```
# Input parameter should be either "train" or "test".
prepare_data <- function(dataset) {

    # Validate the dataset to be worked with.
    if(dataset %in% c("test", "train") == FALSE) {
        stop("Parameter \"dataset\" must be either \"test\" or \"train\".")
    }

    # Variable setup.
    dataset_folder <- "UCI HAR Dataset"
    subject_values <- NULL
    activity_values <- NULL
    main_data <- NULL

    # Read the relevant files.
    if(dataset == "test") {
        # dataset is "test".
        subject_values <- tbl_df(read.table(file.path(dataset_folder,
                                                      "test",
                                                      "subject_test.txt")))
        activity_values <- tbl_df(read.table(file.path(dataset_folder,
                                                       "test",
                                                       "y_test.txt")))
        main_data <- tbl_df(read.table(file.path(dataset_folder,
                                                 "test",
                                                 "X_test.txt")))
    } else {
        # dataset is "train".
        subject_values <- tbl_df(read.table(file.path(dataset_folder,
                                                      "train",
                                                      "subject_train.txt")))
        activity_values <- tbl_df(read.table(file.path(dataset_folder,
                                                       "train",
                                                       "y_train.txt")))
        main_data <- tbl_df(read.table(file.path(dataset_folder,
                                                 "train",
                                                 "X_train.txt")))
    }

    # Check for the correct number of unique subject values.
    if(dataset == "test") {
        # dataset is "test".
        if(subject_values %>% distinct() %>% count() != 9) {
            stop("The count of unique subjects is not the expected value of 9.")
        }
    } else {
        # dataset is "train".
        if(subject_values %>% distinct() %>% count() != 21) {
            stop("The count of unique subjects is not the expected value of 21."
                 )
        }
    }

    # Check that loaded datasets all have the same number of observations.
    if(!identical(nrow(subject_values), nrow(activity_values),
                  nrow(main_data))) {
        stop("Datasets have different count of observations.")
    }

    # Set column names for the subject and activity values datasets.
    colnames(subject_values) <- "subject"
    colnames(activity_values) <- "activity_id"

    # Check that the unique activity values are within the range 1 to 6.
    if(!all(distinct(activity_values)$activity_id %in% c(1,2,3,4,5,6))) {
        stop("The activity values are not within the range of 1 to 6.")
    }

    # Load the activity labels.
    activity_labels <- tbl_df(read.table(file.path(dataset_folder,
                                                   "activity_labels.txt")))

    # Set column names for the activity labels dataset.
    colnames(activity_labels) <- c("activity_id", "activity")

    # Join the activity values and labels datasets, then select only labels.
    activity_values <-
        activity_values %>%
        left_join(activity_labels, by = "activity_id") %>%
        select(activity)

    # Convert the activity column to a character vector.
    activity_values$activity <- as.character(activity_values$activity)

    # Join the subject and activity datasets.
    subjects_and_activities <- bind_cols(subject_values, activity_values)

    # Load the variable labels.
    variable_labels <- tbl_df(read.table(file.path(dataset_folder,
                                                   "features.txt")))
```

```
    # Set variables dataset column names.
    colnames(variable_labels) <- c("variable_id", "variable")

    # Check the number of variable labels (columns).
    if(!identical(nrow(variable_labels), ncol(main_data))) {
        stop("Variable count mismatch between labels and data.")
    }

    # Convert variable labels to character vector.
    variable_labels <- as.character(variable_labels$variable)

    valid_variable_labels <- make.names(variable_labels, unique = TRUE,
                                        allow_ = TRUE)

    # Set column main data column names.
    colnames(main_data) <- valid_variable_labels

    # Get a list of the variables we need to work with, based on matching the
    # list for ".mean.." and ".std..".
    selected_variables <- as.logical(
        grepl(".mean..", valid_variable_labels, fixed = TRUE) +
            grepl(".std..", valid_variable_labels, fixed = TRUE))

    # Create a vector of the column positions in the main data.
    variable_positions <- as.integer(1:length(selected_variables))

    # Create a vector with the column positions we need to select.
    selected_positions <- NULL
    for(i in variable_positions) {
        if(selected_variables[i] == TRUE) {
            selected_positions <- c(selected_positions, i)
        }
    }

    # Select only the required columns.
    main_data <- select(main_data, selected_positions)

    # Join the subject and activity dataset with the main dataset.
    main_data <- bind_cols(subjects_and_activities, main_data)

    # Return the dataset.
    return(main_data)

}
```