**AIM: DATA PREPROCESSING IN R**

**THEORY:**

# Data Preprocessing Techniques

Data preprocessing is a data mining technique which is used to transform the raw data in a useful and efficient format.

**Steps Involved in Data Preprocessing:**

**1. Data Cleaning:**

The data can have many irrelevant and missing parts. To handle this part, data cleaning is done. It involves handling of missing data, noisy data etc.

**(a). Missing Data:**

This situation arises when some data is missing in the data. It can be handled in various ways. Some of them are:

1. **Ignore the tuples:** This approach is suitable only when the dataset we have is quite large and multiple values are missing within a tuple.
2. **Fill the Missing values:** There are various ways to do this task. You can choose to fill the missing values manually, by attribute mean or the most probable value.

**(b). Noisy Data:**

Noisy data is a meaningless data that can’t be interpreted by machines.It can be generated due to faulty data collection, data entry errors etc. It can be handled in following ways :

1. **Binning Method:** This method works on sorted data in order to smooth it. The whole data is divided into segments of equal size and then various methods are performed to complete the task. Each segmented is handled separately. One can replace all data in a segment by its mean or boundary values can be used to complete the task.
2. **Regression:** Here data can be made smooth by fitting it to a regression function.The regression used may be linear (having one independent variable) or multiple (having multiple independent variables).
3. **Clustering:** This approach groups the similar data in a cluster. The outliers may be undetected or it will fall outside the clusters.

**2. Data Transformation:**

This step is taken in order to transform the data in appropriate forms suitable for mining process. This involves following ways:

1. **Normalization :-** It is done in order to scale the data values in a specified range (-1.0 to

1.0 or 0.0 to 1.0)

1. **Attribute Selection :-** In this strategy, new attributes are constructed from the given set of attributes to help the mining process.
2. **Discretization :-** This is done to replace the raw values of numeric attribute by interval levels or conceptual levels.
3. **Concept Hierarchy Generation :-** Here attributes are converted from level to higher level in hierarchy. For Example-The attribute “city” can be converted to “country”.

**3. Data Reduction:**

Since data mining is a technique that is used to handle huge amount of data. While working with huge volume of data, analysis became harder in such cases. In order to get rid of this, we uses data reduction technique. It aims to increase the storage efficiency and reduce data storage and analysis costs.

The various steps to data reduction are:

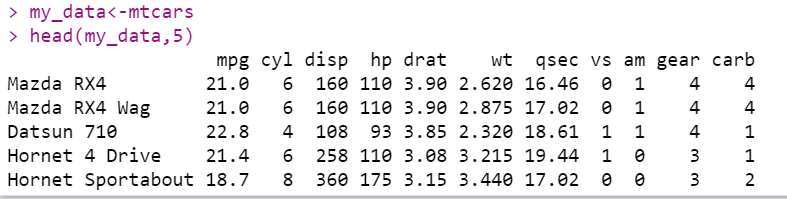
1. **Data Cube Aggregation :-** Aggregation operation is applied to data for the construction of the data cube.
2. **Attribute Subset Selection :-** The highly relevant attributes should be used, rest all can be discarded. For performing attribute selection, one can use level of significance and p- value of the attribute.the attribute having p-value greater than significance level can be discarded.
3. **Numerosity Reduction :-** This enable to store the model of data instead of whole data, for example: Regression Models.
4. **Dimensionality Reduction :-** This reduce the size of data by encoding mechanisms.It can be lossy or lossless. If after reconstruction from compressed data, original data can be retrieved, such reduction are called lossless reduction else it is called lossy reduction. The two effective methods of dimensionality reduction are: Wavelet transforms and PCA (Principal Componenet Analysis).
5. **PRINT HEAD OF MTCARS DATASET [PREDEFINED IN R]:**

**SOURCE CODE:**

my\_data<-mtcars

head(my\_data,5)

