# practical_olivo

Olivo, Mel Adry

2024-03-06

A. Load the built-in warpbreaks dataset.

```
data("warpbreaks")

#View(warpbreaks)
```

1. Find out, in a single command, which columns of warpbreaks are either numeric or integer. What are the data types of each column?

```
str(warpbreaks)
```

```
## 'data.frame':    54 obs. of  3 variables:
##  $ breaks : num  26 30 54 25 70 52 51 26 67 18 ...
##  $ wool   : Factor w/ 2 levels "A","B": 1 1 1 1 1 1 1 1 1 1 ...
##  $ tension: Factor w/ 3 levels "L","M","H": 1 1 1 1 1 1 1 1 1 2 ...
```

```
## The breaks column is a numeric data type
## The wool and tension column are factored characters
```

2. How many observations does it have?

```
(numOfObservations <- nrow(warpbreaks))
```

```
## [1] 54
```

```
## There are 54 observations
```

3. Is numeric a natural data type for the columns which are stored as such? Convert to integer when necessary.

```
is.integer(warpbreaks$breaks) # returns TRUE because breaks column is already an integer
```

```
## [1] FALSE
```

```
warpbreaks$wool <- as.integer(warpbreaks$wool)
warpbreaks$tension <- as.integer(warpbreaks$tension)

## wool and tension column is converted to integer

warpbreaks
```

```
##    breaks wool tension
## 1      26    1       1
## 2      30    1       1
## 3      54    1       1
## 4      25    1       1
## 5      70    1       1
## 6      52    1       1
```

```
## 7      51    1       1
## 8      26    1       1
## 9      67    1       1
## 10     18    1       2
## 11     21    1       2
## 12     29    1       2
## 13     17    1       2
## 14     12    1       2
## 15     18    1       2
## 16     35    1       2
## 17     30    1       2
## 18     36    1       2
## 19     36    1       3
## 20     21    1       3
## 21     24    1       3
## 22     18    1       3
## 23     10    1       3
## 24     43    1       3
## 25     28    1       3
## 26     15    1       3
## 27     26    1       3
## 28     27    2       1
## 29     14    2       1
## 30     29    2       1
## 31     19    2       1
## 32     29    2       1
## 33     31    2       1
## 34     41    2       1
## 35     20    2       1
## 36     44    2       1
## 37     42    2       2
## 38     26    2       2
## 39     19    2       2
## 40     16    2       2
## 41     39    2       2
## 42     28    2       2
## 43     21    2       2
## 44     39    2       2
## 45     29    2       2
## 46     20    2       3
## 47     21    2       3
## 48     24    2       3
## 49     17    2       3
## 50     13    2       3
## 51     15    2       3
## 52     15    2       3
## 53     16    2       3
## 54     28    2       3
```

4. Error messages in R sometimes report the underlying type of an object rather than the user-level class. Derive from the following code and error message what the underlying type. Explain what is the error all about. Do not just copy the error message that was displayed.

```
## A Factor variable are stored internally as integers wihch is the underlying type.
## Doing operations on such a factor would lead to an error mentioning "integer"
```

```
## despite the user treating it as a character.


## R displays the underlying type like integer or numeric instead
## of the user-level class like factors or characters.


## If necessary, convert data to the appropriate type using functions like as.integer().
```

B. Load the exampleFile.txt

1. Read the complete file using readLines.

```
(exampleFile <- readLines("exampleFile.txt"))
```

```
## Warning in readLines("exampleFile.txt"): incomplete final line found on
## 'exampleFile.txt'
```

```
## [1] "// Survey data. Created : 21 May 2013"
## [2] "// Field 1: Gender"
## [3] "// Field 2: Age (in years)"
## [4] "// Field 3: Weight (in kg)"
## [5] "M;28;81.3"
## [6] "male;45;"
## [7] "Female;17;57,2"
## [8] "fem.;64;62.8"
```

2. Separate the vector of lines into a vector containing comments and a vector containing the data. Hint: use grepl.

```
## separating lines with "//" to and put to a vector
(comments <- exampleFile[grepl("//", exampleFile)])
```

```
## [1] "// Survey data. Created : 21 May 2013"
## [2] "// Field 1: Gender"
## [3] "// Field 2: Age (in years)"
## [4] "// Field 3: Weight (in kg)"
```

```
## separating lines WITHOUT "//" and put to a vector
(data <- exampleFile[!grepl("//", exampleFile)])
```

```
## [1] "M;28;81.3"      "male;45;"       "Female;17;57,2" "fem.;64;62.8"
```

3. Extract the date from the first comment line and display on the screen "It was created data."

```
## Using gsub to replace from "Created : " and everything behind it to blank ""

(date <- gsub(".*Created : ", "", comments[1]))
```

```
## [1] "21 May 2013"
```

```
paste0("It was created ", date)
```

```
## [1] "It was created 21 May 2013"
```

4. Read the data into a matrix as follows.

a. Split the character vectors in the vector containing data lines by semicolon (;) using strsplit.

```
(splitLines <- strsplit(data, ";"))
```

```
## [[1]]
## [1] "M"    "28"   "81.3"
##
## [[2]]
## [1] "male" "45"
##
## [[3]]
## [1] "Female" "17"     "57,2"
##
## [[4]]
## [1] "fem." "64"   "62.8"
```

b. Find the maximum number of fields retrieved by split. Append rows that are shorter with NA's.

```
## Get lengths of each rows
(lengthVectors <- sapply(splitLines, length))
```

```
## [1] 3 2 3 3
```

```
## Find the max length from each vectors
(maxFields <- max(lengthVectors))
```

```
## [1] 3
```

```
## Applying a function to each rows in splitLines list by using lapply()
## Combine NA to shorter rows using c()
(splitLines <- lapply(splitLines, function(row) c(row, rep(NA, maxFields - length(row)))))
```

```
## [[1]]
## [1] "M"    "28"   "81.3"
##
## [[2]]
## [1] "male" "45"   NA
##
## [[3]]
## [1] "Female" "17"     "57,2"
##
## [[4]]
## [1] "fem." "64"   "62.8"
```

c. Use unlist and matrix to transform the data to row-column format.

```
(unlistedLines <- unlist(splitLines))
```

```
##  [1] "M"      "28"     "81.3"   "male"   "45"     NA       "Female" "17"
##  [9] "57,2"   "fem."   "64"     "62.8"
```

```
## byrow = TRUE so matrix would be filled horizontally or by rows and not by columns.
(dataMatrix <- matrix(unlistedLines, nrow = length(data), byrow = TRUE))
```

```
##      [,1]     [,2] [,3]
## [1,] "M"      "28" "81.3"
## [2,] "male"   "45" NA
## [3,] "Female" "17" "57,2"
## [4,] "fem."   "64" "62.8"
```

d. From comment lines 2-4, extract the names of the fields. Set these as colnames for the matrix you just created.

```r
## Using gsub to replace everything from the start up to ": " to blank ""
(fields <- gsub(".*: ", "", comments[2:4]))
```

```
## [1] "Gender"         "Age (in years)" "Weight (in kg)"
```

```r
## Changing colnames
(colnames(dataMatrix) <- fields)
```

```
## [1] "Gender"         "Age (in years)" "Weight (in kg)"
```

```r
## The final matrix
dataMatrix
```

```
##      Gender   Age (in years) Weight (in kg)
## [1,] "M"      "28"           "81.3"
## [2,] "male"   "45"           NA
## [3,] "Female" "17"           "57,2"
## [4,] "fem."   "64"           "62.8"
```

^ ^ THANK YOU ^ ^