Codebook for Getting and Cleaning Data Course Project

John Kenneth Velonta

2022-07-10

The first step was to load the features from the features.txt file, which contains the names of the features/variables that we will be utilizing in the assignment.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.8
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
features <- read.delim('./UCI HAR Dataset/features.txt', header=FALSE)
```

The next step was to convert the previously loaded features to a vector for further use.

```
features_char <- unlist(features[['V1']], use.names = FALSE)
```

Next, I loaded the subject data for the test and training datasets. Then I used rbind to merge the test and train subjects.

```
subject_test <- read.delim('./UCI HAR Dataset/test/subject_test.txt', header=FALSE, col.names = "Subjec
subject_train <- read.delim('./UCI HAR Dataset/train/subject_train.txt', header=FALSE, col.names = "Sub
merged_subjects <- rbind(subject_test, subject_train)
```

Next, I loaded the activities file, merged them through rbind, and changed from numeric to verbose activities through a left_join.

```
y_test <- read.delim('./UCI HAR Dataset/test/y_test.txt', header=FALSE, col.names = "Activity")
y_train <- read.delim('./UCI HAR Dataset/train/y_train.txt', header=FALSE, col.names = "Activity")
Merged_activities <- rbind(y_test,y_train)
activity_labels <- read.delim('./UCI HAR Dataset/activity_labels.txt', header=FALSE)
activity_labels <- cbind(activity_labels, do.call(rbind, strsplit(activity_labels$V1, "( +)")))
names(activity_labels) = c('V1', 'Activity', 'V2')
activity_labels <- mutate(activity_labels, Activity= as.numeric(activity_labels$Activity))
Merged_activities <- left_join(Merged_activities, activity_labels, by=('Activity'))
Merged_activities <- Merged_activities[3]
names(Merged_activities) <- 'Activity'
```

Next, the test and training datasets were loaded and merged through rbind.

```
X_test <- read.delim('./UCI HAR Dataset/test/X_test.txt', header=FALSE)
X_train <- read.delim('./UCI HAR Dataset/train/X_train.txt', header=FALSE)
Merged_data <- rbind(X_test,X_train)
```

I used a combination of strsplit, rbind, and cbind, to separate each feature from each observation to different columns.

```
Merged_data_split <- cbind(Merged_data, do.call(rbind, strsplit(Merged_data$V1, "( +)")))
```

Then I had to deselect columns 1 and 2.

```
Merged_data_split <- Merged_data_split[c(3:563)]
```

Then selected the columns representing the means and standard deviations with the use of grep. Any column name with mean or std on them were selected.

```
Merged_data_split[] <- sapply(Merged_data_split, as.numeric)
names(Merged_data_split) <- features_char
mean_stdev_loc <- grep("mean|std",names(Merged_data_split))
Merged_data_mean_stdev <- Merged_data_split[mean_stdev_loc]
```

Then, for tidying the feature names, f were changed to frequency, t were changed to time, and the numbers parentheses were removed through sub.

```
names(Merged_data_mean_stdev) <- sub("[0-9]*[ ]*f","frequency",names(Merged_data_mean_stdev))
names(Merged_data_mean_stdev) <- sub("[0-9]*[ ]*t","time",names(Merged_data_mean_stdev))
names(Merged_data_mean_stdev) <- sub("[0-9]*[ ]*","",names(Merged_data_mean_stdev))
names(Merged_data_mean_stdev) <- sub("\\(\\)","",names(Merged_data_mean_stdev))
names(Merged_data_mean_stdev) <- tolower(names(Merged_data_mean_stdev))
```

I then merged the datasets, activities, and subjects through cbind.

```
merged_all <- cbind(merged_subjects, Merged_activities, Merged_data_mean_stdev)
```

The resulting merged dataframe was then grouped by subject and activity, and then summarized to get the mean for each grouping.

```
merged_all <- merged_all %>% group_by(Subject, Activity) %>%
              summarize(across(everything(),mean))
```

```
## `summarise()` has grouped output by 'Subject'. You can override using the
## `.groups` argument.
```

The resulting dataframe was then exported as a csv file through write_csv and write.table.

```
write_csv(merged_all, './Tidy_Data_Set.csv')
write.table(merged_all, file='./Tidy_Data_Set.txt', row.name=FALSE)
```

```
## # A tibble: 180 x 81
## # Groups:   Subject [30]
##    Subject Activity           `timebodyacc-m~` `timebodyacc-m~` `timebodyacc-m~`
##      <int> <chr>                         <dbl>            <dbl>            <dbl>
##  1       1 LAYING                        0.222         -0.0405           -0.113
##  2       1 SITTING                       0.261         -0.00131          -0.105
##  3       1 STANDING                      0.279         -0.0161           -0.111
##  4       1 WALKING                       0.277         -0.0174           -0.111
##  5       1 WALKING_DOWNSTAIRS            0.289         -0.00992          -0.108
##  6       1 WALKING_UPSTAIRS              0.255         -0.0240           -0.0973
##  7       2 LAYING                        0.281         -0.0182           -0.107
##  8       2 SITTING                       0.277         -0.0157           -0.109
##  9       2 STANDING                      0.278         -0.0184           -0.106
## 10       2 WALKING                       0.276         -0.0186           -0.106
## # ... with 170 more rows, and 76 more variables: `timebodyacc-std-x` <dbl>,
## #   `timebodyacc-std-y` <dbl>, `timebodyacc-std-z` <dbl>,
## #   `timegravityacc-mean-x` <dbl>, `timegravityacc-mean-y` <dbl>,
## #   `timegravityacc-mean-z` <dbl>, `timegravityacc-std-x` <dbl>,
## #   `timegravityacc-std-y` <dbl>, `timegravityacc-std-z` <dbl>,
## #   `timebodyaccjerk-mean-x` <dbl>, `timebodyaccjerk-mean-y` <dbl>,
## #   `timebodyaccjerk-mean-z` <dbl>, `timebodyaccjerk-std-x` <dbl>, ...
```