**OUTPUT:**



1. **READING DATA OF SPECFIC ROWS & COLUMNS:**

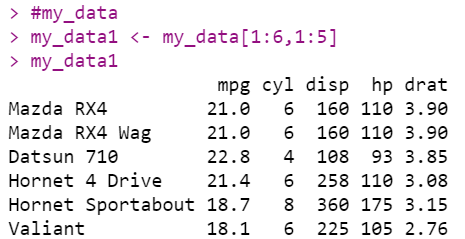
**SOURCE CODE:**

#my\_data

my\_data1 <- my\_data[1:6,1:5]

my\_data1

**OUTPUT:**



1. **RENAME COLUMN NAME USING DPLYR:**

**SOURCE CODE:**

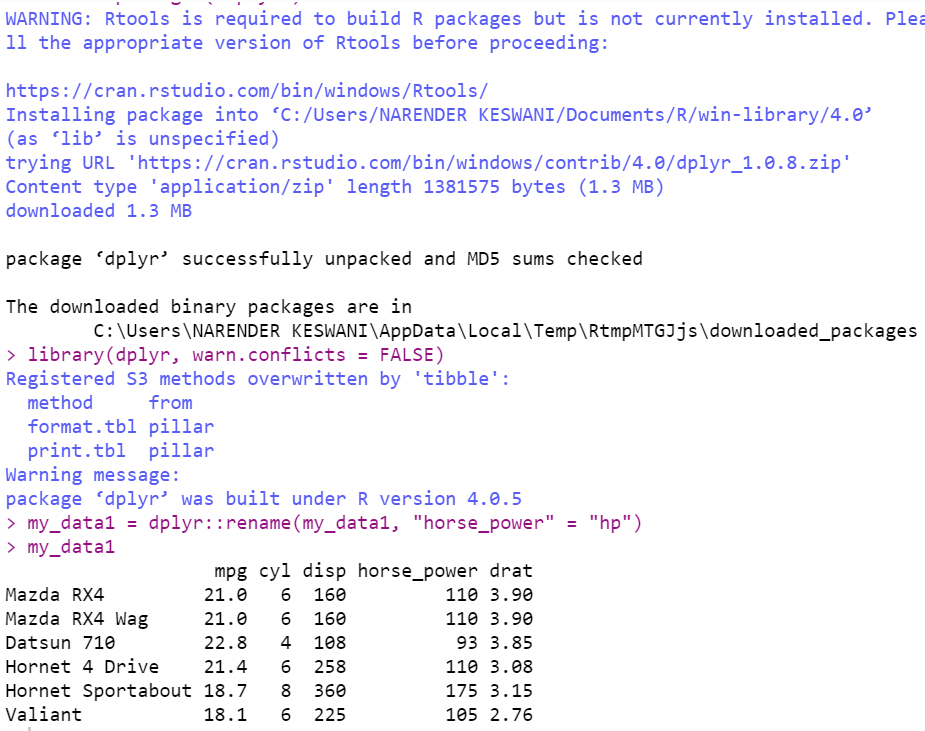
install.packages("dplyr")

library(dplyr, warn.conflicts = FALSE)

my\_data1 = dplyr::rename(my\_data1, "horse\_power" = "hp")

my\_data1

**OUTPUT:**



1. **ADDING NEW COLUMN:**

**SOURCE CODE:**

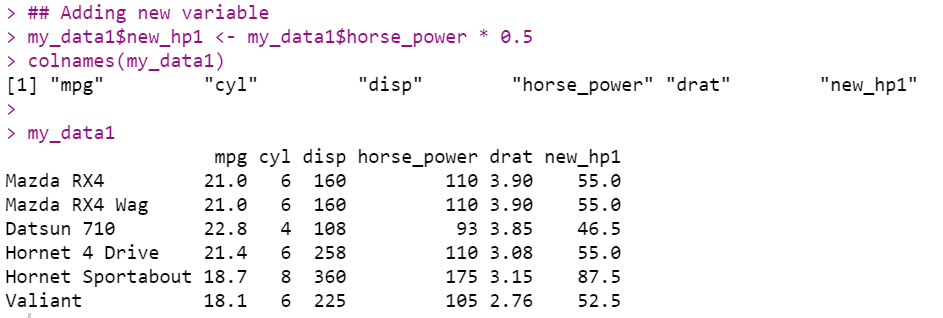
## Adding new variable

my\_data1$new\_hp1 <- my\_data1$horse\_power \* 0.5

colnames(my\_data1)

