**R script – run\_analysis.r**

**Overview**

The task of the course project is to write a script in R called run\_analysis.r which, given a zipped file containing a folder structure with raw data in text format does the following:

1. Merges the training and the test sets to create one data set.
2. Extracts only the measurements on the mean and standard deviation for each measurement.
3. Uses descriptive activity names to name the activities in the data set.
4. Appropriately labels the data set with descriptive activity names.
5. Creates a second, independent tidy data set with the average of each variable for each activity and each subject.

This document will show the R scripts and provide commentary around how these scripts were used to perform each of the steps as listed above. In addition, the scripts used to produce the codebook are also given in the last section of this document.

1. **Assign libraries**

Assign the libraries which will be used in the scripts

library(plyr)

library(dplyr)

library(data.table)

library(reshape2)

1. **Download the file and unzip**

Assign the url, destination path and the name of the file in the destination path:

url <- "https://d396qusza40orc.cloudfront.net/getdata%2Fprojectfiles%2FUCI%20HAR%20Dataset.zip"

filename <- "UCI HAR Dataset.zip"

destpath <- "C:/Users/relazar/Documents/Personal/CourSera/Data Science/03 - Getting and Cleaning Data/Course Project/Data"

Check that the destination folder exists and if doesn’t then create it

if (!file.exists(destpath))

{

dir.create(destpath)

}

Download the file from the internet to the destination path

download.file(url, paste(destpath,"/",filename,sep=""))

Set the work directory as the destination folder and unzip the contents of the zip file into that same folder

setwd(destpath)

unzip(filename)

1. **Upload the text files into R**

In this step all the relevant text files from the master folder will be uploaded into R. for the purpose of this project we are disregarding the inertial signals and are only interested in the contents of the following text files:

1. activity\_labels.txt – the mapping between the activity numbers (1 to 6) and the labels of the activities (walking, walking upstairs etc,)
2. features.txt – the names of the features
3. subject\_train.txt – the subject ID outcome the training data set with the position corresponding to the actual outcome in the X\_train.txt data set
4. X\_train.txt – the actual measurements for all training subjects
5. y\_train.txt – the activity number (1 to 6) corresponding to each outcome in the X\_train.txt data set
6. subject\_test.txt txt – the subject ID outcome the test data set with the position corresponding to the actual outcome in the X\_test.txt data set
7. X\_test.txt – the actual measurements for all test subjects
8. y\_test.txt – the activity number (1 to 6) corresponding to each outcome in the X\_test.txt data set

First, set the work directory to be the master folder of the downloaded data structure from the zip file:

setwd(paste(getwd(),"/UCI HAR Dataset",sep=""))

We then create a function which will create a data frame with the following columns:

1. The full path of the text file
2. The name of the text file without the suffix - this will be the name of the r data set to be assigned
3. The type of the data set - training or test

The inputs of the function are:

1. type - indicating whether the first sub folder is "train" or "test"

find\_file\_names <- function(type)

{

## If the type is not train or test then leave the first sub folder as blank

if (type %in% c("train", "test") == FALSE)

{

subfolder1 <- ""

}

else

{

subfolder1 <- paste(type, "/", sep="")

}

# Assign the path name

pathname <- paste("./",subfolder1, sep="")

# Use the list.files function to list all files within the path name assigned

files <- list.files(path = pathname, pattern = NULL, all.files = TRUE, full.names = TRUE)

# Using the grepl function, return a list of only files which are text files

# i.e. end with the suffix ".txt"

# Exclude the features\_info and README files as these will not be read in

paths <- files[grepl(".txt$", files)==TRUE &

grepl("features\_info.txt$", files)==FALSE &

grepl("README.txt$", files)==FALSE]

# Get the name of each file without the .txt suffix

rname <- rep(NA, length(paths))

for (i in 1:length(paths))

{

lastslash <- regexpr("/[^/]\*$", paths[i])

lastdot <- regexpr("\\.[^\\.]\*$", paths[i])

rname[i] <- substr(paths[i],lastslash[1]+1, lastdot[1]-1)

}

# Create a data frame contain three columns:

# 1. The full path of the text file

# 2. The name of the text file without the suffix - this will be the name of the r data set to be assigned

# 3. The type of the data set - training or test

data.frame(pathname= paths, rdataname = rname, settype = rep(type, length(paths)))

}

We then apply this function to 3 folders: the master folder, the test folder and the train folder to get all the files we need.

masterFiles <- find\_file\_names("")

trainFiles <- find\_file\_names("train")

testFiles <- find\_file\_names("test")

