run_analysis

Daniel Escasa

2021-01-28

```
knitr::opts_chunk$set(echo = TRUE)
```

Introduction

The run_analysis.R script downloads the Human Activity Recognition dataset, tidies it up, and performs various required data manipulations and summaries.

1 Boring admin stuff

1.1 Install dplyr if necessary

```
if (!require("dplyr")) {
    message("Installing dplyr")
    install.packages("dplyr")
}
```

```
## Loading required package: dplyr
```

```
## Warning: As of rlang 0.4.0, dplyr must be at least version 0.8.0.
## * dplyr 0.7.6 is too old for rlang 0.4.10.
## * Please update dplyr with `install.packages("dplyr")` and restart R.
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

1.2 Make sure I'm in the right directory

```
setwd("/home/daniel/Documents/Coursera/DataCleaning.JH/")
utils::sessionInfo()[2]
```

```
## $platform
## [1] "x86_64-pc-linux-gnu (64-bit)"
```

1.3 Download Human Activity Report dataset if necessary

```
if (!file.exists("./UCI_HAR_Dataset.zip")) {
    message("Downloading dataset")
    download.file("https://d396qusza40orc.cloudfront.net/getdata%2Fprojectfiles%2FUCI%20
HAR%20Dataset.zip",
                  destfile = "./UCI_HAR_Dataset.zip",
                  method = "internal",
                  mode = "wb")
}
```

1.4 Extract Human Activity Report dataset if necessary

I've decided to extract it to my DataCleaning.JH directory instead of a DATA directory.

```
if (!file.exists("./UCI HAR Dataset")) {
    message("Extracting dataset")
    unzip("./UCI_HAR_Dataset.zip",
          overwrite = FALSE)
}
```

2 Load the features

First line (`tibble:as_tibbl()`) reads in `features.txt` and coerces it as a data frame.

2.1 Search for the mean and standard deviations

The first two calls to `mutate()` do this.

2.2 Create syntactically valid names in a new features column named `Features`

2.3 Fix variable names from the features data frame

Succeeding `mutate()` function invocations do this. One aim is enhanced readability, e.g., `Time.BodyGyro-arCoeff()-X,2` instead of `tBodyGyro-arCoeff()-X,2` . More precisely, `mutate()` will accomplish the following:

1. determine the features that signify means and standard deviations.

2. replace the prefixes `t` and `f` with more descriptive `Time` and `Frequency`, respectively
3. replace “angle” with “Angle” for consistency in capitalization
4. replace double occurrences of “Body” — i.e., “BodyBody” — with “Body”
5. separate “Acc”, “Gyro”, “Jerk”, and “Mag” from the rest of the feature with dots — i.e., “.Acc”, “.Gyro”, “.Jerk”, and “.Mag”
6. replace double occurrences of the dot (“..”) with a single dot
7. remove a dot at the end of the feature.

```
features <- tibble::as_tibble(read.table("./UCI HAR Dataset/features.txt",
                              col.names = c("Id", "Feature")))

features <- features %>%
    mutate(Is.Mean          = grepl("mean\\(\\)", features$Feature)) %>%
    mutate(Is.Std           = grepl("std\\(\\)", features$Feature)) %>%
    mutate(Feature.Variable = make.names(features$Feature, unique = TRUE)) %>%
    mutate(Feature.Variable = gsub("^t", "Time.", Feature.Variable)) %>%
    mutate(Feature.Variable = gsub("\\.t", ".Time.", Feature.Variable)) %>%
    mutate(Feature.Variable = gsub("^f", "Frequency.", Feature.Variable)) %>%
    mutate(Feature.Variable = gsub("\\.f", ".Frequency.", Feature.Variable)) %>%
    mutate(Feature.Variable = gsub("^angle\\.", "Angle.", Feature.Variable)) %>%
    mutate(Feature.Variable = gsub("BodyBody", "Body", Feature.Variable)) %>%
    mutate(Feature.Variable = gsub("Acc", ".Acc", Feature.Variable)) %>%
    mutate(Feature.Variable = gsub("Gyro", ".Gyro", Feature.Variable)) %>%
    mutate(Feature.Variable = gsub("Jerk", ".Jerk", Feature.Variable)) %>%
    mutate(Feature.Variable = gsub("Mag", ".Mag", Feature.Variable)) %>%
    mutate(Feature.Variable = gsub("\\.\\.", ".", Feature.Variable)) %>%
    mutate(Feature.Variable = gsub("\\.\\.", ".", Feature.Variable)) %>%
    mutate(Feature.Variable = gsub("\\.$", "", Feature.Variable)) %>%
    mutate(Feature.Variable = gsub("(^|[\\.])([[:alpha:]])", "\\1\\U\\2",
                    Feature.Variable, perl=TRUE))
```

