.

- You need to know the pattern of missingness and how to treat them is a requirement to reduce the bias and to produce accurate inference from data.

- Lets get familiar with the pattern of missingness and explore various options of how to deal with them.

Missing Data Mechanism

Three types of missing data:

- Missing Completely at Random (MCAR):

  MCAR happens when missingness is totally unrelated to other variables in the dataset. Example: Missing data on Gender

- Missing at Random (MAR):

  MAR happens when the missingness is related to the information in your study. Other variables (but not the variable that is missing itself) in the dataset can be used to predict missingness. For example: Missing data for variable income can be estimated using other variables such as experience and designation.

  Most missing data imputation methods are based on MAR

- Missing not at Random (MNAR):

  MNAR happens when data is missing for a specific segment. For example: Missing data for variable income for all senior managers.

Data Snapshot

basicsalary data consist salary of each employee with it's Location & Grade. The data has 12 rows and 6 columns with 2 missing values.

Variables

First_Name	Last_Name	Grade	Location	ba	ms
Alan	Brown	GR1	DELHI	17990	16070
Agatha	Williams	GR2	MUMBAI	12390	6630

Observations

Columns	Description	Type	Measurement	Possible values
First_Name	First Name	character	-	-
Last_Name	Last Name	character	-	-
Grade	Grade	character	GR1, GR2	2
Location	Location	character	DELHI, MUMBAI	2
ba	Basic Allowance	numeric	Rs.	positive values
ms	Management Supplements	numeric	Rs.	positive values

Replacing Missing Values with NA while Importing the Data

A missing value is one whose value is unknown. Missing values in R appears as NA. NA is not a string or a numeric value, but an indicator of missingness. Our data has two missing values, let's see what happens when we import this data in R.

```
# Import Data and check how R treats missing data while importing
```

```
salary_data<-read.csv("basicsalary.csv",header=TRUE)
salary_data
```

```
   First_Name Last_Name Grade Location    ba    ms
1        Alan     Brown   GR1    DELHI 17990 16070
2      Agatha  Williams   GR2   MUMBAI 12390  6630
3      Rajesh     Kolte   GR1   MUMBAI 19250 14960
4       Ameet    Mishra   GR2    DELHI 14780  9300
5        Neha       Rao         MUMBAI 19235 15200
6       Sagar    Chavan   GR2   MUMBAI 13390  6700
7       Aaron     Jones   GR1   MUMBAI 23280 13490
8        John     Patil   GR2   MUMBAI 13500 10760
9       Sneha     Joshi   GR1    DELHI 20660    NA
10     Gaurav     Singh   GR2    DELHI 13760 13220
11      Adela    Thomas   GR2    DELHI 13660  6840
12       Anup      Save   GR2   MUMBAI 11960  7880
```

Note **read.csv()** replaces blank fields in numeric column with NA and in character column it is kept as blank only.
If any field contains string "**NA**", then **read.csv()** assumes that the value is missing and converts it into **NA**.

Replacing Missing Values with NA while Importing the Data

If we want R to recognize all missing values the same way(the "correct" way i.e. **NA**) when we import the data, we will use **na.strings=** argument.

```
salary_data<-read.csv("basicsalary.csv",header=TRUE,na.strings="")
salary_data
```

```
   First_Name Last_Name Grade Location    ba    ms
1        Alan     Brown   GR1    DELHI 17990 16070
2      Agatha  Williams   GR2   MUMBAI 12390  6630
3      Rajesh     Kolte   GR1   MUMBAI 19250 14960
4       Ameet    Mishra   GR2    DELHI 14780  9300
5        Neha       Rao  <NA>   MUMBAI 19235 15200
6       Sagar    Chavan   GR2   MUMBAI 13390  6700
7       Aaron     Jones   GR1   MUMBAI 23280 13490
8        John     Patil   GR2   MUMBAI 13500 10760
9       Sneha     Joshi   GR1    DELHI 20660    NA
10     Gaurav     Singh   GR2    DELHI 13760 13220
11      Adela    Thomas   GR2    DELHI 13660  6840
12       Anup      Save   GR2   MUMBAI 11960  7880
```

- **read.csv()** by default leaves blank fields in character variable as is, so we need to specify a blank string "" to **na.strings=** for R to treat it as missing value and convert it to **NA.**
- **na.strings=** takes a character vector of strings which are to be interpreted as **NA** values.
- The **<NA>** vs **NA** just means that some of our columns are character and some are numeric.

