Cleaning\_Description

Mackenzie Fry

October 25, 2017

library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(tidyr)  
library(readr)  
library(maps)  
library(purrr)

##   
## Attaching package: 'purrr'

## The following object is masked from 'package:maps':  
##   
## map

library(knitr)

clean\_yearly\_person\_file <- function(year) {  
 # 1. Read data in.  
 person\_file <- paste0("data-raw/yearly\_person\_data/person\_", year, ".csv")  
 df <- read\_csv(person\_file)  
   
   
 # 2. Convert all column names to lowercase.  
 colnames(df) <- tolower(colnames(df))  
   
 df <- df %>%  
 # 3. Limit variables.  
 select(st\_case, veh\_no, per\_no, state, per\_typ, lag\_hrs, lag\_mins,  
 inj\_sev, age, alc\_res, contains("drugres"), sex) %>%  
 # 4. Limit to relevant `per\_typ` and `inj\_sev` values, then remove those variables.  
 filter(per\_typ == 1 & inj\_sev == 4) %>%  
 select(-per\_typ, -inj\_sev) %>%  
 # 5. Create a `unique\_id`. Note: to be unique, `year` needs to be pasted on.  
 unite(unique\_id, st\_case, veh\_no, per\_no) %>%  
 mutate(year = year,  
 unique\_id = paste(unique\_id, year, sep = "\_")) %>%  
 # 6. Limit to study states and then remove the `state` variable.  
 filter(state %in% c(6,  
 15,  
 17,  
 33,  
 44,  
 54)) %>%  
 select(-state) %>%  
 # 7. Convert `sex` to a factor with levels "Male" and "Female".  
 mutate(sex = ifelse(sex == 9, NA, sex),  
 sex = factor(sex, levels = c(1, 2),  
 labels = c("Male", "Female"))) %>%  
 # 8. Use measured alcohol blood level to create `Alcohol` (logical for whether  
 # alcohol was present). Then remove the `alc\_res` variable.  
 mutate(alc\_res = ifelse(alc\_res > 94, NA, alc\_res / 10),  
 Alcohol = alc\_res >= 0.01) %>%  
 select(-alc\_res) %>%  
 # 9. Specify missing values for the lag minutes.  
 mutate(lag\_mins = ifelse(lag\_mins == 99, NA, lag\_mins))  
 # 10. Save lag hours coded as missing as `NA`.  
 if(year <= 2008){  
 df <- df %>%  
 mutate(lag\_hrs = ifelse(lag\_hrs %in% c(99, 999), NA, lag\_hrs))  
 } else {  
 df <- df %>%  
 mutate(lag\_hrs = ifelse(lag\_hrs == 999, NA, lag\_hrs))  
 }  
   
