Human Activity Recognition Data preparation

Joan Camilo Mina

23-09-2020

humanActivityRecognitionCourseProject

This is a repo where i have include my proposed solution to the course´s project named getting and cleaning data in wich is used a data set related to human activity recognition using Samsung Galaxy SII sensors.

Task 1: Data Acquisition and merging

The first task of the script is creating url references to the location of the files, and then loading the data into 4 separated dataframes: **x_train, y_train, x_test, y_test** using read.fwf function with a width of 16 characters per variable getting a total of **10299 observation and 561 variables**. After this I proceed to merge the data using the **rbind** function over the pairs **x_train/x_test and y_train/y_test.**

Setting urls

```
x_training_set_url <-
  "./data/UCI HAR Dataset/train/X_train.txt"
y_training_set_url <-
  "./data/UCI HAR Dataset/train/y_train.txt"
train_subject_url <-"./data/UCI HAR Dataset/train/subject_train.txt"
x_test_set_url <-
  "./data/UCI HAR Dataset/test/X_test.txt"
y_test_set_url <-
  "./data/UCI HAR Dataset/test/y_test.txt"
test_subject_url <- "./data/UCI HAR Dataset/test/subject_test.txt"
variables_names_url <- "./data/UCI HAR Dataset/features.txt"
activities_names_url <- "./data/UCI HAR Dataset/activity_labels.txt"
```

Loading files into dataframes

```
#X training data
X_train_data <- read.fwf(x_training_set_url,widths=(rep.int(16,561)))
#Y training data
y_train_data <- read.fwf(y_training_set_url,widths=c(1))
#X Test Data
X_test_data <- read.fwf(x_test_set_url,widths=(rep.int(16,561)))
#Y Test Data
y_test_data <- read.fwf(y_test_set_url,widths=c(1))
print(dim(X_train_data))
```

Merging datasets

```
full_X_data <- rbind(X_train_data,X_test_data)
str(full_X_data)
full_y_data <- rbind(y_train_data,y_test_data)
str(full_y_data)
```

Task 2: Extract variables of interest

In this part of the data preparation, the **variables.txt** file was read into a dataframe **variables_names** and using the **grep** function and **regular expressions** were extracted the indexes **(mean_std_variables)** of those **features related to mean() and std()** calculus. All this was saved into a new dataframe called **X_mean_std_data**.

```
#Load the Variables´s Names into a DF
variables_names <- read.csv(variables_names_url, sep=" ", header=FALSE)
names(variables_names) <- c("index","varname")
variables_names$varname <- tolower(variables_names$varname)


#Search for variables's indexes with mean() or std() expressions
mean_std_variables <- grep("mean()|std()", variables_names$varname)
#Use indexes to extract only those variables related to mean() and std()
X_mean_std_data <- full_X_data[ , mean_std_variables]
```

Task 3: Use descriptive activity names

In this preparation´s step the **activity_labels.txt** file was loaded into a dataframe **activities_labels** with the later porpose of using that dataframe to convert the **y_full_data´s single variable** to a factor using the indexes as levels and the activities´s descriptions as labels.

```
names(full_y_data) <- c("activity")
activities_labels <- read.csv(activities_names_url,sep=" ",header=FALSE)
names(activities_labels) <- c("level","label")
full_y_data$activity <- factor(full_y_data$activity,levels=activities_labels$level,
                               labels=activities_labels$label)
```

Task 4: Setting appropiate variables´s names

Now the variables´s names for the **X_mean_std_data** are being taken from the dataframe **variables_names** using the indexes included in **mean_std_variables**.

```
names(X_mean_std_data) <- variables_names[variables_names$index %in% mean_std_variables, "varname"]
```

Task 5: Average grouping by activity and subject

In this part of the process was necessary to implement the following actions:

1. Loading subjects data from train and test sets into **full_subject** dataframe
2. Combine **X_mean_std_data** with both **subject and activity** dataframes

3. Group the resulting data by activity and subject using **group__by__at** getting a new dataframe named **by__activity__subject**
4. Calculate the mean of each variable using the **summarise__at** function, having in mind the indexes vector previously used wich has the variables of interest **mean__std__variables**

Loading subject data and combining X__mean__std__data with subject and activity data

```
library(dplyr)

train_subject <- read.csv(train_subject_url,header=FALSE)
test_subject <- read.csv(test_subject_url,header=FALSE)
full_subject <- rbind(train_subject,test_subject)

#Binds subject data to X_mean_std_data and sets a name for the new column
X_mean_std_data <- cbind(X_mean_std_data, full_subject)
names(X_mean_std_data)[names(X_mean_std_data)=="V1"]<-"subject"

#Binds activity data to X_mean_std_data and sets a name for the new column
X_mean_std_data <- cbind(X_mean_std_data, full_y_data)
names(X_mean_std_data)[names(X_mean_std_data)=="full_y_data"]<-"activity"
```

Grouping by activity and subject and calculating the mean for every variable

```
#Groups data by activity and subject
by_activity_subject <- group_by_at(X_mean_std_data, vars("activity","subject"))

#Calculates The mean of each feature using the groups created in prev line
mean_by_activity_subject <-
  summarise_at(by_activity_subject,variables_names$varname[mean_std_variables],
               mean, na.rm=TRUE)
```