**R VISUALIZATION**

**1. DATA CLEANSING**

Data Cleaning is the process to transform raw data into consistent data that can be easily analyzed.

**Purpose of Data Cleaning**

The following are the various purposes of data cleaning:

1. Eliminate Errors
2. Eliminate Redundancy
3. Increase Data Reliability
4. Delivery Accuracy
5. Ensure Consistency
6. Assure Completeness
7. Standardize your approach

The first state (aka Raw data) is the data as it comes in. Raw data may lack headers, contain wrong data types, wrong category labels, unknown or unexpected character encoding, and so on. Once this preprocessing has taken place, data can be deemed Technically correct Data. That is, in this state data can be read into an R data.frame, with correct names, types, and labels, without further trouble. However, this does not mean that the values are error-free or complete. Consistent data is the stage where data is ready for statistical inference. It is the data that most statistical theories use as a starting point.

**Cleaning Data in R**

Here, this involves various steps, as from the initial raw data have to move toward the consistent and highly efficient data which is ready to be implemented as per the requirements and produces the highly precise and accurate statistical results.

***head(airquality)***

In R, missing values are represented by the symbol **NA** (not available). Impossible values (e.g., dividing by zero) are represented by the symbol **NaN** (not a number).

To handle the missing value we will check the columns of the datasets, if we found some missing data inside the columns then this generates the NA values as an output, which can be not good for every model. So let’s check it using [mean()](https://www.geeksforgeeks.org/calculate-arithmetic-mean-in-r-programming-mean-function/) methods.

***mean(airquality$Solar.R)***

***mean(airquality$Wind)***

***9.957516***

This means there are no missing values in this column

**Removing data irregularities with is.na() function**

New\_df = airquality

New\_df$Ozone = ifelse(is.na(New\_df$Ozone),

median(New\_df$Ozone,

na.rm = TRUE),

New\_df$Ozone)

summary(New\_df)

**2 DATA TRANSFORMATIONS IN R**

Data transformations in R are generally carried out using external packages tidyverse and dplyr in R.

***install.packages("tidyverse")***

***library(tidyverse)***

Note : you will need to install the package only once but need to load the package every time you start your environment

***2.1 Using arrange() function***

Arrange rows by variables

***data(mpg)***

***view(mpg)***

First, we will order the observations as per the displ column. Values of this column will be arranged in ascending order by default and other columns will follow the order of the displ column.

***mpg\_arr = mpg %>% arrange(displ)***

***view(mpg\_arr)***

***2.2 Using select() function***

Select variables by name

***mpg\_ds = mpg %>% select(displ,cty)***

***mpg\_ds***

**2.3 Using Filter() function** filter()

Filter observations by conditions

***diamonds\_filter1 <- diamonds %>% filter(cut == 'Ideal')***

***View(diamonds\_filter1)***

**2.4 group\_by() & summarize()**

Group by variables & reduce multiple values down to a single value

These are very useful functions coupled together to summarize values into groups. Here we will use the msleep dataset. It contains sleep and body weight data for some of the mammals.

***?msleep***

***data(msleep)***

***colnames(msleep)***

***msleep <- msleep %>% select(name, order, sleep\_total, bodywt)***

***View(msleep)***

We will summarize sleep\_total to its average values- sleep\_avg and bodywt to its maximum values- bodywt\_max. This summarization will be grouped as per order and the number of each order observation will be under the count column. All these will be done by group\_by() and summarize() with mathematical functions- n(), mean() and max().

***msleep\_groupby\_summarize <- msleep %>% group\_by(order) %>%***

***summarise(***

***count = n(),***

***sleep\_avg = mean(sleep\_total),***

***bodywt\_max = max(bodywt)***

***)***

***View(msleep\_groupby\_summarize)***

**3. Sampling Data in R**

A sample is a subset of data taken from a population. We take samples when entire data is not available or when we want to make inference about the entire population.

Sampling methods fall under two categories

1. Probabilistic Sampling
2. Non-Probabilistic Sampling

**Probabilistic Random Sampling is further classified as**

1. Simple Random Sampling
2. Systematic Sampling
3. Stratified Sampling
4. Cluster Sampling

**Simple Random Sampling**

In a simple random sampling, every case in the population has an equal probability of getting selected in the sample.

***sample(1:10,10)***

***sample(1:10, replace=T)***

We have to take samples out of red and green , but with priority to No

***sample(c("yes","no"),10,replace=T,prob=c(0.3,0.7))***

**Systematic Sampling**

Systematic sampling is used in situations where the population data is an ordered list or is arranged in time Using the below R code, from a population P which contains the units sold on all the days of the week over a period of 14 days, we have created a systematic sample with just the units sold on alternate days.

install.packages("TeachingSampling")

library(TeachingSampling)

P <- c("Mon-8", "Tues-4", "Wed-4", "Thurs-6", "Fri-7","Sat-45","Sun-34","Mon-21", "Tues-11","Wed-34","Thurs-16","Fri-10","Sat-17","Sun-19")

systematic\_sample <- S.SY(14,2)

systematic\_sample

P[systematic\_sample]

**4. Test and Training Splits**

The sample() method in base R is used to take a specified size data set as input. The data set may be a vector, matrix or a data frame. This method then extracts a sample from the specified data set. The sample chosen contains elements of a specified size from the data set which can be either chosen with or without replacement.

# creating the data set

mat = matrix(c(1:21), nrow = 7, ncol = 3, byrow = TRUE)

print ("Dataset")

print (mat)

# divide the matrix into training set 70% and

# testing 30% respectively with replacement

sample <- sample(c(TRUE,FALSE), nrow(mat),replace=TRUE, prob=c(0.7,0.3))

# creating training dataset

train\_dataset <- mat[sample, ]

# creating testing dataset

test\_dataset <- mat[!sample, ]

print("Training Dataset")

print (train\_dataset)

print("Testing Dataset")

print (test\_dataset)

**5. Creating Sample Groups**

# importing required libraries

library("plyr")

# create dataframe

data\_frame<-data.frame(col1=c(rep('G1',50),rep('G2',50),rep('G3',50)),

col2=rep(letters[1:5],30)

)

print("Original DataFrame")

head(data\_frame)

# pick 3 samples of each from data frame

data\_mod <- ddply(data\_frame,.(col1),function(x) x[sample(nrow(x),5),])

print("Modified DataFrame")

print (data\_mod)