 # 11. Limit to deaths within an hour of the accident then remove those variables.  
 df <- df %>%  
 filter((lag\_hrs < 1) | (lag\_hrs == 1 & lag\_mins == 0)) %>%  
 select(-lag\_hrs, -lag\_mins)  
 # 12. Save age values coded as missing as `NA`.  
 if(year <= 2008){  
 df <- df %>%  
 mutate(age = ifelse(age == 99, NA, age))  
 } else {  
 df <- df %>%  
 mutate(age = ifelse(age %in% c(998, 999), NA, age))  
 }  
 # 13. Use age to create age categories and then remove `age` variable.  
 df <- df %>%  
 mutate(agecat = cut(age, breaks = c(0, 25, 45, 65, 1000),  
 labels = c("< 25 years",  
 "25--44 years",  
 "45--64 years",  
 "65 years +"),  
 include.lowest = TRUE, right = FALSE)) %>%  
 select(-age)  
 # 14. Gather all the columns with different drug listings (i.e., `drugres1`,  
 # `drugres2`, `drugres3`). Convert from the numeric code listings to  
 # drug categories.  
 gathered\_df <- df %>%  
 gather(drug\_number, drug\_type\_raw, contains("drugres")) %>%  
 mutate(drug\_type = ifelse(drug\_type\_raw %in% 100:295,  
 "Narcotic", NA),  
 drug\_type = ifelse(drug\_type\_raw %in% 300:395,  
 "Depressant", drug\_type),  
 drug\_type = ifelse(drug\_type\_raw %in% 400:495,  
 "Stimulant", drug\_type),  
 drug\_type = ifelse(drug\_type\_raw %in% 600:695,  
 "Cannabinoid", drug\_type),  
 drug\_type = ifelse(drug\_type\_raw %in% c(500:595, 700:996),  
 "Other", drug\_type),  
 drug\_type = ifelse(drug\_type\_raw == 1,  
 "None", drug\_type),  
 drug\_type = factor(drug\_type)) %>%  
 select(-drug\_type\_raw, -drug\_number) %>%  
 # 15. Filter out any observations where both alcohol and drug data is missing.  
 filter(!(is.na(Alcohol) & is.na(drug\_type)))  
 # 16. Create a subset with only individuals with at least one non-missing  
 # listing for drugs. (Write a sentence or two for each step in this pipe chain.)  
 non\_missing\_drugs <- gathered\_df %>%  
 filter(!is.na(drug\_type)) %>%  
 group\_by(unique\_id, drug\_type) %>%  
 summarize(has\_drug = TRUE) %>%  
   
 mutate(row\_num = 1:n()) %>%  
 spread(drug\_type, has\_drug, fill = FALSE) %>%  
 select(-row\_num)  
 # 17. Join this back into the full dataset. (Write a sentence or two for each  
 # step in this pipe chain.)  
 df <- df %>%  
 select(-contains("drugres")) %>%  
 full\_join(non\_missing\_drugs, by = "unique\_id") %>%  
 select(-None) %>%  
 gather(drug\_type, positive\_for\_drug, Alcohol, Cannabinoid,  
 Depressant, Narcotic, Other, Stimulant) %>%  
 mutate(drug\_type = factor(drug\_type)) %>%  
 unique()  
 return(df)  
}

load("~/r\_prog/data/fars\_analysis/data/clean\_fars.RData")

dim(clean\_fars)

## [1] 156413 6

length(unique(clean\_fars$unique\_id))

## [1] 25593

summary(clean\_fars)

## unique\_id sex year agecat   
## Length:156413 Male :121072 Min. :1999 < 25 years :39149   
## Class :character Female: 35335 1st Qu.:2002 25--44 years:61235   
## Mode :character NA's : 6 Median :2004 45--64 years:39870   
## Mean :2004 65 years + :16108   
## 3rd Qu.:2007 NA's : 51   
## Max. :2010   
## drug\_type positive\_for\_drug  
## Alcohol :25593 Mode :logical   
## Cannabinoid:26260 FALSE:127597   
## Depressant :25988 TRUE :16894   
## Narcotic :26086 NA's :11922   
## Other :26179   
## Stimulant :26307

clean\_fars %>%  
 mutate(year\_cat = cut(year, breaks = c(1999, 2002, 2006, 2010),  
 labels = c("1999-2002", "2003-2006",  
 "2007-2010"),  
 include.lowest = TRUE, right = TRUE)) %>%  
 filter(!is.na(sex)) %>%  
 group\_by(drug\_type, sex, year\_cat) %>%  
 summarize(n\_non\_missing = sum(!is.na(positive\_for\_drug)),  
 positive\_test = sum(positive\_for\_drug, na.rm = TRUE),  
 perc\_positive = round(100 \* positive\_test / n\_non\_missing, 1)) %>%  
 select(drug\_type, sex, year\_cat, perc\_positive) %>%  
 unite(sex\_year\_cat, sex, year\_cat) %>%  
 spread(sex\_year\_cat, perc\_positive) %>%  
 kable(col.names = c("Drug type", "F 1999-2002",  
 "F 2003-2006", "F 2007-2010",  
 "M 1999-2002", "M 2003-2006",  
 "M 2007-2010"))

