# Practical Exam Bagilidad

Bagilidad Ace

2024-03-07

Loading Warpbreak dataset

```
data("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? #ans. #The breaks columns of warpbreaks are numeric. #The wool columns of warpbreaks are factor. #The tension columns of warpbreaks are factor.

```
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 ...
```

2. How many observations does it have? #ans. There are total of 54 observations.

```
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. Ans. Yes, numeric is a natural data type for columns

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


str(warpbreaks)
```

```
## 'data.frame':    54 obs. of  3 variables:
##  $ breaks : num  26 30 54 25 70 52 51 26 67 18 ...
##  $ wool   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ tension: int  1 1 1 1 1 1 1 1 1 2 ...
```

```
#View(warpbreaks)
```

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.

```
# When R returns an object's underlying type rather than its user-level class, it indicates that the ex
```

B. Load the 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'
```

```
lines
```

```
## [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.

```
# Separate the lines into comments and data
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[!grepl("//", lines)]
data
```

```
## [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.”

```r
date_line <- lines[grepl("^//", lines)]
date_str <- gsub("^//|Survey data. Created : ", "", date_line[1])

print(paste("It was created", date_str))
```

```
## [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.

```r
list <- strsplit(data, split = ";")
list
```

```
## [[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
#Find the maximum number of fields retrieved by split
max_fields <- max(sapply(list, length))
max_fields
```

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

```r
cat("The maximum number of fields retrieved by split are: ",max_fields)
```

```
## The maximum number of fields retrieved by split are:  3
```

```r
#Append rows that are shorter with NA's.
splitList<-lapply(list,function(row) c(row, rep(NA,max_fields-length(row))))
splitList
```

```
## [[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.

```
splitList_unlist<- unlist(splitList)
splitList_unlist
```

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

```
data_matrix <- matrix(splitList_unlist, nrow = length(data),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 <- c("Gender", "Age (in years) ", "Weight (in kg)")

# Set column names
colnames(data_matrix) <- field_names
print(data_matrix)
```

```
##      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"
```