Pract_Exam_Gallenero

2024-03-06

#A Load the built-warpbreaks dataset

```
#Load the warpbreaks dataset
data ("warpbreaks")
```

#1. Find out, in a single command, which columns of warpbreaks are either numeric or integer.

```
Numeric_cols <- sapply(warpbreaks, is.numeric)
Numeric_cols
```

```
##  breaks    wool tension
##    TRUE   FALSE   FALSE
```

```
#2. Is numeric a natural data type for the columns which are stored as such? Convert tointeger when nec
Integer_cols <- sapply(warpbreaks, is.integer)
Integer_cols
```

```
##  breaks    wool tension
##   FALSE   FALSE   FALSE
```

```
numeric_or_integer_cols <- warpbreaks[, Numeric_cols| Integer_cols]
numeric_or_integer_cols
```

```
##  [1] 26 30 54 25 70 52 51 26 67 18 21 29 17 12 18 35 30 36 36 21 24 18 10 43 28
## [26] 15 26 27 14 29 19 29 31 41 20 44 42 26 19 16 39 28 21 39 29 20 21 24 17 13
## [51] 15 15 16 28
```

#Error messages in R sometimes report the underlying type of an object rather than theuser-level class. Derive from the following code and error message what theunderlying type.

```
#4 ERROR MESSAGE
#Error in 1:ncol(numeric_or_integer_columns) : argument of length 0
```

#B. Load the example.File.txt

```
#Read the complete file using readLines.
lines <- readLines("exampleFile.txt")
```

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

```r
#Separate the vector of lines into a vector containing comments and a vector containing the data. Hint:
comments <- lines[grepl("^//", lines)]
comments
```

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

```r
data_lines <- lines[!grepl("^//", lines)]
data_lines
```

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

```r
#Extract the date from the first comment line.

date <- gsub("^// Survey data. Created : ", "", comments[1])
date
```

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

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

```r
split_data <- strsplit(data_lines, ";")
split_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.

```r
maximum_fields <- max(sapply(split_data, length))
maximum_fields
```

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

```r
split_data <- lapply(split_data, function(x) c(x, rep(NA, maximum_fields - length(x))))
split_data
```

```
## [[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.

```
data_matrix <- matrix(unlist(split_data), ncol = maximum_fields, byrow = TRUE)
data_matrix
```

```
##      [,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.

```
field_Names <- gsub("^// Field [0-9]+: ", "", comments[2:4])
field_Names
```

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

```
colnames(data_matrix) <- field_Names
colnames(data_matrix)
```

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