Factors

R uses the *factor* data type to represent categorical data. The possible values of a factor are called *levels*. For example: suppose a variable `sex` can assume values `male` or `female`. Then this variable should be encoded in R as a factor `sex` with two levels, `male` and `female`.

- Sometimes people confuse the factor type with the character type. Characters are often used for labels in graphs, column names or row names. Factors must be used when you want to represent a categorical variable and want to analyze it. Factor objects can be created from character objects or from numeric objects, using the function **`factor()`**. For example, to create a vector of length five of type factor, do the following:

```
> sex <- c("male","male","female","male","female") # Character type
> sex
```

```
[1] "male"   "male"   "female" "male"   "female"
```

The object `sex` is now a character object. You need to transform it to factor:

```
> sex <- factor(sex)
> sex
```

```
[1] male   male   female male   female
Levels: female male
```

Use the function **`levels`** to see the different levels of a factor variable:

```
> levels(sex)
```

```
[1] "female" "male"
```

Note that the result of the levels function is of type character.

- Another way to generate the `sex` variable is as follows:

```
> sex <- c(1,1,2,1,2)
> sex
```

```
[1] 1 1 2 1 2
```

Now, the object `sex` is an integer variable. To use it as a categorical variable, you need to transform it to a factor:

```
> sex <- factor(sex)
> sex
```

```
[1] 1 1 2 1 2
Levels: 1 2
```

Note that now the R output states that the vector is a factor with two levels, 1 and 2. You could rename the levels, such that level "1" becomes "male" and level "2" becomes "female":

```
> levels(sex) <- c("male","female")
> sex
```

```
[1] male   male   female male   female
Levels: male female
```

- A *very* useful R function that takes some time to get used to is **`tapply()`**. In many cases, your data consist of measurement on individuals/cases that belong to different groups/treatments/levels. Technically, you'll have a vector containing all measurements, and a vector of type factor that specifies to which group the case belongs. The function **`tapply()`** allows you to apply a function, such as **`mean()`** or **`sd()`**, to you data, similarly to the function **`apply()`**. But the advantage of **`tapply()`** is that it will apply the function separately to each group/level.

  An example may help. Suppose we have obtained the age of 5 students:

```
> age <- c(18, 25, 23, 29, 37)
```

  Their respective genders are given by the `sex` variable you defined above:

```
> sex
```

```
[1] male   male   female male   female
Levels: male female
```

  Now, we would like to know the mean age of the males and the mean age of the women. This can be obtained in one go:

```
> (meanAge <- tapply(age, sex, mean))
```

```
  male female
    24     30
```

  The mean age of the males is 24, the mean age of the females is 30.