Replacing Missing Values with NA while Importing the Data

Our data might employ a different string to signal missing values like this:

	First_Name	Last_Name	Grade	Location	ba	ms
1	Alan	Brown	GR1	DELHI	17990	16070
2	Agatha	Williams	GR2	MUMBAI	12390	6630
3	Rajesh	Kolte	GR1	MUMBAI	19250	14960
4	Ameet	Mishra	GR2	DELHI	14780	9300
5	Neha	Rao	missing	MUMBAI	19235	15200
6	Sagar	Chavan	GR2	MUMBAI	13390	6700
7	Aaron	Jones	GR1	MUMBAI	23280	13490
8	John	Patil	GR2	MUMBAI	13500	10760
9	Sneha	Joshi	GR1	DELHI	20660	missing
10	Gaurav	Singh	GR2	DELHI	13760	13220
11	Adela	Thomas	GR2	DELHI	13660	6840
12	Anup	Save	GR2	MUMBAI	11960	7880

To proceed with the analysis we will have to convert these missing values to correct missing value notation i.e. **NA** for R to recognise these as missing values.

Replacing Missing Values with NA while Importing the Data

```
# Convert ‘missing’ value to ‘NA’
```

```
salary_data<-read.csv("basicsalary.csv", header=TRUE,
na.strings=“missing")
salary_data
```

```
 First_Name Last_Name Grade Location    ba    ms
1          Alan     Brown   GR1    DELHI 17990 16070
2        Agatha  Williams   GR2   MUMBAI 12390  6630
3        Rajesh     Kolte   GR1   MUMBAI 19250 14960
4         Ameet    Mishra   GR2    DELHI 14780  9300
5          Neha       Rao  <NA>   MUMBAI 19235 15200
6         Sagar    Chavan   GR2   MUMBAI 13390  6700
7         Aaron     Jones   GR1   MUMBAI 23280 13490
8          John     Patil   GR2   MUMBAI 13500 10760
9         Sneha     Joshi   GR1    DELHI 20660    NA
10       Gaurav     Singh   GR2    DELHI 13760 13220
11        Adela    Thomas   GR2    DELHI 13660  6840
12         Anup      Save   GR2   MUMBAI 11960  7880
```

na.strings= converts ‘**missing**’ value to ‘**NA**’

Detecting NA's

```
# Check whether our data has missing values or not
na.fail(salary_data)
Error in na.fail.default(salary_data) : missing values in object
```

na.fail() returns the object only if it contains no missing values otherwise it returns an error message indicating there is one or more missing values in the data.

```
# Check total missing values
sum(is.na(salary_data))
[1] 2
```

is.na for dataframe returns a logical matrix with the same dimensions as the data frame, and with dimnames taken from the row and column names of the data frame. Here, **sum()** returns the total no. of missing values in the data.

Detecting NA's

```
# Check the count of NA's for each column
```

```r
na_count<-sapply(salary_data,function(y) sum(is.na(y)))
na_count
```

```
First_Name  Last_Name      Grade   Location         ba         ms
         0          0          1          0          0          1
```

sapply() is a part of 'apply' family functions. It applies the given user defined function on the data specified.

```
# Check Number of missing data per column
```

```r
summary(salary_data)
```

```
# Output
```

```
   First_Name    Last_Name   Grade      Location        ba                 ms
 Aaron   :1    Brown  :1   GR1 :4   DELHI :5   Min.   :11960   Min.   : 6630
 Adela   :1    Chavan :1   GR2 :7   MUMBAI:7   1st Qu.:13472   1st Qu.: 7360
 Agatha  :1    Jones  :1   NA's:1              Median :14270   Median :10760
 Alan    :1    Joshi  :1                       Mean   :16155   Mean   :11005
 Ameet   :1    Kolte  :1                       3rd Qu.:19239   3rd Qu.:14225
 Anup    :1    Mishra :1                       Max.   :23280   Max.   :16070
 (Other):6     (Other):6                                       NA's   :1
```

Using **summary()** we can check how many NA's our data contains.

