PRACTICAL EXAM

Jirraine Octaviano

2024-03-07

A. Load the built-in warpbreaks dataset.

```
data ("warpbreaks")
```

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

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

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

```
# -The data types in each column are breaks, wool, and tension.
```

2. How many observations does it have?

```
observations <- nrow(warpbreaks)
observations
```

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

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

```
integer_columns <- sapply(warpbreaks, is.integer)
integer_columns
```

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

```
numeric_or_integer_columns <- warpbreaks[, numeric_columns | integer_columns]
numeric_or_integer_columns
```

```
##  [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
```

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.

```
## Factor variables are internally stored as integers, which represent the underlying type.
## The operation on such a factor may result in an error mentioning "integer."
## Despite being treated as a character by the user.


## R displays the underlying type, such as integer or numeric, instead of the user-level class, like fa
```

```
## Use functions like as.integer() to convert data to the appropriate type if necessary.
```

B. Load the exampleFile.txt

```
data <- read.csv("exampleFile.txt")
```

1. Read the complete file using readLines.

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

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

```
cat(lines, sep = "\n")
```

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

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

```
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)"
```

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

```
## [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."

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

```
## [1] "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.

```
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.

```
max_fields <- max(sapply(split_data, length))
max_fields
```

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

```
split_data <- lapply(split_data, function(x) c(x, rep(NA, max_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 = max_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.

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

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

```
colnames(data_matrix) <- fieldNames
colnames(data_matrix)
```

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