my\_data1

**OUTPUT:**



1. **READING DATA FROM CSV FILE:**

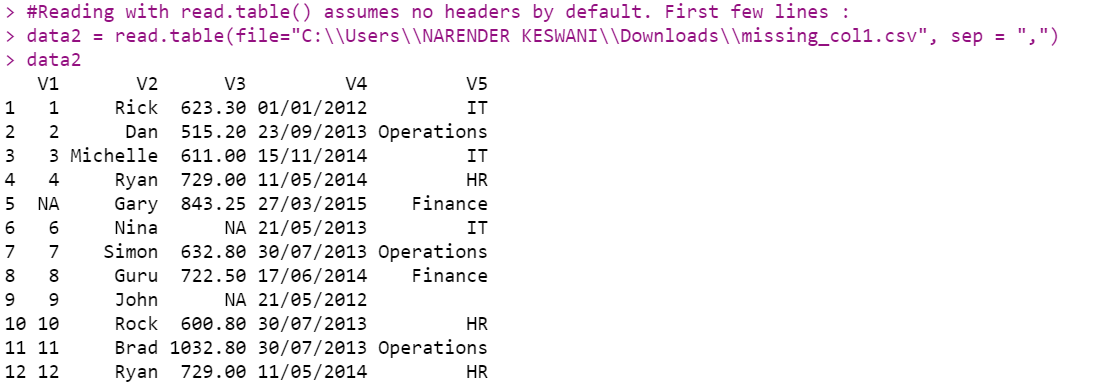
**SOURCE CODE:**

#Reading with read.table() assumes no headers by default. First few lines :

data2 = read.table(file="C:\\Users\\NARENDER KESWANI\\Downloads\\missing\_col1.csv", sep = ",")

data2

**OUTPUT:**



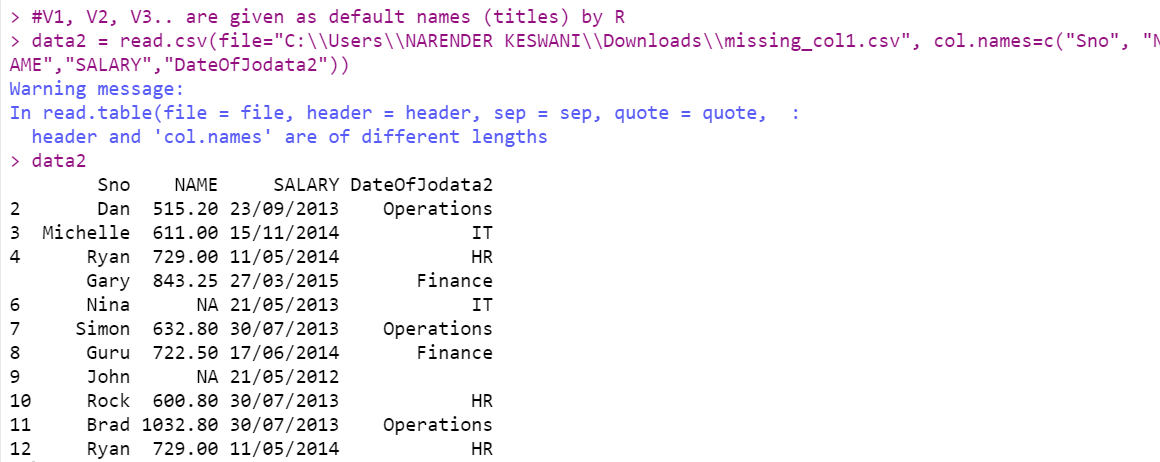
1. **READING DATA FROM SPECFIC COLUMNS OF CSV FILE:**

