ADS2: Problem Set 1 Notes

Dmytro Shytikov

2023-09-23

Importing data into R

Modifying the data frame can be done by many means. For example:

```
thymocytes <- rbind(thymocytes, c("KO", 13, 54, NA, NA),
                    c("KO", 15.54, 42, NA, NA),
                    c("WT", 8.67, 85.4, NA, NA),
                    c("WT", 7.87, 77.45, NA, NA),
                    c("WT", 9.5, 80.21, NA, NA))
tail(thymocytes)
```

```
##    genotype    DN    DP    SP Pathology
## 20       WT 10.69 78.82 10.49    Normal
## 21       KO    13    54  <NA>      <NA>
## 22       KO 15.54    42  <NA>      <NA>
## 23       WT  8.67  85.4  <NA>      <NA>
## 24       WT  7.87 77.45  <NA>      <NA>
## 25       WT   9.5 80.21  <NA>      <NA>
```

You can create a new matrix that contains all the data, but you will need to spend some time on reformatting this matrix to make it acceptable for combining with the original data frame `thymocytes` as matrices cannot be used to store diverse types of data (the data can be saved as either numbers or characters; no factors, no dates, no lifespan).

```
##      genotype DN      DP      SP Pathology
## [1,] "KO"     "13"    "54"    NA NA
## [2,] "KO"     "15.54" "42"    NA NA
## [3,] "WT"     "8.67"  "85.4"  NA NA
## [4,] "WT"     "7.87"  "77.45" NA NA
## [5,] "WT"     "9.5"   "80.21" NA NA
```

There are many other ways to load these new data into your existing data frame. You are free to find your own solution.

Another problem here is with the data that were supposed to be present, but are absent – they are `Not Available` or `NA` (remember, your colleagues forgot to record this information!). In this case, you must not replace these missing values with random numbers (unless you have very good reasoning), but rather mark these data as `NAs` (check the code!). This type of data is very important and there are numerous functions that allow you to operate with it. You will learn about it later in the course.

Data export can be done with the `write.csv()` command. Remember to save your data in `.csv` whenever possible as this format can be used by many other programs freely. Try the `xlsx` and `readxl` packages by yourself.

Importing and exporting `.csv` tables

This task is straightforward. You need to load an external file `ADS2week1.csv` into R using the `read.csv()` command.

Next, you will need to extract the text in the table either manually, or using a `for` loop (probably, you have already used this type of command structures in the IBI1 course), or any other way that suits you. Then, combine this text using commands that allow string operations. For example, the `paste()` command.

```
data <- read.csv("ADS2week1.csv", row.names = NULL)

head(data)
```

```
##   X column.1  column.2 column.3  column.4
## 1 1       It   Convert    Then,  Finally,
## 2 2       is        it   remove    export
## 3 3        a      into     NAs.       the
## 4 4  strange   several     <NA> resultant
## 5 5     data character     <NA>      text
## 6 6   frame!  strings!     <NA>        in
```

```
data <- data[,-1] # We do not need the first column

# Generally, you should use diverse names for your data. Otherwise, R will be confused.
# For instance, there is a special command `data()` in R.

PS1 <- c() # You create an empty vector where you will store your data

for(i in 1:ncol(data)){
  linE <- data[!is.na(data[,i]), i]
  # So, in each round, you choose one column from your data: `data[, i]`
  # and you choose only those rows that do not include NAs: `data[!is.na(data[, i]), ]`
  # As a result, you get a character vector.

  linE <- paste(linE, collapse = " ")
  # After this, you just combine the vector in a text string
  # You are welcome to try to beautify it further.

  linE <- trimws(linE)
  # This line will remove white spaces on the around the string.
  # For instance, the `paste(data, collapse = " ")` command will add " " on the right side
  # of your string.

  PS1[i] <- linE # So, you add your string into the PS1 vector you created earlier
}

writeLines(PS1, "ADS2_PS1.txt")
```

Exporting the resultant text may be done using the `writeLines()` command. Just check the command's syntax! e.g. using `?writeLines`.

The code above is neither the only possible nor the most optimal. So, you are free to experiment and offer your solutions.

Altogether, this week you have learned basic commands that allow you to create data, change these, convert from one type to another, and export. You will learn other methods of operating and cleaning your data later.

Originally, created by D Shytikov in 2022.

Last update by D Shytikov in 2023.