Combine all the above 3 into the data frame which will contain all the information:

all\_file\_names <- rbind(masterFiles, trainFiles, testFiles)

all\_file\_names

# pathname rdataname settype

# ./activity\_labels.txt activity\_labels

# ./features.txt features

# ./train/subject\_train.txt subject\_train train

# ./train/X\_train.txt X\_train train

# ./train/y\_train.txt y\_train train

# ./test/subject\_test.txt subject\_test test

# ./test/X\_test.txt X\_test test

# ./test/y\_test.txt y\_test test

Use the assign() function in R to loop through every element in the above data frame. The “pathname” indicates the path from where the file will be read in using the read.table() function and the “rdataname” will be the R object for which each respective file will be read in to:

for (i in 1:nrow(all\_file\_names))

{

assign(as.character(all\_file\_names$rdataname[i]), read.table(as.character(all\_file\_names$pathname[i]), sep=""))

}

1. **Combine and join all data sets**

In this step, two main activities take place:

1. The test and training sets are combined into one set for each of the following: subject ID; activity number; outcome
2. The three sets above are then joined together along with the mapping from the activity number to each label

Combine the test and training sets:

subject <- rbind(subject\_train, subject\_test)

y <- rbind(y\_train, y\_test)

X <- rbind(X\_train, X\_test)

Create a unique id by which to join all 3 sets together

create\_id <- function(data)

{

mutate(data, id = 1:nrow(data))

}

subject <- create\_id(subject)

y <- create\_id(y)

X <- create\_id(X)

For the subject and y set rename the column names

setnames(subject, "V1", "subject")

setnames(y, "V1", "activityNum")

Join all three data sets together

datasets <- list(subject, y, X)

all\_data <- join\_all(datasets, by="id")

Join the activity labels as well - start by renaming the fields in the activity\_labels and y data sets

setnames(activity\_labels, c("V1","V2"), c("activityNum", "activity"))

Join together with the previous data set created

all\_data\_2 <- join\_all(list(activity\_labels, all\_data), by="activityNum")

1. **Extract only mean and stdev measurements**

We use the feature vector to find only the fields that contain calculations of the mean or standard deviation of the outcomes. We then use the feature vector to link back to the data set created in the previous step to assign the names and keep only the relevant fields

Begin by setting new names for the columns

setnames(features, names(features), c("featNum", "featName"))

Subset only those names containing "mean or "std" using the grepl function

featmeanstd <- features[grepl("mean\\(\\)|std\\(\\)", features$featName)==TRUE,]

Create a field name equivalent to that in the training and test sets in order to find these fields in the big data set

featmeanstd <- mutate(featmeanstd, field = paste("V",featmeanstd$featNum, sep=""))

head(featmeanstd)

# featNum featName field

# 1 tBodyAcc-mean()-X V1

# 2 tBodyAcc-mean()-Y V2

# 3 tBodyAcc-mean()-Z V3

# 4 tBodyAcc-std()-X V4

# 5 tBodyAcc-std()-Y V5

# 6 tBodyAcc-std()-Z V6

Keep a list of fieldnames we want to keep in the all\_data\_2 set

fieldnames <- c("activityNum","activity","subject", "id", featmeanstd$field)

fieldnames

[1] "activityNum" "activity" "subject" "id" "V1" "V2" "V3" "V4"

[9] "V5" "V6" "V41" "V42" "V43" "V44" "V45" "V46"

[17] "V81" "V82" "V83" "V84" "V85" "V86" "V121" "V122"

[25] "V123" "V124" "V125" "V126" "V161" "V162" "V163" "V164"

[33] "V165" "V166" "V201" "V202" "V214" "V215" "V227" "V228"

[41] "V240" "V241" "V253" "V254" "V266" "V267" "V268" "V269"

[49] "V270" "V271" "V345" "V346" "V347" "V348" "V349" "V350"

[57] "V424" "V425" "V426" "V427" "V428" "V429" "V503" "V504"

[65] "V516" "V517" "V529" "V530" "V542" "V543"

Now select only the columns containing the names in the above vector from the large data set with all the outcomes

all\_data\_2 <- all\_data\_2[,names(all\_data\_2) %in% fieldnames]

Rename fields in the large data set

RenameField <- c("activityNum","activity","subject", "id", as.character(featmeanstd$featName))

setnames(all\_data\_2, names(all\_data\_2), RenameField)

names(all\_data\_2)

[1] "activityNum" "activity" "subject"

[4] "id" "tBodyAcc-mean()-X" "tBodyAcc-mean()-Y"

[7] "tBodyAcc-mean()-Z" "tBodyAcc-std()-X" "tBodyAcc-std()-Y"

[10] "tBodyAcc-std()-Z" "tGravityAcc-mean()-X" "tGravityAcc-mean()-Y"

[13] "tGravityAcc-mean()-Z" "tGravityAcc-std()-X" "tGravityAcc-std()-Y"

[16] "tGravityAcc-std()-Z" "tBodyAccJerk-mean()-X" "tBodyAccJerk-mean()-Y"