**SOURCE CODE:**

#V1, V2, V3.. are given as default names (titles) by R

data2 = read.csv(file="C:\\Users\\NARENDER KESWANI\\Downloads\\missing\_col1.csv", col.names=c("Sno", "NAME","SALARY","DateOfJodata2"))

**OUTPUT:**



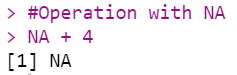
1. **OPERATION WITH NA:**

**SOURCE CODE:**

#Operation with NA

NA + 4

**OUTPUT:**



1. **CREATE A VECTOR V WITH 1 NA VALUE:**

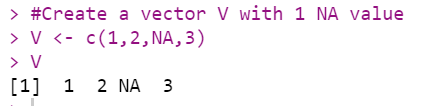
**SOURCE CODE:**

#Create a vector V with 1 NA value

V <- c(1,2,NA,3)

V

**OUTPUT:**



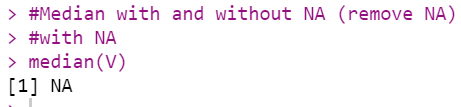
1. **FIND MEDIAN:**
2. **WITH NA:**

**SOURCE CODE:**

#Median with NA

median(V)

**OUTPUT:**



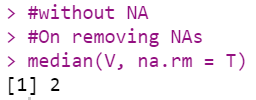
1. **WITHOUT NA:**

**SOURCE CODE:**

#On removing NAs

median(V, na.rm = T)

**OUTPUT:**



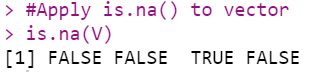
1. **CHECK WHETHER IT IS NA OR NOT:**

**SOURCE CODE:**

#Apply is.na() to vector

is.na(V)

**OUTPUT:**



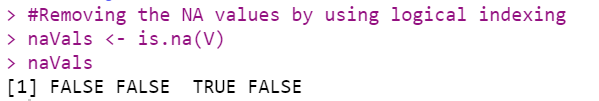
1. **REMOVING THE NA VALUES BY USING LOGICAL INDEXING:**

**SOURCE CODE:**

#Removing the NA values by using logical indexing

naVals <- is.na(V)

**OUTPUT:**

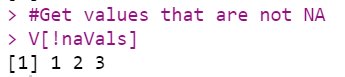


1. **Get values that are not NA**

**SOURCE CODE:**

V[!naVals]

**OUTPUT:**



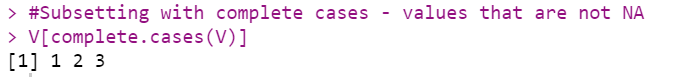
1. **SUBSETTING WITH COMPLETE CASES - VALUES THAT ARE NOT NA:**

**SOURCE CODE:**

#Subsetting with complete cases - values that are not NA

V[complete.cases(V)]

**OUTPUT:**



1. **SUBSETTING A DATA FRAME WITH COMPLETE CASES:**

**SOURCE CODE:**

#Subsetting a data frame with complete cases

#Complete Data of Prime Ministers. Notice NAs

dataC <- read.csv(file ="C:\\Users\\NARENDER KESWANI\\Downloads\\na\_data.csv", na.strings = "")

