CH:3 DATA PREPARATION AND OTHER TRICKS

HOW TO REMOVE PERCENTAGE SIGN FROM DATA OF COLUMN

Syntax: gsub(pattern, replacement, string)

Example: StudentData$result = gsub("%","",StudentData$result)

REMOVE DOLLAR SIGN

```
df1 <- data.frame(ID=1:5,sales=c('$14.45', '$13.39', '$17.89',
'$18.99', '$20.88’))
```

```
df1$sales = gsub(“\\$", "", df1$sales)
```

OR

```
df1$sales = as.numeric(gsub(“\\$", "", df1$sales))
```

To convert the data into numeric type

Because the data in vector is in string format

sapply() Function

- applies a certain operation to all the elements of the object
- sample_data<- data.frame(x=c(1,2,3,4,5,6), y=c(3,2,4,2,34,5))
- sapply(sample_data, max)

apply() function: takes matrix or data frame as an argument along with function and whether it has to be applied by row or column and returns the result in the form of a vector or array or list of values obtained.

Syntax:

apply(x, margin, function)

x: determines the input array including matrix.

margin: If the margin is 1 function is applied across row, if the margin is 2 it is applied across the column.

function: determines the function that is to be applied on input data.

Example:

```
sample_matrix <- matrix(C<-(1:10),nrow=3, ncol=10)
apply( sample_matrix, 1, sum)
apply( sample_matrix, 2, max)
```

lapply() function: a list, vector, or data frame as input and gives output in the form of a list object with specified function.

```
names <- c("abc", "def","pqr")
lapply(names, toupper)
```

Use of aggregate() Function

- **Use for fetching grouping data**
- **Example:**

```
data = data.frame(subjects=c("java", "python", "java",
                             "java", "php", "php"),
                  id=c(1, 2, 3, 4, 5, 6),
                  names=c(“ram", “shyam", "mohan",
                          “meera", “jeet", “het"),
                  marks=c(89, 89, 76, 89, 90, 67))

print(aggregate(data$marks, list(data$subjects),
FUN=sum))
```

We can use min or max or mean function also

FUN represents sum/mean/min/ max.

DISPLAYING R OBJECTS

- **head(x,n) : used to view first n observations**
- **tail (x,n) : used to view last n observations of dataset**
- **str(x) : horizontally display the internal structure of dataset**
- **fix(x): Used to display dataset in new window**
- **View(x): Used to view the dataset**

Note: You can use available dataset as X like airquality, mtcars

WORKING WITH TIME AND DATE

- **Sys.time()**
- **as.numeric(Sys.time())**
- **Sys.Date()**
- **month.abb**
- **month.name**

For more functionality for date and time add package: “lubridate”

Example:

```
curr_date = Sys.Date()

year(curr_date)

month(curr_date)

mday(curr_date)
```

Example:

```
dates<-c("2022-07-11","2012-04-19", "2017-03-08")

year(dates)
```

```
d1=as.date('5-5-24',format='%d-%m-%Y')

d1=as.date('5/5/24',format='%d/%m/%Y')

d2=d1+1

difftime(d1,d2)

difftime(d1,d2,units=c("Weeks")
```

TEXT MANIPULATION

grep() method : **used for pattern matching and replacement and return value or index according to parameter**

Syntax:

grep(pattern,x,ignore.case=TRUE/FALSE,value=TRUE/FALSE)

Example:

x=c("Rachana","Beena","rachana","Advika")

grep("Rachana",x)

grep("Rachana",x,ignore.case=TRUE)

grep("Rachana",x,ignore.case=TRUE,value=TRUE)

grep("^R",x) // Starting character should be R

grep("a$",x) //Ending character should be a

grepl() function: used for pattern matching and replacement and return true or false

Syntax:

```
grepl(pattern,x)
```

Example:

```
x=c("Rachana","Beena","rachana","Advika")
grepl("Rachana",x)
```

- **unlist(list): convert a list to vector**

- **nchar(x): count the characters of x object**

- **substring(x,start,stop): extract substrings in a character type**

  Example: substr("Rachana",1,4)

Paste(...,sep="",collapse=NULL): concatenate the two string values by separating with delimiters

Example:

```
paste('abc','def',sep='_')

paste(c('abc','def','pqr'),collapse='&')

paste(c('abc','def','pqr'),4,sep='_',collapse='&')
```

strptime() function: parse the given representation of date and time with the given template

Syntax: strptime(x, format, tz = “”)

x: given representation of date and time

format: given template in which parsing is done

tz: a character string specifying the time zone to be used for the conversion

Example:1

```
x <- "13:15:17"

y <- strptime(x, "% H:% M:% S")
```

Example:2

```
x <- "10-02-2020 05:05:06 AM"

y <- strptime(x, "% d-% m-% Y % H:% M:% S")
```

Thank You…