Excluding Missing Values from Analysis

- When R encounters missing value, it attempts to perform the requested procedure and returns a missing (**NA**) value as a result. One way of dealing with missing values is to remove them while performing that procedure.

```
x<-c(10,30,12,NA, 9)
mean(x)
```

```
[1] NA
```

```
# We can calculate mean by dropping missing value like in the next example.
# remove missing value
```

```
mean(x,na.rm=TRUE)
```

```
[1] 15.25
```

na.rm= if set to **TRUE** will remove all **NA**'s while performing the requested procedure. By default, it is set to **FALSE**.

* `na.rm=` can be used with many R functions, such as mean, median, sd, var, and so forth.

Excluding Missing Values from Analysis

Case wise deletion (complete case analysis) is the easiest way to deal with missing data. It simply removes all the cases with missing data anywhere in the data i.e. analyzing only the cases with complete data.

```
# Case wise deletion
na.omit(salary_data)
# Output
   First_Name Last_Name Grade Location    ba    ms
1        Alan     Brown   GR1    DELHI 17990 16070
2      Agatha  Williams   GR2   MUMBAI 12390  6630
3      Rajesh     Kolte   GR1   MUMBAI 19250 14960
4       Ameet    Mishra   GR2    DELHI 14780  9300
6       Sagar    Chavan   GR2   MUMBAI 13390  6700
7       Aaron     Jones   GR1   MUMBAI 23280 13490
8        John     Patil   GR2   MUMBAI 13500 10760
10     Gaurav     Singh   GR2    DELHI 13760 13220
11      Adela    Thomas   GR2    DELHI 13660  6840
12       Anup      Save   GR2   MUMBAI 11960  7880
```

na.omit() is used for case deletion.
Here, **na.omit()** removes 2 rows that contains missing values

* Complete case analysis is widely used method for handling missing data, and is a default method in many statistical packages. But it has limitations like it may introduce bias and some useful information will be omitted from analysis.

Data Snapshot

Consumerpreference data consist information about 73 respondents & their preferences about 7 attributes on scale of 1-Least Important to 5-Most Important.

Variables

Sn	Gender	Color	Weight	Shape	Camera	Ram	Processor	Internet
1	M	3	3	4	3	3	3	3

Observations

Columns	Description	Type	Measurement	Possible values
Sn	Serial No. (Index Variable)	Character		-
Gender	Gender	Categorical	M;F	2
Color	Color	Categorical	1 to 5	5
Weight	Weight	Categorical	1 to 5	5
Shape	Shape	Categorical	1 to 5	5
Camera	Camera	Categorical	1 to 5	5
Ram	Ram	Categorical	1 to 5	5
Processor	Processor	Categorical	1 to 5	5
Internet	Internet	Categorical	1 to 5	5

Imputing Missing Values

- If the amount of missing data is very small relative to the size of the data then case wise deletion may be the best strategy in order not to bias the analysis, however deleting available data points deprives the data of some amount of information.
- Basically, we need to decide how we're going to use our missing data, if at all, then either remove cases from our data or impute missing values before wiping out potentially useful data points from our data and proceed with out analysis.
- Single imputation can be done by replacing missing values with the mean / median / mode (or any other procedure) of the other values in the variable.

In this tutorial we will primarily focus on performing single imputation .

> * Mean/ Median / mode imputations are simple but, like complete case analysis, can introduce bias on mean and deviation. Furthermore, they ignore relationship with other variables

Imputing Missing Values

Treating missing values in the variable 'Processor'

```
# Import the data and
# Check number of missing values for each variable
```

```
consumer_pref<-read.csv("consumerpreference.csv",header=TRUE)
sapply(consumer_pref,function(y) sum(is.na(y)))
```

```
       Sn     Gender      Color     Weight      Shape     Camera
        0          0          0          1          0          1
      Ram  Processor   Internet
        0          1          0
```

Our data has 3 missing values.

```
# Treating missing values in variable 'Processor'
# Median imputation
```

```
consumer_pref$Processor[is.na(consumer_pref$Processor)]<-median(consum
er_pref$Processor,na.rm=TRUE)
sum(is.na(consumer_pref$Processor))
```

```
[1] 0
```

Here, we are calculating median of the values in variable 'Processor' using **median()** and replacing the **NA** value with the median value. Now there are no missing values in variable '**Processor**'.