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Drug type | F 1999-2002 | F 2003-2006 | F 2007-2010 | M 1999-2002 | M 2003-2006 | M 2007-2010 |
| Alcohol | 26.4 | 24.3 | 27.1 | 43.2 | 42.9 | 43.3 |
| Cannabinoid | 2.8 | 5.7 | 7.3 | 5.8 | 10.3 | 11.8 |
| Depressant | 3.4 | 3.8 | 4.8 | 2.0 | 2.5 | 3.2 |
| Narcotic | 4.2 | 4.9 | 7.0 | 2.2 | 3.4 | 4.0 |
| Other | 5.6 | 6.6 | 7.2 | 4.3 | 4.5 | 4.2 |
| Stimulant | 7.2 | 9.1 | 8.7 | 10.5 | 11.9 | 9.2 |

## Description of Process 1-18

1. In this first step, we are reading in the raw data from the FARS data. Right now there are 11 different files representating data from years 1999 to 2010. When we create person\_file we are telling R to paste0 a file location from our current working directory. We are working in our project fars\_analysis, which is automatcially the working directory. therefore, all we have to say is "data-raw" the folder the data is in. and person\_ indicates the beginning part of the name of all the files, year is referring to the specific year of the file. and then the final part is .csv which is the type of file it is saved as. We can now, create a dataframe, called df for now, that reads in using readr library the csv from the person\_file. We do this because now we can read in all the files we need,from this code line rather than reading each data set in separately.
2. In the second step, we are calling colnames(df) to refer to the column names of the df dataframe we just created in step 1. In this step we are telling R, that we want to make all the column names have lower case letters. We want to this so that it is easier to work with the data knowing there aren't any random lettings that might be capatilized. This step makes it less likely for us to make errors later.
3. As of now, df has every variable that the persons data contains. For the purpose of our fars analysis, we want to clean up our data, and make it more tidy for our specific use. So in step 3, we are limiting or selecting which varaibles we are going to want. in this case we want the following variables: st\_case which is the state the case happened in, veh\_no indcates the vehicle id, per\_no is the id representing the observation, state the state of the person in fips code format, per\_typeis the type of person they were in the accident (pedestrian, driver, passenger etc.,), lag\_hrs, lag\_mins, both lag hours and lag minutes indicate how long after the incident the person died.inj\_sev indicates the severity of the injury which could be no injuries to fatal, age represents the cases age alc\_res indicates the result of the blood alcohol levels, and we say contains(drug\_res) because we want all the columns that have some varaition of drug\_res in them (ex: drug\_res1, drug\_res2) classifies whether or not a person was using drugs and then what kind, and sex male or female. We are narrowing down our variables because we know from the Brandy and Li study these are the main variables they were focusing on, and any other varaibles are just making our tables untidy and hard more complicated.
4. We want to narrow down variables per\_type and inj\_serv to only the values we want to see. In the Brandy and Li study, the researchers wanted to study accidents where the driver had a fatality. To narrwo our data down, from looking at the FARS documentation, we know that per\_type can be range from pedestrians, to passengers, to the driver. We only want to see values of the drivers, so we are going to use the filter function from dplyr. We want per\_type == 1, but we also want to narrow down inj\_serv as well. For severity of injuries we want to see only fatalities. Here we want, inj\_serv == 4 which is the value for fatalites. To write the code, we want to find accidents where the driver had a fatality. This is why we use the & sign in the filter statement because we not only want driver, but also a fatality of the driver. Now that we have filtered to this specific argument. We want to work on other varaibles, so we can select everything except for those two values so we know what variables we still need to work on.
5. Our next step is to create a id that makes sense for our data set. Right now, we have three id variables measuring the same observations, which is untidy. To tidy our data, we want to use the uniteargument from the tidyr library to combine multiple columns into one. we want to unite st\_case, veh\_no, and per\_no into a column called unique\_id We also want to include the specific year as well. To do this we want to mutate year to be equal to year, and then create a new variable overwriting the old unique\_id which is now adding year. We are saying for this unique\_id variable we want to include what we had for unique\_id and include year with it, but to seperate (sep) by a"-" symbol. We did this so we could have a unqiue\_id variable that was specific to the information we would want as an observation indicator.
