data$activity <- factor(data$activity, labels = c("X", "Y", "Z", ...)) where X, Y, and Z are your activity names (6 of them). Since there are 6, it renames the numbers 1-6 in the activity number and renames them. It also orders the factor levels which is nice.

thanks David, names helps . I followed the following to get the diagram .

  1. a) cbind X\_train.txt (7352x561) with subject\_train (7352x1)

      b) cbind the results with ytrain  to get a df with 7352x563

  2.  a) cbind X\_test.txt (2947x561) with subject\_test (2947x1 )

       b) cbind the results with ytest to get a df with 2947x563

   3.  row bind results of 1b) and 2b) above to get a df with 10299x563

How/where do I bind the activity data? The files in intertial signal have 128 cols , so I can't rbind them. And I don't know how to use those extra cols if I cbind them to x and y respectively.

the x{train,test} data gives me mean /std, and I don't understand the role of these intertial signal files and the activity labels.

I have finally put together actions to the diagram given by Dave.

Step 0: read all the files : listed along with dimensions

features.txt # 561x2  
X\_train.txt  # 7352x561head  
X\_test.txt  # 2947x561  
subject\_train.txt # 7352x1  
subject\_test.txt # 2947x1  
Y\_train.txt # 7532x1  
Y\_test.txt # 2947x1

1. rbind xtrain and xtest  # 10299x561

2. filter mean cols # 10299x46

3. set table headers using features

5. a) rbind subject train and subject test, and set the subject col names # 10299x1

   b) rbind ytrain with ytest, and set the activity col names  # 10299x1

  c) cbind xtrain mean cols from 2  with ytrain /test from 5b) .   # 10299x47

       cbind the result with subject trains/test from 5a)    # 10299x48

4. replace the activity with readable name. (Yes! I did 4 after 5)

I tried to use the"lapply" function, and here is what I got:

z<- subdata[, lapply(.SD, mean), by = c("Subject","Activity")]

Error in `[.data.frame`(subdata, , lapply(.SD, mean), by = c("Subject", :

unused argument (by = c("Subject", "Activity"))

subdata is a tidy data set with dimension of (10299x81), when I type names(subdata), here is what I got:

names(subdata)

[1] "Subject" "Activity"

[3] "tBodyAcc-mean()-X" "tBodyAcc-mean()-Y"

[5] "tBodyAcc-mean()-Z" "tBodyAcc-std()-X"

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

[9] "tGravityAcc-mean()-X" "tGravityAcc-mean()-Y"

[11] "tGravityAcc-mean()-Z" "tGravityAcc-std()-X" #......etc

I am not sure where I did wrong, any help will be appreciated.

Y\_test and Y\_train are the activity labels.  They should be cbind to the x\_train and x\_test data frames to provide the activity column in the data.  As stated in other threads and the directions, there is no need to use the  [Inertial Signals.](https://d396qusza40orc.cloudfront.net/getdata/lecture_slides/03_04_reshapingData.pdf" \o "Link: https://d396qusza40orc.cloudfront.net/getdata/lecture_slides/03_04_reshapingData.pdf)

I got at the fifth part of the course project to make an independent tidy data set with the average of each variable for each activity and each subject by using the reshape2 package after reading the following two web sites:  
  
[http://stackoverflow.com/questions/1407449/for-each-group-summarise-means-for-all-variables-in-dataf...](http://stackoverflow.com/questions/1407449/for-each-group-summarise-means-for-all-variables-in-dataframe-ddply-split)  
  
<http://seananderson.ca/2013/10/19/reshape.html>

aql **<-** melt(airquality, id.vars **=** **c**("month", "day"))

**head**(aql)

   features<-read.table("features.txt")  
   #I use this  to get indices of features with "mean"  
   mea<- grep("mean", features)   
   # the same for sd

Reading those two posts and trying various approaches allowed me to use two lines of code to go from my result in question four to a defensible answer in peer review (I hope) for question 5.  For background, I ended up choosing 66 columns with either mean or std in them.  I see others have chosen fewer, but there is likely no one answer....    as long as it is explained in the codebook.  With the Subject and Activity columns, my data frame has 68 columns, all with painstakingly chosen descriptive names.  
  