Imputing Missing Values

Treating missing values in the variable ‘Camera’

```
# Check the ‘Gender’ value corresponding to the NA value
```

```
g<-subset(consumer_pref,is.na(consumer_pref$Camera),select=c(Gender))
g
```

```
9 M
```

Corresponding ‘**Gender**’ value to the **NA** in Variable ‘**Camera**’ is ‘**M**’.

```
# Calculate median of values in variable ‘Camera’ whose corresponding
# ‘Gender value is ‘M’
```

```
g_subset_camera<-subset(consumer_pref,consumer_pref$Gender=="M",
select=c(Camera))
consumer_pref$Camera[is.na(consumer_pref$Camera)]<-apply(g_subset_came
ra,2,function(x) median(x,na.rm=TRUE))
sum(is.na(consumer_pref$Camera))
```

```
[1]
```

Using **subset()** we have subsetted the values of variable ‘**Camera**’ whose corresponding ‘**Gender**’ value is ‘**M**’. Then, using **apply()** (because **g_subset_camera** is a dataframe), calculated the median of the subsetted values and replaced **NA** with that value. As a result, we are left with no missing values in variable ‘**Camera**’.

Imputing Missing Values

Treating missing values in the variable 'Weight'

```
# Replace the NA value in 'Weight' with the corresponding value
# in the variable which is highly correlated with 'Weight'
```

```
cor(consumer_pref[3:9],use="pairwise.complete.obs")
```

```
               Color     Weight      Shape     Camera        Ram
Color      1.0000000  0.9172664  0.9421043 -0.2749990 -0.2779061
Weight     0.9172664  1.0000000  0.8656048 -0.3434737 -0.2978006
shape      0.9421043  0.8656048  1.0000000 -0.2297692 -0.2751586
Camera    -0.2749990 -0.3434737 -0.2297692  1.0000000  0.8326233
Ram       -0.2779061 -0.2978006 -0.2751586  0.8326233  1.0000000
Processor -0.2319926 -0.2650182 -0.2296934  0.8522504  0.9049753
Internet  -0.2556812 -0.2839609 -0.2320419  0.8601564  0.8957823
           Processor   Internet
Color     -0.2319926 -0.2556812
Weight    -0.2650182 -0.2839609
shape     -0.2296934 -0.2320419
Camera     0.8522504  0.8601564
Ram        0.9049753  0.8957823
Processor  1.0000000  0.8631503
Internet   0.8631503  1.0000000
```

- Here, we are computing the correlation between the 7 attributes using **cor()** giving the vector of indices of those 7 attributes..
- **use="pairwise.complete.obs"** uses the non-**NA** values when calculating the correlation between 7 attributes.

Interpretation :

- We have found that 'Weight' is highly correlated with 'Color' as they have the highest correlation coefficient.

Imputing Missing Values

```
s<-subset(consumer_pref,is.na(consumer_pref$Weight))
s
```

```
      Sn Gender Color Weight Shape Camera Ram Processor Internet
10    10      M     5     NA     4      3   2         3        2
```

This command subsets the row which contains NA in the variable ‘Weight’.

```
consumer_pref$Weight[is.na(consumer_pref$Weight)]<-s$Color
consumer_pref[10,]
```

```
      Sn Gender Color Weight Shape Camera Ram Processor Internet
10    10      M     5      5     4      3   2         3        2
```

This command replaces **NA** in ‘**Weight**’ with corresponding value of highly correlated variable i.e. ‘**Color**’

Quick Recap

In this session, we learnt how to deal with different types of missing values. Here is the quick recap:

Replacing missing values with NA	• Blank fields in numeric column are replaced with NA and in character column it is kept as blank only while importing data. • **`na.strings=`**: to specify the character vector of strings which are to be treated as NA
Recoding values to missing	• **`na.fail:`** returns the object if it does not contain any missing values otherwise it gives error indicating the missingness in data. • **`is.na():`** returns the logical matrix which indicates which elements are missing. • **`sapply():`** using this function we can calculate the count of NA's per column • **`summary()`**: check number of NA's per variable
Excluding missing values from analysis	• **`na.rm=TRUE:`** drops missing value • **`na.omit():`** performs case wise deletion
Imputing missing values	• Single imputation can be done by replacing missing values with the mean / median / mode (or any other procedure) of the other values in the variable