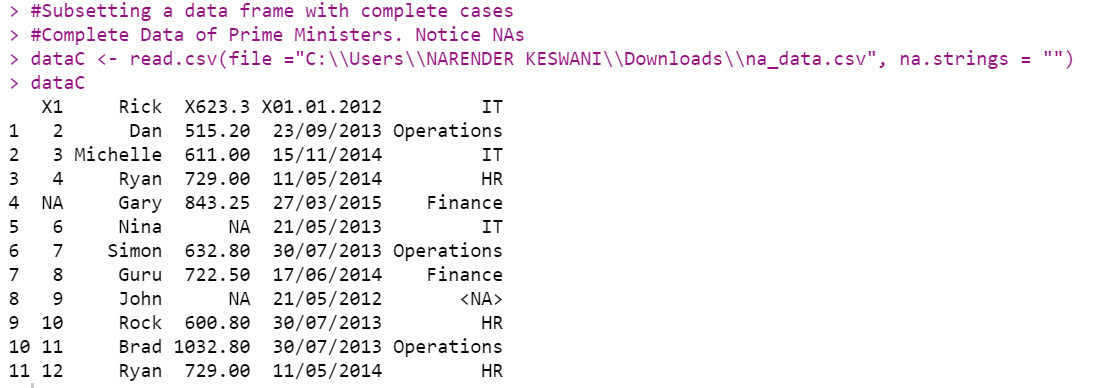
dataC

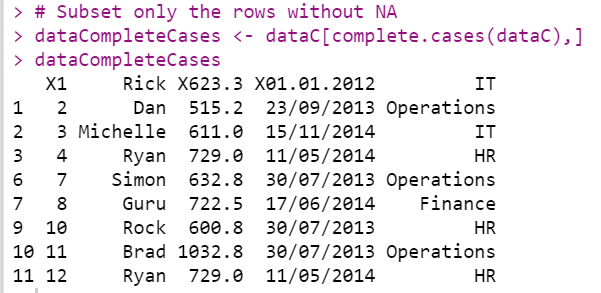
# Subset only the rows without NA

dataCompleteCases <- dataC[complete.cases(dataC),]

dataCompleteCases

**OUTPUT:**





1. **MEAN IMPUTATION & MEDIAN IMPUTATION**

**SOURCE CODE:**

install.packages('Hmisc')

library(Hmisc)

## create a vector

x = c(1,2,3,NA,4,4,NA)

x

# mean imputation - from package, mention name of function to be used

x <- impute(x, fun = mean)

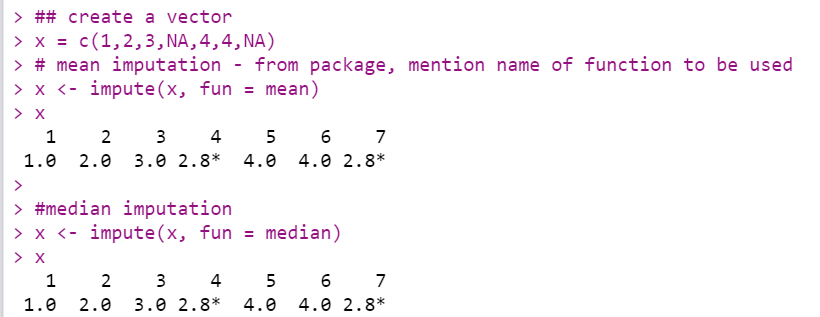
x

#median imputation

x <- impute(x, fun = median)

x

**OUTPUT:**



1. **CONVERT:**
2. **Convert Character into Factor(categorical data):**

**SOURCE CODE:**

#Convert Character into Factor(categorical data)

#Create gender vector

gender\_vector <- c("Male", "Female", "Female", "Male", "Male")

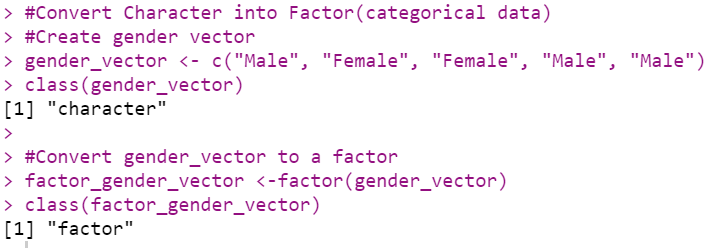
class(gender\_vector)

#Convert gender\_vector to a factor

factor\_gender\_vector <-factor(gender\_vector)

class(factor\_gender\_vector)