6. In the Brandy and Li study, the reserachers were only interested in six states specifically because they had the best data on tracking alcohol and drug testing from accidents. The states were California, Hawaii, Illinois, New Hampshire, Rhode Island, and West Virginia. So, we now want to limit what states we include in our data to just those states. To do this, we use the filter function from dplyr. for this statement we can use the %in% to indcate we want to choose from the state column those that are in 6, 15, 17, 33, 44, and 54. These are the fips codes for those states. Then like we did in step 4, we want to remove the state varible. We do so by selecting everything but state.
7. Next, we want to convert our sex variable from numeric value to a factor value. Right now 1 indcates a male and 2 indciates a female, and 9 indcates missing values. First, we give the missing values which were represented by a 9, a factor value of NA. We use the mutate function from dplyr to say if our sex == 9, it will be NA, for sex. Now, we can deal with our binary female and male numeric values, for this we are still using mutate to change again the sex variable. We are saying we want to factor sex at the levels of 1, and 2 which are the current numeric levels sex can be, and then we want to label these labels "male" if the level is 1, and "female", if the level is 2. We want to change these values to factors because it is easier to look at and understand, but it is also more tidy for what we want to do later on by grouping by sex.
8. In this step we are working with the alc\_res variable. We want to manipulate this column so that we can see which results were positive. This is telling us if alcohol was present or not. to do this, we want make a varaible with a logical statement that says TRUE if alcohol was present, and FALSE if alcohol was not present. To do this, we are going to use the mutate function from dplyr again, we want to mutate alc\_res, to say if it alc\_res is less more than 94, than it is missing or NA, alc\_res/10. Then we will mutate again, and create our logical statement, Alcohol is if alc\_res is greater than or equal to .01. We are doing this because if it is greater than .01 than it alcohol is present and the result wll be TRUE. Next, as we've done in past steps we delete the alc\_res variable because we no longer need it.
9. Next we are going to idenitfy missing values for the lag\_minutes variable. If a value was missing for lag\_minutes it was givena value of 99. So will use the mutate function from dplyr library, to say if lag\_mins is == 99, it will be NA for lag\_mins.
10. Now, we want to save lag\_hours that have NA values. Because the coding changed after 2008, we need to specify that there are two values 99, 999 that both indicate missing values as NA. This is why we used the mutate function from the dplyr library to say if the value for lag\_hrs is 99 or 999 then we say it is NA. By using mutate agaian, we can say that `lag\_hrs is NA if it is exactly equal to 999.
11. Now that we have taken care of the missing values, we want to specify the lag horus that we are focused on which is within the first hour. We use the filter function from the dplry library to say that from the df dataframe we have been working on, we want to filter to lag\_hours that are less than one, or is symbolized by the |, lag\_hours that have 1 hour AND the lag\_mins is exactly equal to 0. We are typing this, beause we want to find observations that are less than an hour, or exactly equal to an hour. Then as we have done before, we can say we want to get rid of both lag\_hours and lag\_mins because we no longer need to make edits to them.
12. Next we are focusing on the age variable. First, we want to save the missing values for age. Because the code is different after 2008, we need to specify what missing values indcate for the different years. So, here we will use the mutate function form the dplyr library, to say if the year is before 2008, mutate age to be missing if the value is 99, if it is not then we say to mutate age if the value is either 998 or 999 to be missing.
13. Now that we have delt with the missing values, we need to create and replicate the age categories that the Brandy and Li study used. This age groups were broken down by less than 25, 25-44, 45-64, and 65 and older. Again, we will be using the mutate function from the dplyr library to make these break points and different categories. In this code we are using agecat to print the ages by the cuts we create. Here were saying for age, create breaks at 0, 25, 45, 65, 1000). We are including 1000 to incorporate our NA values. so our labels then are less than 25, 25-44, 45-64, and 65+. With our statements at the end, we are saying we want to include the lowest as true, and then right is saying not to include the 1000 (as they are missing). Then as we've done in the other steps we are going to select all variables except age.
