Reshaping Categorical Data with multiple values

UshaKiran.Kota

April 26, 2016

Categorical Data handling

Part 1 -Introduction

Sometime survey response data collection comprises of response levels of a single variable with multiple functions and multiple values. For e.g. In a hypothetical sruvey such as how frequently people buy or eat fruits, the data collection can be of following nature :

```
##      ID  AgeC GenderC doyoubuyappleseveryday doyoubuyapplesonceinaweek
## 1: 1234 25-30  female                    NA                          1
## 2: 1235   31+    male                     1                         NA
## 3: 1236 25-30  female                    NA                         NA
## 4: 1237 20-24  female                    NA                         NA
## 5: 1238 25-30    male                    NA                         NA
## 6: 1239 25-30    male                    NA                         NA
##    doyoubuyapplesonceinamonth doyoubuypearseveryday
## 1:                         NA                    NA
## 2:                         NA                    NA
## 3:                         NA                    NA
## 4:                         NA                    NA
## 5:                         NA                     1
## 6:                         NA                    NA
##    doyoubuypearsonceinaweek doyoubuypearssonceinamonth
## 1:                       NA                         NA
## 2:                       NA                         NA
## 3:                       NA                         NA
## 4:                        1                         NA
## 5:                       NA                         NA
## 6:                        1                         NA
##    doyoueatappleseveryday doyoueatapplesonceinaweek
## 1:                     NA                         1
## 2:                     NA                        NA
## 3:                     NA                         1
## 4:                     NA                        NA
## 5:                     NA                        NA
## 6:                     NA                        NA
##    doyoueatapplesonceinamonth doyoueatpearseveryday
## 1:                         NA                    NA
## 2:                         NA                     1
## 3:                         NA                    NA
## 4:                         NA                    NA
## 5:                          1                    NA
## 6:                         NA                    NA
##    doyoueatpearsonceinaweek doyoueatpearsonceinamonth
## 1:                       NA                         1
## 2:                       NA                        NA
```

```
## 3:                    NA                    NA
## 4:                     1                    NA
## 5:                    NA                    NA
## 6:                    NA                    NA
```

Categorical Data

Data is called categorical when an observation(a value) in the data can belong to a category such as "Gender (M/F)", "Age (range of Age)", "frequency - everyday , once a week, once a month"

In the above hypothetical data, the response variables such as "doyoubuyapples" or "doyoueatapples" are coded as "NA/1" depending on the response.

Multiple values

On observation , one can find that the data can be restructured to have 3 response variables, that are categorical in nature : 1. "fruit" that can take value as "apples" or "pears" 2. "frequency of buying" taking values "everyday" "once in a week" "once in a month 3."frequency of eating" taking values same as #2

Hence all 3 variables "fruit", "frequency of buying" and "frequency of eating" can now be re-coded as categorical variables taking multiple values as in the following use cases , *A person*:

1. did not buy fruits , but ate fruits , say, once a week
2. does not eat fruits, but bought fruits: say once a week apples, once a month pears
3. buys and eats same fruits same frequency: say , buy apples every day and eat apples every day
4. buys and eats same fruits with different frequencies : buys apples once a week, but eats them every day 5.buys and eats different fruits : buys apples every day , eats pears every day
5. buys and eats different fruits with different frequencies: buys pears once a month, eats apples once a week

There are more possible cases , essentially creating observations of multiple values for the same categorical variable such "fruit". Coding such observations would leading leaving "NA" values for frequencies that are not applicable for an observation. For.e.g a person who buys fruits but does not eat fruits would have a row in the data represented as *{apples, onceinaweek, NA}*

Grouping such data with a single variable "fruit" and taking multiple values as in frequency of buying and frequency of eating comprises of following problems 1. Treating NA values as valid 2. merge multiple columns of different lengths within the same data table 3. merge rows within to aggregate data by similar value

Reshaping Data

Although for survey convenience the data collection has been organized to take multiple variables (columns) for 2 functions such as "buy fruit" and "eat fruit", for data analysis, this data is in in "wide" format which can be **melted** to "long" format to eliminate multiple variables of same function The idea is to analyse data any association between these response variables.