**OUTPUT:**



1. **CREATE ORDINAL CATEGORICAL VECTOR:**

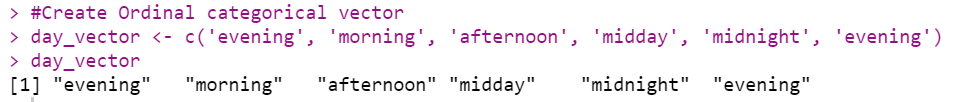
**SOURCE CODE:**

#Create Ordinal categorical vector

day\_vector <- c('evening', 'morning', 'afternoon', 'midday', 'midnight', 'evening')

day\_vector

**OUTPUT:**



1. **CONVERT VECTOR INTO A FACTOR WITH ORDERED LEVEL:**

**SOURCE CODE:**

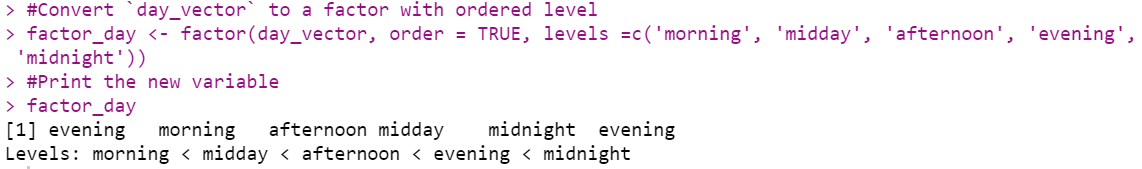
#Convert `day\_vector` to a factor with ordered level

factor\_day <- factor(day\_vector, order = TRUE, levels =c('morning', 'midday', 'afternoon', 'evening', 'midnight'))

#Print the new variable

factor\_day

**OUTPUT:**



1. **CREATE DATAFRAME FROM VECTOR:**

**SOURCE CODE:**

# Creating vectors

age <- c(40, 49, 48, 40, 67, 52, 53)

salary <- c(103200, 106200, 150200, 10606, 10390, 14070, 10220)

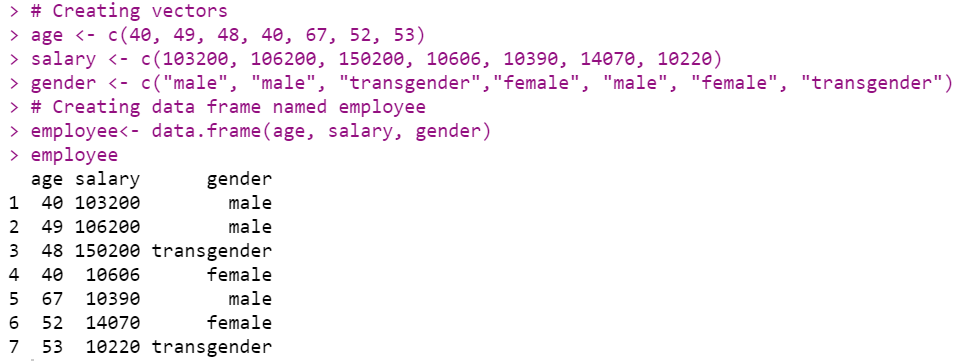
gender <- c("male", "male", "transgender","female", "male", "female", "transgender")

# Creating data frame named employee

employee<- data.frame(age, salary, gender)

employee

**OUTPUT:**



1. **CERATE FACTOR WITH LABELS:**

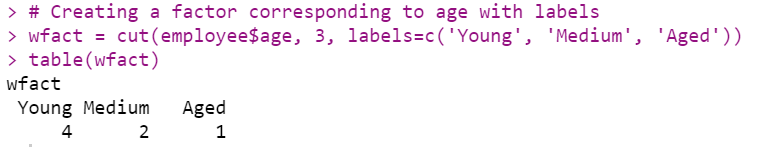
**SOURCE CODE:**

# Creating a factor corresponding to age with labels

wfact = cut(employee$age, 3, labels=c('Young', 'Medium', 'Aged'))

table(wfact)

**OUTPUT:**



**CONCLUSION:**

From this practical, I have learned data preprocessing techniques in R.