14. Next we are going to take all the columns that relate to drugs including those that have drug\_res or informat in them and gather them, drug\_number, and drug\_type\_raw into one column to ultimately create a drug\_type column. First, we are creating a new dataframe from our df called gathered\_df. We need to do this to get variabe that are similar to those in the Brady and Li paper. First, we used the gather function from the tidyr library to gather all those columns into one. Once we gathered the columns, we can then use our mutate function from the dplyr library to create the drug\_type column and different levels of new variable. In this part we looked to the FARS document to figure out what the different types of drugs were and there code. We also needed to see how Brady and Li coded their drugs because they combined a few drug types into a new catetgroy other. So if the raw\_drug\_type value ranged from 100 to 295 it was classified as Narcotics, 300 to 395 they were classified as Depressants, 400 to 495 the drug type was Stimulant, 600-695 was identified as Cannabinoid, 500 to 595, and 700 to 996 were classified as other, and finally, the last category for the drug\_type was if drug\_type\_raw was exactly 1, then it was identified as none. Now that we created this new column of drug\_type, we can then select all variables or columns except for drug\_type\_raw, and drug\_number as they are no longer needed and replaced with drug\_type.
15. Now that we have defined our variables, we want to remove missing values from the columns of drug\_type and Alcohol. This step is simple, we are going to use the filter function from the dplyr library and we want to use the "!" symbol which indicates "is not" and then we see is.na which means missing values. Our statement is saying we want to filter to values that are not missing for Alcohol and drug\_type. We are removing missing values because in the study from Brady and Li they were assessing the relevance of alcohol and drug involvement in fatal accidents and if we don't have information on either of them then they won't be useful in our analysis.
16. In the next step, we are going to create a subset with only individuals that have a least one non-missing listing for drugs. Here, we are trying to find individuals who tested positive or were under the influence of at least one drug type. We created a new dataframe and called it non-missing\_drugs we are pulling the data from our gathered\_df from step 15. We are going to use the filter function from the dplyr library to filter our gathered\_df to only drug\_type with out missing values. After that, we can then group\_by the unique\_id we created in step 5, and also drug\_type. Then we can summarize with a logical statement of has\_drug = TRUE. We are doing this to find out how many individuals has\_drug or took a drug. Next we are going to ungroup, and then mutate a new column called row\_num which is equal to 1 to "n" or the last number. Finally, we are going to use the spread function and spread drug\_type, has\_drug and then we are not going to fill. becaues we say fill= FALSE. Finally we can remove that row\_num column by selecting all varibles except for row\_num.
17. Now, we can join the non\_missing\_drugs dataframe back to the df dataframe we had worked on in the beginning. First, we want to select using the dplyr function, all varaibles that do no contain drugres in their title. Next again using dplry, we can use the full\_join which will return all values that are the same in both data frames and leave an NA value for the values that are not in both. Then we selected everything except for the None variable. Then using the tidyr function we gathered the following variables into one coloumn drug\_type, positive\_for\_drug, Alcohol, Cannabanoid, Depressant, Narcotic, Other, and Stimulant. Using the dplyr library we can mutate the variable drug\_type to be a factor. and then lastly, we can use the unique function to get only unique values. By using the return(df), we are saying we want to df with all the changes.
18. The last step, Finally we want to use the clean\_yearly\_file function across all of the data sets we have and save to one single data set. To do so , we create clean\_fars, We use to the map\_df to apply a function we created to different data sets. Here we say, we want to use (1999:2010), and use our function clean\_yearly\_person\_file which we created back in the beginning in step 1 to step 17. This means for each data set we want to do steps 1:17 which we created as a function clean\_yearly\_file. After we run this code, we then use the save function and we save it as clean\_fars into our data folder of our fars\_analysis project which is our current working directory which is why we can use relative pathways such as "data/clean\_fars.RData"