Some Functions for Advanced Data Handling in R

In this segment, we will discuss some functions in R which are frequently used for advanced data handling. These commands will be helpful for practical analysis as well as data cleaning. We will need the "ISwR" library.

```
library("ISwR")
attach(thuesen)
```

The “cut” function

The “cut” function divides the range of x into intervals and codes the values in x according to which interval they fall.

Let’s create a factor in which the “blood.glucose” variable in the “thuesen” data set will be divided into four intervals: (4, 7], (7, 9], (9, 12], and (12, 20]. The related code will be:

```
int <- cut(blood.glucose, c(4, 7, 9, 12, 20))
```

Now let’s change the level names to “low”, “intermediate”, “high”, and “very high”.

```
levels(int) <- c("low", "intermediate", "high", "very high")
```

The “sub” function

The “sub” function replaces the first match of a string, if the parameter is a string vector, replaces the first match of all elements. Let's use the "sub" function and replace the "Data Analytics: Basic Methods" to "Data Analytics: Advanced Methods"

```
x <- "Data Analytics: Basic Methods"
y <- sub("Basic","Advanced", x)
y
## [1] "Data Analytics: Advanced Methods"
```

The “gsub” function

The “gsub” function replaces all matches of a string, if the parameter is a string vector, returns a string vector of the same length and with the same attributes. Elements of string vectors which are not substituted will be returned unchanged.

```
x <- c("CIND 123: Spring Term CMTH 642: Spring Term")
gsub("Spring","Summer",x)
## [1] "CIND 123: Summer Term CMTH 642: Summer Term"
x
## [1] "CIND 123: Spring Term CMTH 642: Spring Term"
```

The “split” function:

The “split” function divides the data in the vector x into the groups defined by f. The replacement forms replace values corresponding to such a division. The “unsplit” function reverses the effect of split. Let's split the "energy" data set by the "stature" column.

```
split(energy,energy$stature)
## $lean
##    expend stature
## 2    7.53    lean
## 3    7.48    lean
## 4    8.08    lean
## 5    8.09    lean
## 6   10.15    lean
## 7    8.40    lean
## 8   10.88    lean
## 9    6.13    lean
## 10   7.90    lean
## 13   7.05    lean
## 16   7.48    lean
## 20   7.58    lean
## 22   8.11    lean
##
## $obese
##    expend stature
## 1    9.21   obese
## 11  11.51   obese
## 12  12.79   obese
## 14  11.85   obese
## 15   9.97   obese
## 17   8.79   obese
## 18   9.69   obese
## 19   9.68   obese
## 21   9.19   obese
```

And now, let's expand the "expend" column of the "energy" dataset by the "stature" column.

```
split(energy$expend,energy$stature)
## $lean
##  [1]  7.53  7.48  8.08  8.09 10.15  8.40 10.88  6.13  7.90  7.05  7.48
## [12]  7.58  8.11
##
## $obese
## [1]  9.21 11.51 12.79 11.85  9.97  8.79  9.69  9.68  9.19
```

The “merge” function

The “merge” function will merge the two data frames by common columns or row names.

Let's create two data frames with one common column indicating "ID".

```
data.frame.A<-data.frame(ID=1:4,
                         Gender=c("f", "f", "m", "m"),
                         Age=c(30, 24, 26, 18))
data.frame.B<-data.frame(ID=1:4,
                         Weight=c(60, 54, 70, 76),
                         Height=c(160, 170, 172, 186))
```

Let's write a code to join these two data frames by their common ID column.

```
total <- merge(data.frame.A, data.frame.B, by="ID")
```