Homework #4–5

Fatality Analysis Reporting System (FARS) in action

With R, so far you know how to read in, clean, explore, and visulaize data. You're at the point where can use these skills to start to answer research questions at a larger level. Brady and Li (2013) wrote "Trends in Alcohol and Other Drugs Detected in Fatally Injured Drivers in the United States, 1999-2010" (http://aje.oxfordjournals.org/content/early/2014/01/27/aje.kwt327.full.pdf+html) using the same publically available FARS data (from the National Highway and Traffic Safety Administration) that we've been working with in class. For this homework, you will be replicating some of the results in that article.

Questions about the article (due for Homework #4)

First, read through the paper to get a feel for Brady and Li's motivation for conducting this study, research question, and their overall results. Then, think about how they used the FARS dataset to answer their research question, and answer the following questions about the article. For many of these questions, you will need to consult the FARS documentation file (https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812316).

1. In the in-course exercises, we have been analyzing data with accident as the observation unit. This study uses a different observation unit. What is the unit of observation in the Brady and Li study? When you download the FARS data for a year, you get a zipped folder with several different datasets. Which of the FARS datasets provides information at this observation level (and so will be the one you want to use for this analysis)?
2. This study only analyzes a subset of the available FARS data. Enumerate all of the constraints that are used by the study to create the subset of data they use in their study (e.g., years, states, person type, injury type). Go through the FARS documentation and provide the variable names for the variables you will use to create `filter` statements in R to limit the data to this subset. Provide the values that you will want to **keep** from each variable.
3. The study gives results stratified by age category, year, year category (e.g., 1999–2002), alcohol level, non-alcohol drug category, and sex. For each of these stratifications, give the variable name for the FARS variable you could use to make the split (i.e., the column name you would use in a `group_by` statement to create summaries within each category or the column name you would `mutate` to generate a categorical variable). Describe how each of these variables are coded in the data. Are there any values for missing data that you'll need to `mutate` to `NA` values in R? Are there any cases where coding has changed over the study period?

Set up an R Project (due for Homework #4— just the set-up of the project, not completion of all files within the project)

Now that we are doing larger-scale projects in R, you should set up your work as an R Project. Take the following steps to create a project for this assignment:

1. In RStudio, "File" -> "New Project" -> "New Directory". Navigate to the directory on your computere where you'd like to save your project and save the project.
2. This will save a new directory on your computer. It will only have one file in it, a ".Rproj" file.
3. Add the following subdirectories:
    - `data-raw`: This is where you'll put the raw FARS data you download from the Department of Transportation's website. You will store all the yearly files in a subdirectory of `data-raw` called `yearly_fars_data`. In the `data-raw` directory, you will also store an R script called "clean-data.R" that cleans the data into the final dataset you'll use for analysis. This script will include code defining a function called `clean_yearly_person_file` that will fully clean one year's data, as

well as a loop that applies this function to every study year to create one large cleaned dataframe called `clean_fars` that you will save in the `data` subdirectory. More details are given later in this assignment on the code that should be included in "clean-data.R".

- `data`: This is where you will store the cleaned dataset you'll use for analysis. You will store this cleaned data as "fars_data.Rdata". This file will be the output from the "clean-data.R" script in the `data-raw` subdirectory.
- `R`: This is where you will save an R script called "fars_analysis.R". This script will include the code where you define three functions (`perc_cis`, `test_trend_ca`, and `test_trend_log_reg`) to analyze the cleaned FARS data. More details are given about writing these functions later in the assignment.
- `writing`: This is where you'll save a file called "fars_analysis.Rmd" that replicates some of the analysis in the Brady and Li paper. This document should render a file that looks like the example "fars_analysis.pdf" document. If you have TeX on your computer, render this as a pdf; otherwise render it as a Word document.

To turn in this final homework (Homework #5), you will push your entire project directory to GitHub and send Rachel and me the link. (We will work on setting this up in an in-course exercise.)

Hint: Throughout this homework, remember that if you're working at the console, your working directory by default is your project directory. If you're working on an R Markdown file, the default working directory is the directory where that .Rmd file is saved.

