Assignment 3

Chuheng Hu

Sep 28, 2016

## In this assignment you will be practising data tidying. You will be using the data we have collected from class and data generated from the instructor wearing a wristband activity tracker.

## First, you need to import into R a data set containing information about Charles' activity for the last three weeks. You can find this data set within the Assignment 3 repository you cloned to create this project.

## Install packages for manipulating data

We will use two packages: tidyr and dplyr

#Load packages  
library(tidyr,dplyr)

## Upload wide format instructor data (instructor\_activity\_wide.csv)

data\_wide <- read.table("~/Desktop/HUDK4050 Core mthds educ dara mining/RSTUDIO/CLASS 7/instructor\_activity\_wide.csv", sep = ",", header = TRUE)  
  
#Now view the data you have uploaded and notice how its structure: each variable is a date and each row is a type of measure.  
View(data\_wide)  
  
#R doesn't like having variable names that consist only of numbers so, as you can see, every variable starts with the letter "X". The numbers represent dates in the format year-month-day.#

## This is not a convenient format for us to analyze. What we need is for each type of measure to be a column. Your fisrt task is to convert wide format to long format data. To do this we will use the "gather" function: gather(data, time, variables)

The gather command requires the following input arguments:

* data: Data object
* key: Name of new key column (made from names of data columns)
* value: Name of new value column
* ...: Names of source columns that contain values

data\_long <- gather(data\_wide, date, variables)  
#Rename the variables so we don't get confused about what is what!  
 names(data\_long) <- c("variables", "date", "measure")  
#Take a look at your new data, looks weird huh?  
View(data\_long)

## Now convert this long format into separate columns using the "spread" function to separate by the type of measure

The spread function requires the following input:

* data: Data object
* key: Name of column containing the new column names
* value: Name of column containing values

instructor\_data <- spread(data\_long, variables, measure)

## Now we have a workable instructor data set!The next step is to create a workable student data set. Upload the data "student\_activity.csv". View your file once you have uploaded it and then draw on a piece of paper the structure that you want before you attempt to code it. Write the code you use in the chunk below. (Hint: you can do it in one step)

student\_data <- read.table("~/Desktop/HUDK4050 Core mthds educ dara mining/RSTUDIO/CLASS 7/student\_activity.csv", sep = ",", header = TRUE)  
student\_data<- spread(student\_data, variable, measure)

## Now that you have workable student data set, subset it to create a data set that only includes data from the second class.

To do this we will use the dplyr package (We will need to call dplyr in the command by writing dplyr:: because dplyr uses commands that exist in other packages but to do different operations.)

Notice that the way we subset is with a logical rule, in this case date == 20160204. In R, when we want to say that something "equals" something else we need to use a double equals sign "==". (A single equals sign means the same as <-).

student\_data\_2 <- dplyr::filter(student\_data, date == 20160204)

Now subset the student\_activity data frame to create a data frame that only includes students who have sat at table 4. Write your code in the following chunk:

student\_t4<-dplyr::filter(student\_data, table == 4)

## Make a new variable

It is useful to be able to make new variables for analysis. We can either apend a new variable to our dataframe or we can replace some variables with a new variable. Below we will use the "mutate" function to create a new variable "total\_sleep" from the light and deep sleep variables in the instructor data.

instructor\_data <- dplyr::mutate(instructor\_data, total\_sleep = s\_deep + s\_light)

Now, refering to the cheat sheet, create a data frame called "instructor\_sleep" that contains ONLY the total\_sleep variable. Write your code in the following code chunk:

instructor\_sleep<-dplyr::select(instructor\_data,total\_sleep)

Now, we can combine several commands together to create a new variable that contains a grouping. The following code creates a weekly grouping variable called "week" in the instructor data set:

instructor\_data <- dplyr::mutate(instructor\_data, week = dplyr::ntile(date, 3))

Create the same variables for the student data frame, write your code in the code chunk below:

student\_data <- dplyr::mutate(student\_data, week = dplyr::ntile(date, 3))

## Sumaraizing

Next we will summarize the student data. First we can simply take an average of one of our student variables such as motivation:

student\_data %>% dplyr::summarise(mean(motivation))

## mean(motivation)  
## 1 1.802469

#That isn't super interesting, so let's break it down by week:  
  
student\_data %>% dplyr::group\_by(date) %>% dplyr::summarise(mean(motivation))

## # A tibble: 3 <U+00D7> 2  
## date `mean(motivation)`  
## <int> <dbl>  
## 1 20160128 1.766667  
## 2 20160204 1.884615  
## 3 20160211 1.760000

Create two new data sets using this method. One that sumarizes average motivation for students for each week (student\_week) and another than sumarizes "m\_active\_time" for the instructor per week (instructor\_week). Write your code in the following chunk:

student\_week<-student\_data %>% dplyr::group\_by(week) %>% dplyr::summarise(mean(motivation))  
instructor\_week<-instructor\_data %>% dplyr::group\_by(week) %>% dplyr::summarise(mean(m\_active\_time))

## Merging

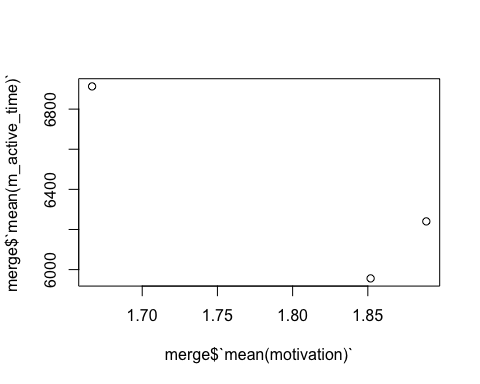
Now we will merge these two data frames using dplyr.

merge <- dplyr::full\_join(instructor\_week, student\_week, "week")

## Visualize

Visualize the relationship between these two variables (mean motivation and mean instructor activity) with the "plot" command and then run a Pearson correlation test (hint: cor.test()). Write the code for the these commands below:

plot(merge$`mean(motivation)`,merge$`mean(m\_active\_time)`)



cor.test(merge$`mean(motivation)`,merge$`mean(m\_active\_time)`)

##   
## Pearson's product-moment correlation  
##   
## data: merge$`mean(motivation)` and merge$`mean(m\_active\_time)`  
## t = -2.0734, df = 1, p-value = 0.2861  
## alternative hypothesis: true correlation is not equal to 0  
## sample estimates:  
## cor   
## -0.9007108

finally save your markdown document and your plot to this folder and commit,push and pull your repo to submit.