Réshape library functions in R - melt()/dcast() help to acheive reduce the redundant frequency columns (12 in this case) to into 3 variables : In this hypothetical data, these 3 variables are "fruits" with their associated "buy" and "eat" frequencies

This document explains how to handle categorical data with multiple response values for a single explanatory variable

Part 2 -Reshape Data

The following code describes the reshape methods used to handle all the 3 problems discussed above

create auxiliary tables Each response variable in the data table is of type "function-item-frequency". For e.g. "doyoubuyappleseveryday" has "buy-apples-everyday" The reshaping of data is essentially to eliminate such long and multiple variables and reduce them to 1/4 This can be acheived with string match functions such as "grepl" In this document, the string matching of the variable names for "function", "item" and "frequency" is performed against a single column auxiliary tables for "fruit" and "frequency"

```
##   A
## A "apples"
## B "pears"
```

```
##   A
## A "everyday"
## B "onceinaweek"
## C "onceinamonth"
```

Part 2a -Treating NAs as valid observations

We know by now, that the data that has NA in the columns are valid data In the first step, to reduce the multiple "buy columns", melt() and dcast() can be used with the help of auxiliary tables and creating 2 new variables that can replace 6 of the "buy" variables

However during melt() we omit using na.rm = T, because such a flag will remove all the valid NA values in the Data. For e.g. we have an ID = 1236, who does not buy fruits but eats fruits. So while melting for "buy" variables, if na.rm = T is set, this observation is lost leading to loss of data.

```
#melt for buying  fruit frequencies
#search for match string "buy" in the variable names and melt() them as #measure variables

id.cols.buy.m <-names(repro.dt)[-grep("buy",names(repro.dt))]


repro.buy.m<-data.table(melt(repro.dt, id.vars = id.cols.buy.m,
                             measure.vars =grep("buy",names(repro.dt),
                                        value=T ),
                             variable.name = "buy.fruit",
                             value.name = "buy.freq"))

#create 2 new columns "fruit" and "freq.pur" (for frequency of buying/purchase)
repro.buy.m[, c("fruit", "freq.pur") := list(NA, NA)]

#use the auxiliary table "fruit" to find a matching fruit from a vector of
#fruits that matches with the current row of data table
#once there is a match assign the value to "fruit" variable only when #"buy.freq" is a valid value.

repro.buy.m$fruit = apply(repro.buy.m, 1, function(u){
  bool = sapply(fruit[,1], function(x) grepl(x, u[['buy.fruit']]))
  if(any(bool) & !(is.na(u[['buy.freq']]))) fruit[bool] else NA
})
```

```
#use the auxiliary table "frequency" to find a matching frequency from a #vector of frequencies that ma
#once there is a match assign the value to "freq.pur" variable only when the #ID bought a fruit, else l

repro.buy.m$freq.pur= apply(repro.buy.m, 1, function(u){
  bool = sapply(freq.levels[,1], function(x) grepl(x, u[['buy.fruit']]))
  if(!is.na(u[['fruit']])) freq.levels[bool] else NA
})
```

Check the non-unique rows

Omitting na.rm = T in the melt() call will result in duplicate/non-unique rows for all those columns that are left aside from "melt"–> in this case the columns that have a matching string as "eat"

We look at the number of non-unique rows for a single ID 1236, with a valid "NA" value for "fruit.pur" column, (the ID does not buy fruits), however, has a value set for "eat" variables-(The ID does not buy but eats fruits)- eliminate the non-unique rows

```
##      ID  AgeC GenderC doyoueatappleseveryday doyoueatapplesonceinaweek
## 1: 1236 25-30  female                    NA                          1
## 2: 1236 25-30  female                    NA                          1
## 3: 1236 25-30  female                    NA                          1
## 4: 1236 25-30  female                    NA                          1
## 5: 1236 25-30  female                    NA                          1
## 6: 1236 25-30  female                    NA                          1
##    doyoueatapplesonceinamonth doyoueatpearseveryday
## 1:                         NA                    NA
## 2:                         NA                    NA
## 3:                         NA                    NA
## 4:                         NA                    NA
## 5:                         NA                    NA
## 6:                         NA                    NA
##    doyoueatpearsonceinaweek doyoueatpearsonceinamonth
## 1:                       NA                         NA
## 2:                       NA                         NA
## 3:                       NA                         NA
## 4:                       NA                         NA
## 5:                       NA                         NA
## 6:                       NA                         NA
##                    buy.fruit buy.freq fruit freq.pur
## 1:     doyoubuyappleseveryday       NA    NA       NA
## 2:  doyoubuyapplesonceinaweek       NA    NA       NA
## 3: doyoubuyapplesonceinamonth       NA    NA       NA
## 4:      doyoubuypearseveryday       NA    NA       NA
## 5:   doyoubuypearsonceinaweek       NA    NA       NA
## 6: doyoubuypearssonceinamonth       NA    NA       NA

##      ID  AgeC GenderC doyoueatappleseveryday doyoueatapplesonceinaweek
## 1: 1236 25-30  female                    NA                          1
##    doyoueatapplesonceinamonth doyoueatpearseveryday
## 1:                         NA                    NA
##    doyoueatpearsonceinaweek doyoueatpearsonceinamonth
## 1:                       NA                         NA
##                buy.fruit buy.freq fruit freq.pur
## 1: doyoubuyappleseveryday       NA    NA       NA
```