[19] "tBodyAccJerk-mean()-Z" "tBodyAccJerk-std()-X" "tBodyAccJerk-std()-Y"

[22] "tBodyAccJerk-std()-Z" "tBodyGyro-mean()-X" "tBodyGyro-mean()-Y"

[25] "tBodyGyro-mean()-Z" "tBodyGyro-std()-X" "tBodyGyro-std()-Y"

[28] "tBodyGyro-std()-Z" "tBodyGyroJerk-mean()-X" "tBodyGyroJerk-mean()-Y"

[31] "tBodyGyroJerk-mean()-Z" "tBodyGyroJerk-std()-X" "tBodyGyroJerk-std()-Y"

[34] "tBodyGyroJerk-std()-Z" "tBodyAccMag-mean()" "tBodyAccMag-std()"

[37] "tGravityAccMag-mean()" "tGravityAccMag-std()" "tBodyAccJerkMag-mean()"

[40] "tBodyAccJerkMag-std()" "tBodyGyroMag-mean()" "tBodyGyroMag-std()"

[43] "tBodyGyroJerkMag-mean()" "tBodyGyroJerkMag-std()" "fBodyAcc-mean()-X"

[46] "fBodyAcc-mean()-Y" "fBodyAcc-mean()-Z" "fBodyAcc-std()-X"

[49] "fBodyAcc-std()-Y" "fBodyAcc-std()-Z" "fBodyAccJerk-mean()-X"

[52] "fBodyAccJerk-mean()-Y" "fBodyAccJerk-mean()-Z" "fBodyAccJerk-std()-X"

[55] "fBodyAccJerk-std()-Y" "fBodyAccJerk-std()-Z" "fBodyGyro-mean()-X"

[58] "fBodyGyro-mean()-Y" "fBodyGyro-mean()-Z" "fBodyGyro-std()-X"

[61] "fBodyGyro-std()-Y" "fBodyGyro-std()-Z" "fBodyAccMag-mean()"

[64] "fBodyAccMag-std()" "fBodyBodyAccJerkMag-mean()" "fBodyBodyAccJerkMag-std()"

[67] "fBodyBodyGyroMag-mean()" "fBodyBodyGyroMag-std()" "fBodyBodyGyroJerkMag-mean()"

[70] "fBodyBodyGyroJerkMag-std()"

Sort the data set by the subject and activity number

all\_data\_3 <- arrange(all\_data\_2, subject, activityNum)

1. **Restructure the data**

In this step we want to create a long and narrow data set in which we split the data into several columns with the following variables for each observation:

Column 1: Domain Signals {Time; Frequency}

Column 2: Accelarator {Body; Gravity; Not Available}

Column 3: Instrument {Accelerometer; Gyroscope}

Column 4: Jerk {Yes; No}

Column 5: Magnitude {Yes; No}

Column 6: Measure {Mean; Standard Deviation}

Column 7: Axis {X; Y; Z; Not Applicable}

Column 8: Value <The outcome>

First, melt the data set by creating a column showing the named feature of each of the feature variables

all\_data\_4 <- melt(all\_data\_3, id.vars = c("activityNum","activity","subject","id"))

Next, for each variable from the melted data set we want to assign values that are consistent with the columns defined in the beginning of this section. We do this using the grepl() function and creating Boolean vectors which indicate if the component we are interested in is present in each of the variable names. We then create a vector with the relevant name of the variable in each position corresponding to the metled data set and add that vector as a new field in the melted data set

Column 1: Domain Signals {Time; Frequency}

domvec <- data.frame(t=grepl("^t",all\_data\_4$variable),f=grepl("^f",all\_data\_4$variable))

domvec$val[which(domvec$t==FALSE & domvec$f == TRUE)] <- "Frequency"

domvec$val[which(domvec$t==TRUE & domvec$f == FALSE)] <- "Time"

all\_data\_4$DomainSignal <- domvec$val

Column 2: Accelarator {Body; Gravity; Not Available}

accvec <- data.frame(b=grepl("Body",all\_data\_4$variable),g=grepl("Gravity",all\_data\_4$variable))

accvec$val[which(accvec$b==TRUE & accvec$g==FALSE)] <- "Body"

accvec$val[which(accvec$b==FALSE & accvec$g==TRUE)] <- "Gravity"

all\_data\_4$Acceleration <- accvec$val

Column 3: Instrument {Accelerometer; Gyroscope}

insvec <- data.frame(a=grepl("Acc",all\_data\_4$variable),c=grepl("Gyro",all\_data\_4$variable))

insvec$val[which(insvec$a==FALSE & insvec$c==TRUE)] <- "Gyroscope"

insvec$val[which(insvec$a==TRUE & insvec$c==FALSE)] <- "Accelerometer"

all\_data\_4$Instrument <- insvec$val

Column 4: Jerk {Yes; No}

jerkvec<- data.frame(yn=grepl("Jerk",all\_data\_4$variable))