Pull the raw data and save it locally (due for Homework #4)

FARS raw data is available by year from here. Download the zipped FARS "dbf" data files for all years and save the "person" file for each year locally.

You may choose one of the following ways to do this step:

- If you are feeling less ambitious, you can download the files by hand. Extract the "person" file for each year and save all these in `data-raw/yearly_person_data`.
- If you want an extra challenge, try to save download and save this data directly using R. If you do this, save your code in an R script called "download-data.R" in the `data-raw` subdirectory of the project. If you do this, you may find the `download.file` function very useful, as it can download a file with ftp. You'll need to check the naming conventions for the FARS ftp files. These conventions are different for 1999 and 2000 compared to other years, so you'll need an if / else statement within the function or loop you write. For example, for 1999, you need to download the data from ftp://ftp.nhtsa.dot.gov/fars/1999/DBF/FARSDBF99.zip while for 2001 you need to download the data from ftp://ftp.nhtsa.dot.gov/fars/2001/DBF/FARS2001.zip. Save these zipped files in a separate subdirectory of `data-raw` (e.g., `yearly_fars_data`). You can then write more code to unzip each file, extract the "person" data, convert it to a `.csv` file, and write those to the `yearly_person_data` subdirectory of `data-raw`. I will include an example of this code in the final homework solution.

Your final output of this step should be a `yearly_person_data` subdirectory in the `data-raw` subdirectory with a separate "dbf" or "csv" file for each year that contains the person-level FARS data for that year.

Write a script to generate a clean dataset (due for Homework #4)

Next, you will write some code to create a clean version of the data.

First, write a function called `clean_yearly_person_file` that inputs `year` and outputs a cleaned dataframe for that year. Save this function to "clean-data.R" in the `data-raw` subdirectory.

The cleaned yearly dataset should have filtered out any observations that should be excluded from analysis based on the Brady and Li paper (e.g., non-drivers, fatality not within one hour, etc.). It should have the following variables:

- `unique_id`: A unique identifier for each driver.
- `sex`: A factor with levels of "Male" and "Female".
- `year`: An integer with the 4-digit study year.
- `agecat`: A factor with age categories, as defined in Figures 1 and 3 of the Brady and Li paper.
- `drug_type`: A factor with the levels "Alcohol", "Cannabinoid", "Depressant", "Narcotic", "Stimulant", and "Other". These categorizations are based on categories defined in the FARS data documentation.
- `positive_for_drug`: Logical, whether that driver tested positive for that drug. For "Alcohol", this is based on a BAC $\geq$ 0.01 g/dL, as specified in the Brady and Li paper.

Note that this is not a tidy dataset— there are two levels of observation in this dataset (person and person-drug combination), which results in lots of repeated information (e.g., `age_cat` will be the same for a person across all their observations for different drugs). However, this data frame is in a format that will make it quick and convenient to write code to replicate results in the paper.