Eliminate the redundant variables

Cleaned up data - Phase I

```
##      ID  AgeC GenderC doyoueatappleseveryday doyoueatapplesonceinaweek
## 1: 1210 25-30  female                      1                         NA
## 2: 1210 25-30  female                      1                         NA
## 3: 1234 25-30  female                     NA                          1
## 4: 1234 25-30  female                     NA                          1
## 5: 1235   31+    male                     NA                         NA
## 6: 1235   31+    male                     NA                         NA
##    doyoueatapplesonceinamonth doyoueatpearseveryday
## 1:                         NA                    NA
## 2:                         NA                    NA
## 3:                         NA                    NA
## 4:                         NA                    NA
## 5:                         NA                     1
## 6:                         NA                     1
##    doyoueatpearsonceinaweek doyoueatpearsonceinamonth  fruit    freq.pur
## 1:                       NA                        NA apples onceinaweek
## 2:                       NA                        NA     NA          NA
## 3:                       NA                         1 apples onceinaweek
## 4:                       NA                         1     NA          NA
## 5:                       NA                        NA apples    everyday
## 6:                       NA                        NA     NA          NA
```

Part2b. Reshape to remove redundant variables

melt()/dcast operations are used in a series to reshape the data so as to reduce the dimensions from 15 to 6 variables. However, reshape for only those rows of data that are unique by remaining variables–>using unique() here drops the duplicates – non-unique rows from the next melt operation, leaving the data table clean from all redundant variables with non-unique rows

Add another set of new variables for eat category fruits and frequencies of the same "fruit.et", "freq.et"

During the dcast, filter by NA rows again. However, *when filter for eat frequencies = NA removes rows of ID that buy fruits* ***but do not*** *eat fruits* Hence use filter either by fruit purchase or fruit eaten, a row that has both these observations = NA would mean that the ID neither bought fruits nor ate fruits

```
id.cols.eat <-colnames(repro.buy.c)[-grep("eat",names(repro.buy.c))]

measure.cols.eat <-grep("eat",names(repro.buy.c),value=T )

repro.eat.m<-data.table(melt(unique(repro.buy.c,by = measure.cols.eat),
                             id.vars = id.cols.eat,
                             measure.vars = grep("eat",
                                                 names(repro.buy.c),
                                                 value=T ),
                             variable.name = "eat.fruit",
                             value.name = "eat.freq"))


#add another column for those fruits that are eaten
repro.eat.m$fruit.et = apply(repro.eat.m, 1, function(u){
```

```
    bool = sapply(fruit[,1], function(x) grepl(x, u[['eat.fruit']]))
    if(any(bool) & !is.na(u[['eat.freq']])) fruit[bool] else NA

})
# #add fruit eating freq column
#
repro.eat.m$freq.et= apply(repro.eat.m, 1, function(u){

  bool = sapply(freq.levels[,1], function(x) grepl(x, u[['eat.fruit']]))
  if(!is.na(u[['eat.freq']]) & !is.na(u[['fruit.et']])) freq.levels[bool]
  else NA

  })


#remove the redundant "eat" columns

id.cols.eat.m <- colnames(repro.eat.m)[-grep("eat.fruit",names(repro.eat.m))]

f <- as.formula(paste(paste(id.cols.eat.m, collapse = " + "), "~ eat.fruit"))

repro.buy.eat.c<-data.table(dcast(data = repro.eat.m[!is.na(fruit) |
                                                       !is.na(fruit.et),],
                                  f, value.var = "eat.freq",
                                  function(x) length(x)))

repro.buy.eat.c<-repro.buy.eat.c[, which(grepl("eat",
                                 colnames(repro.buy.eat.c))):=NULL]
```

Clean Data - Phase II

All the 12 variables of "function_item_frequency" are now collapsed to 4 variables, and now the cleaned data is as in the table below:

However, there is a side effect at the end of this reshape. After the last reshape sequence, IDs with multiple buy and eat frquencies are duplicated either by, buy or eat frequencies for fruits that are bought but not eaten and vice versa.