jerkvec$val[which(jerkvec$yn==TRUE)]<- "Yes"

jerkvec$val[which(is.na(jerkvec$val))]<- "No"

all\_data\_4$Jerk <- jerkvec$val

Column 5: Magnitude {Yes; No}

magvec<- data.frame(yn=grepl("Mag",all\_data\_4$variable))

magvec$val[which(magvec$yn==TRUE)]<- "Yes"

magvec$val[which(is.na(magvec$val))]<- "No"

all\_data\_4$Magnitude <- magvec$val

Column 6: Measure {Mean; Standard Deviation}

msdvec <- data.frame(m=grepl("mean()",all\_data\_4$variable),s=grepl("std()",all\_data\_4$variable))

msdvec$val[which(msdvec$m==FALSE & msdvec$s==TRUE)] <- "Standard Deviation"

msdvec$val[which(msdvec$m==TRUE & msdvec$s==FALSE)] <- "Mean"

all\_data\_4$Measure <- msdvec$val

Column 7: Axis {X; Y; Z; Not Applicable}

axisvec <- data.frame(x=grepl("-X",all\_data\_4$variable),

y=grepl("-Y",all\_data\_4$variable),

z=grepl("-Z",all\_data\_4$variable))

axisvec$val[which(axisvec$x==FALSE & axisvec$y==FALSE & axisvec$z==FALSE)] <- "Not Applicable"

axisvec$val[which(axisvec$x==TRUE & axisvec$y==FALSE & axisvec$z==FALSE)] <- "X"

axisvec$val[which(axisvec$x==FALSE & axisvec$y==TRUE & axisvec$z==FALSE)] <- "Y"

axisvec$val[which(axisvec$x==FALSE & axisvec$y==FALSE & axisvec$z==TRUE)] <- "Z"

all\_data\_4$Axis <- axisvec$val

1. **Summarise and output the tidy data set**

This step will summarise the melted data set with the new variables assigned by taking the count and average of each bucket. The final tidy data set will then be outputted by writing to a CSV file

Group by the names by which we wish to summarise

all\_data\_5 <- group\_by(all\_data\_4, subject, activity, DomainSignal, Acceleration, Instrument,

Jerk, Magnitude, Measure, Axis)

Take the mean of the value and count of occurences in each bucket

all\_data\_6 <- summarise(all\_data\_5, Average =mean(value), Count = n())

Order the data set by the subject and the activity

all\_data\_7 <- arrange(all\_data\_6, subject, activity)

Export the tidy data by writing it to a CSV file

write.table(all\_data\_7,"./Tidy Data Set.csv",sep=",")

1. **Create and export the codebook**

Finally, in this step we create the codebook which lists the variables in the tidy data set and provides a description for each variable

Get the unique list of values for each variable

ActivityName <- gsub("\_", " ", paste(unique(all\_data\_4$activity), sep="", collapse=", "))

DomainSignal <- paste(unique(all\_data\_4$DomainSignal), sep="", collapse=", ")

Acceleration <- paste(unique(all\_data\_4$Acceleration), sep="", collapse=", ")

Instrument <- paste(unique(all\_data\_4$Instrument), sep="", collapse=", ")

Jerk <- paste(unique(all\_data\_4$Jerk), sep="", collapse=", ")

Magnitude <- paste(unique(all\_data\_4$Magnitude), sep="", collapse=", ")

Measure <- paste(unique(all\_data\_4$Measure), sep="", collapse=", ")

Axis <- paste(unique(all\_data\_4$Axis), sep="", collapse=", ")

Create a description vector of each variable in the tidy data set

desc <- c("The ID of the subject who performed the activity. Values are numeric and range from 1 to 30",

paste("The name of activity undertaken. The activities are: {", ActivityName, "}", sep=""),

paste("The time or frequency domain signal. The singals are: {", DomainSignal, "}", sep=""),

paste("The acceleration signal: {", Acceleration, "}", sep=""),

paste("The measuring instrument. The instruments are: {", Instrument, "}", sep=""),

paste("An indication of whether the Jerk signal was calculated. Possible values are: {", Jerk, "}", sep=""),

paste("An indication of whether the magnitude of the signals was calculated. Possible values are: {", Magnitude, "}", sep=""),

paste("The measures which were calculated from the original observations. Possible values are: {", Measure, "}", sep=""),

paste("The axis on which the signals were calculated. Possible values are: {", Axis, "}", sep=""),

"The average value for each bucket",

"The count of each bucket"

)

The final codebook is a data frame containing two columns:

1. The name of each variable in the tidy data set

2. The description of each variable in the tidy data set

codebook <- data.frame(VariableName = names(all\_data\_7), Description = desc)

Export the codebook into a csv file

write.table(codebook,"./Code Book.csv",sep=",")