First, I assigned an object, Molten, to be melt(your.data.frame.name, id.vars = c("Subject", "Activity"))  where Subject and Activity are the column names in my data frame  
  
Then I used dcast as follows:   final.data.frame.hurray <- dcast(Molten, Subject + Activity ~ variable, fun = mean)

Thanks, David. Your solution (and Wickham's paper) are kind of blowing my mind.  
  
I'd watched the lecture videos but not really "gotten" what Tidy Data was until this activity (well, actually, if I'm honest, I'm not sure I do really *get*tidy data).  The distinction between variables as column names and variable values is amazing.  
  
Your **Molten** data frame seems to be what Wickham would consider "tidy" (one observation per row & stacked columns).  I'm still trying to grasp what good this does for us.  Especially since the next step is to "unstack" our variable values back into column names!   
  
I tried cross tabulating as a means of understanding... but I don't understand what values it's placing in each field.

xt <- xtabs(value ~ Subject + Activity,data=Molten)

xt[1:2,]

Activity

Subject WALKING WALKING\_UPSTAIRS WALKING\_DOWNSTAIRS SITTING STANDING LAYING

1 -1211.392670 -1103.048156 -483.023886 -2248.981816 -2630.100250 -2249.220529

2 -1267.295316 -858.365285 -330.088539 -2238.630988 -2637.254010 -2354.225554

After selecting all variables that included mean or std in some form I ended up with 86 variables, so after cbinding the activity names and subject numbers I had a dataframe with 88 columns.  
  
I then used the reshape2 package to melt the dataframe into 4 columns - subject, activity, feature, and measured value. This had 885714 rows.   
  
For step 5 I used the dplyr group\_by function with summarise 

final <- group\_by(final, subject, activity, feature)

summary <- summarise(final, mean(value))

 This resulted in a final dataframe with 4 columns -  subject, activity, feature, and mean value - grouped by subject, activity and feature

head(summary, 3)

Source: local data frame [3 x 4]

Groups: subject, activity

subject activity feature mean(value)

1 1 LAYING timeBodyAccelMeanX 0.22159824

2 1 LAYING timeBodyAccelMeanY -0.04051395

3 1 LAYING timeBodyAccelMeanZ -0.11320355

I'm just about to start working on the supporting documents now, but I'm a little bit lost on just what needs to go in each document.  
I'm assuming the README file is basically a method document for how the analysis was conducted and the code book is just a data dictionary explaining the features. I hope this is right, but I guess I'll find out soon enough.

A simple aggregate() is what I used to get required result in wide format. melt() form reshape2 or gather() from tidyr can be used to convert this to long format with ease.

 You should create one R script called run\_analysis.R that 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 variable names.
5. From the data set in step 4, creates a second, independent tidy data set with the average of each variable for each activity and each subject.

Good luck!

dataset[,lapply(.SD,mean),by=list(Activity,Subject)]

 library "data.table"

If you use dataset[, lapply(.SD, mean), by = subject] but get an error message "... unused argument (by = subject)", it is because the dataset is still a data.frame. Convert it into a data.table using dataset\_DT <- data.table(dataset) and the lapply function will work beautifully.

I used a different option mydata [, lapply(.SD, mean), <other options>] to generate a table of 180 rows with each representing 1 subject and 1 activity. I then used melt() to create a tidy table i.e. in long format.

data$activity <- factor(data$activity, labels = c("X", "Y", "Z", ...)) where X, Y, and Z are your activity names (6 of them). Since there are 6, it renames the numbers 1-6 in the activity number and renames them. It also orders the factor levels which is nice.

df$words <- "skiing"

df$words[df$number == 1] <- "swimming"

df$words[df$number == 2] <- "climbing"