```
## Warning: The `printer` argument is deprecated as of rlang 0.3.0.
## This warning is displayed once per session.
```

mutate() rocks!

3 Load activities

As in the previous chunk, `tibble:as_tibbl(read.table())` reads in `labels.txt` and coerces it as a data frame. The data frame gets assigned column names “Id” and “Activity”

```
activities <- tibble::as_tibble(read.table("./UCI HAR Dataset/activity_labels.txt",
                col.names = c("Id", "Activity")))
```

4 Load the training dataset

Again, coerce `X_train.txt` as a data frame.

4.1 Assign `features` dataset to `train` column names

4.2 Add subject data and activity data to the training dataset

```
train <- tibble::as_tibble(read.table("./UCI HAR Dataset/train/X_train.txt"))
colnames(train) <- features$Feature.Variable
train <- cbind(
        rename(tibble::as_tibble(read.table("./UCI HAR Dataset/train/subject_train.txt"
)),
               Subject.Id = V1),
        rename(tibble::as_tibble(read.table("./UCI HAR Dataset/train/y_train.txt")),
               Activity.Id = V1),
        Dataset.Partition = c("Training"),
        train)
```

5 Load the test dataset

5.1 Assign `features` dataset to `test` column names

Add subject data, and activity data to the test dataset

Mostly the same as the previous chunk, applied to the `test` dataset

```
test <- tibble::as_tibble(read.table("./UCI HAR Dataset/test/X_test.txt"))
colnames(test) <- features$Feature.Variable
test <- cbind(
    rename(tibble::as_tibble(read.table("./UCI HAR Dataset/test/subject_test.txt")),
           Subject.Id = V1),
    rename(tibble::as_tibble(read.table("./UCI HAR Dataset/test/y_test.txt")),
           Activity.Id = V1),
    Dataset.Partition = c("Test"),
    test)
```

6 Merge the training and test datasets.

6.1 Add descriptive activity names from activities.

6.2 Select the mean and std deviation features only.

6.3 Group by subject and activity.

```
merged <- rbind(train, test) %>%
    left_join(activities, by = c("Activity.Id" = "Id")) %>%
    select(Subject.Id, Activity,
           one_of(
               filter(features, Is.Mean == TRUE | Is.Std == TRUE) %>%
                   select(Feature.Variable) %>% .[["Feature.Variable"]])) %>%
    group_by(Subject.Id, Activity)
```

```
## Warning: `as_data_frame()` is deprecated as of tibble 2.0.0.
## Please use `as_tibble()` instead.
## The signature and semantics have changed, see `?as_tibble`.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
## Warning: `chr_along()` is deprecated as of rlang 0.2.0.
## This warning is displayed once per session.
```

7 Create a tidy summary of feature means grouped by subject and activity.

```
tidy_summary <- summarise_each(merged, funs(mean))
```

```
## `summarise_each()` is deprecated.
## Use `summarise_all()`, `summarise_at()` or `summarise_if()` instead.
## To map `funs` over all variables, use `summarise_all()`
```

```
## Warning: `is_lang()` is deprecated as of rlang 0.2.0.
## Please use `is_call()` instead.
## This warning is displayed once per session.
```

```
## Warning: `lang()` is deprecated as of rlang 0.2.0.
## Please use `call2()` instead.
## This warning is displayed once per session.
```

```
## Warning: `mut_node_car()` is deprecated as of rlang 0.2.0.
## This warning is displayed once per session.
```

8 Write tidy summary to file.

```
write.table(tidy_summary, "tidy_summary.txt", row.names = FALSE)
```