Observe rows 2,4,10,11 in the below table which is a result of last reshape

These rows show that when there are observations of fruits that differ in name and eat/buy frequencies for the **same ID**, the previous values for existing frequencies are duplicated. *This is not a desired result*

Hence, further below, we make use of two more auxiliary variables to eliminate this side effect

```
##       ID  AgeC GenderC  fruit    freq.pur fruit.et      freq.et
##  1: 1210 25-30  female apples onceinaweek   apples     everyday
##  2: 1210 25-30  female apples onceinaweek       NA           NA
##  3: 1234 25-30  female apples onceinaweek   apples  onceinaweek
##  4: 1234 25-30  female apples onceinaweek    pears onceinamonth
##  5: 1234 25-30  female apples onceinaweek       NA           NA
##  6: 1235   31+    male apples    everyday    pears     everyday
##  7: 1235   31+    male apples    everyday       NA           NA
##  8: 1236 25-30  female     NA          NA   apples  onceinaweek
```

```
##  9: 1237 20-24  female  pears onceinaweek    pears  onceinaweek
## 10: 1237 20-24  female  pears onceinaweek       NA           NA
## 11: 1238 25-30    male  pears    everyday   apples onceinamonth
## 12: 1238 25-30    male  pears    everyday       NA           NA
## 13: 1239 25-30    male  pears onceinaweek       NA           NA
```

Part2c Merge columns of different lengths-using reshape

At the end of the Reshape process, the data table has now 2 variables that hold same item by different functions : in this case, fruit is the variable that is grouped, by buy or eat functions

The data when subset by "fruit" and "buy frequencies" results in row count that is different from the row count when the data is subset by "fruit" and "eat frequencies"

In an ideal case, a logical grouping would be having a single fruit column with different "buy/eat frequencies"

Taking the help of 2 more auxiliary variables freq.pur.x and freq.et.x, and setting NA values appropriately such a side effect can be reduced.

So the data table can be reshaped again to acheive this:

```
#melt again now to merge fruit columns (buy or eat - they are still fruits)-- so have just one column

id.cols.buy.eat.m<-colnames(repro.buy.eat.c)[-grep("fruit|fruit.et",
                                            names(repro.buy.eat.c))]

repro.buy.eat.m<-data.table(melt(repro.buy.eat.c,
                                 id.vars = id.cols.buy.eat.m,
                                 measure.vars = grep("fruit|fruit.et",
                                 names(repro.buy.eat.c),
                                 value=T ),
                                 variable.name = "fruit.final",
                                 value.name = "buy.eat.f"))

#set the freq.et value to NA if the fruit is in buy category and not in eat category

repro.buy.eat.m$freq.et.x = apply(repro.buy.eat.m, 1,
                            function(u){

                            if(u[['fruit.final']] == 'fruit' ) NA
                            else u[['freq.et']]

})

#set the freq.pur value to NA if the fruit is in buy category and not in eat category

repro.buy.eat.m$freq.pur.x = apply(repro.buy.eat.m, 1, function(u){

  if(u[['fruit.final']] == 'fruit.et') NA else u[['freq.pur']]
})

#add the merged column fruit with a condition to set either freq.pur or freq.et

id.cols.buy.eat.m <- colnames(repro.buy.eat.m)[-grep("final",
                                              names(repro.buy.eat.m))]
```

```
f <- as.formula(paste(paste(id.cols.buy.eat.m,
                            collapse = " + "), "~ fruit.final"))

#you just need to filter out rows that have NAs for "value.name" -- they are redundant
repro.buy.eat.final<-data.table(dcast(data =
                                        repro.buy.eat.m[!is.na(buy.eat.f),],
                                        f, value.var =
                                        "buy.eat.f",
                                        function(x) length(unique(x))))

#drop the redunant fruit columns -- because you have merged them now into buy.eat.f
repro.buy.eat.final<-repro.buy.eat.final[, which(grepl("fruit", colnames(repro.buy.eat.final))):=NULL]

#now rename the final-- merged column to "fruit"
rename(repro.buy.eat.final,replace = c("buy.eat.f" = "fruit"))

#drop old columns
repro.buy.eat.final<-repro.buy.eat.final[, c("freq.pur", "freq.et") := NULL]

#rename new columns
setnames(repro.buy.eat.final,old =c("freq.pur.x","freq.et.x"), new  = c("freq.pur", "freq.et"))

#get uique rows
repro.buy.eat.final<-unique(repro.buy.eat.final, by=c("ID", "freq.pur", "freq.et"))
```

Part2d Merged columns of different lengths within Data table

```
##       ID  AgeC GenderC  fruit      freq.et    freq.pur
##  1: 1210 25-30  female apples     everyday          NA
##  2: 1210 25-30  female apples           NA onceinaweek
##  3: 1234 25-30  female apples           NA onceinaweek
##  4: 1234 25-30  female  pears onceinamonth          NA
##  5: 1234 25-30  female apples  onceinaweek          NA
##  6: 1235   31+    male apples           NA    everyday
##  7: 1235   31+    male  pears     everyday          NA
##  8: 1236 25-30  female apples  onceinaweek          NA
##  9: 1237 20-24  female  pears  onceinaweek          NA
## 10: 1237 20-24  female  pears           NA onceinaweek
## 11: 1238 25-30    male apples onceinamonth          NA
## 12: 1238 25-30    male  pears           NA    everyday
## 13: 1239 25-30    male  pears           NA onceinaweek
```

Part2e- Merge rows within - Aggregate data by similar values

observe rows 2,5,10 in the above table: These IDs are of same fruits either with similar or different buy/eat frequencies. Such rows can be merged to a single row with appropriate buy/eat frequencies set. The following step aggregates the data by ID, merging rows of similar fruits.

At the end of the merge, we need to order rows by unique ID, fruit and frequencies, and again filter for unique rows by ID and fruit to get the final result desired

```
repro.buy.eat.final[, freq.pur:=(freq.pur[ fruit %in%
                                           repro.buy.eat.final$fruit &
                                           !is.na(freq.pur)]),
                                           by= c("ID", "fruit")]
```

```
##       ID  AgeC GenderC  fruit      freq.et    freq.pur
##  1: 1210 25-30  female apples     everyday onceinaweek
##  2: 1210 25-30  female apples           NA onceinaweek
##  3: 1234 25-30  female apples           NA onceinaweek
##  4: 1234 25-30  female  pears onceinamonth          NA
##  5: 1234 25-30  female apples  onceinaweek onceinaweek
##  6: 1235   31+    male apples           NA    everyday
##  7: 1235   31+    male  pears     everyday          NA
##  8: 1236 25-30  female apples  onceinaweek          NA
##  9: 1237 20-24  female  pears  onceinaweek onceinaweek
## 10: 1237 20-24  female  pears           NA onceinaweek
## 11: 1238 25-30    male apples onceinamonth          NA
## 12: 1238 25-30    male  pears           NA    everyday
## 13: 1239 25-30    male  pears           NA onceinaweek
```

```
#Get the final result

unique(repro.buy.eat.final[order(ID, fruit,
                                 freq.pur,freq.et)],
                                 by = c("ID", "fruit"))
```

```
##       ID  AgeC GenderC  fruit      freq.et    freq.pur
##  1: 1210 25-30  female apples     everyday onceinaweek
##  2: 1234 25-30  female apples  onceinaweek onceinaweek
##  3: 1234 25-30  female  pears onceinamonth          NA
##  4: 1235   31+    male apples           NA    everyday
##  5: 1235   31+    male  pears     everyday          NA
##  6: 1236 25-30  female apples  onceinaweek          NA
##  7: 1237 20-24  female  pears  onceinaweek onceinaweek
##  8: 1238 25-30    male apples onceinamonth          NA
##  9: 1238 25-30    male  pears           NA    everyday
## 10: 1239 25-30    male  pears           NA onceinaweek
```

References

http://stackoverflow.com/questions/24536771/r-datatable-conditionally-replacing-column-values

http://stackoverflow.com/questions/12399592/na-in-data-table

http://stackoverflow.com/questions/20987295/rename-multiple-columns-by-names

http://stackoverflow.com/questions/27955491/r-data-table-conditional-aggregation

http://stackoverflow.com/questions/12353820/sort-rows-in-data-table-r

This question has been raised in Stackoverflow forum http://stackoverflow.com/questions/36741643/melting-on-selected-columns-with-na-rm-t-removes-the-data-row-completely-alth

Acknowledgements

I sincerely thank Coursolve http://www.coursolve.org/ for providing me with an opportunity via the digital internships in Data Science that include such high value Data handling challenges within

I sincerely thank Mathew Dowle (mattjdowle@gmail.com) for encouraging me to post this question and continue to work on the challenges within.