Here is an example of how the function should work, with an example of how the output dataframe should look (I'm grouping by `drug_type` to show how each person now has a listing for each of the drug categories, including "Alcohol", with either TRUE, FALSE, or NA):

```
data_1999 <- clean_yearly_person_file(1999)
data_1999 %>%
  group_by(drug_type) %>%
  slice(1:3)
```

```
## Source: local data frame [18 x 6]
## Groups: drug_type [6]
##
##         unique_id    sex  year       agecat   drug_type positive_for_drug
##             <chr> <fctr> <dbl>       <fctr>      <fctr>             <lgl>
## 1  60003_1_1_1999   Male  1999 25--44 years     Alcohol             FALSE
## 2  60006_1_1_1999   Male  1999   < 25 years     Alcohol              TRUE
## 3  60013_1_1_1999   Male  1999 45--64 years     Alcohol             FALSE
## 4  60003_1_1_1999   Male  1999 25--44 years Cannabinoid             FALSE
## 5  60006_1_1_1999   Male  1999   < 25 years Cannabinoid             FALSE
## 6  60013_1_1_1999   Male  1999 45--64 years Cannabinoid             FALSE
## 7  60003_1_1_1999   Male  1999 25--44 years  Depressant             FALSE
## 8  60006_1_1_1999   Male  1999   < 25 years  Depressant             FALSE
## 9  60013_1_1_1999   Male  1999 45--64 years  Depressant             FALSE
## 10 60003_1_1_1999   Male  1999 25--44 years    Narcotic             FALSE
## 11 60006_1_1_1999   Male  1999   < 25 years    Narcotic             FALSE
## 12 60013_1_1_1999   Male  1999 45--64 years    Narcotic             FALSE
## 13 60003_1_1_1999   Male  1999 25--44 years       Other             FALSE
## 14 60006_1_1_1999   Male  1999   < 25 years       Other             FALSE
## 15 60013_1_1_1999   Male  1999 45--64 years       Other             FALSE
## 16 60003_1_1_1999   Male  1999 25--44 years   Stimulant             FALSE
## 17 60006_1_1_1999   Male  1999   < 25 years   Stimulant             FALSE
## 18 60013_1_1_1999   Male  1999 45--64 years   Stimulant             FALSE
```

If you are struggling with writing this function, I've included some step-by-step tips at the end of this document.

Next, copy the following code into your "clean-data.R" script. You will use this code to loop through all the study years, apply your function to each year, join all the years together to create a single dataset, and save that dataset in the `data` subdirectory.

```
# Apply the function to clean the data across the study years
for(study_year in 1999:2010){
  df <- clean_yearly_person_file(study_year)
```

```
  if(study_year == 1999){
    clean_fars <- df
  } else {
    clean_fars <- rbind(clean_fars, df)
  }
}
save(clean_fars, file = "data/clean_fars.RData")
```

Note: Be sure that when you run this code, your working directory is the project directory.

After cleaning the data, you should have one large dataset called `clean_fars`. It should be saved in a file called "clean_fars.RData" in the `data` subdirectory. Summary statistics for your cleaned data should look something like this (do not worry if they do not exactly match— or if they don't exactly match numbers from the Brady and Li study— but they should be in the general ballpark):

```
load("../data/clean_fars.RData")
dim(clean_fars)
```

```
## [1] 163128      6
```

```
length(unique(clean_fars$unique_id))
```

```
## [1] 25593
```

```
summary(clean_fars)
```

```
##   unique_id             sex              year              agecat
##  Length:163128      Male  :126318   Min.   :1999   < 25 years  :40692
##  Class :character   Female: 36804   1st Qu.:2002   25--44 years:64116
##  Mode  :character   NA's  :     6   Median :2004   45--64 years:41766
##                                     Mean   :2004   65 years +  :16494
##                                     3rd Qu.:2007   NA's        :   60
##                                     Max.   :2010
##        drug_type     positive_for_drug
##  Alcohol    :27188   Mode :logical
##  Cannabinoid:27188   FALSE:133621
##  Depressant :27188   TRUE :17575
##  Narcotic   :27188   NA's :11932
##  Other      :27188
##  Stimulant  :27188
```

If you have cleaned your data correctly, you should be able to run the following code to show measurements of the prevalence of positive drug tests over the full study period by drug type and get results that are fairly similar to those shown below (this table will be included in your "fars_analysis" document):

```
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) %>%
knitr::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	27.0	24.5	27.2	43.4	43.1	43.6
Cannabinoid	2.7	5.4	6.9	5.6	9.9	11.5
Depressant	3.3	3.7	4.6	1.9	2.3	3.0
Narcotic	4.1	4.7	6.7	2.1	3.2	3.8
Other	5.5	6.4	6.9	4.1	4.3	4.1
Stimulant	7.0	8.7	8.2	10.1	11.5	8.9

Figures (due for Homework #5)

Use the cleaned dataset to recreate the three figures from the Brady and Li paper in your "fars_analysis" document. The final figures should look similar to the ones shown in the example "fars_analysis.pdf" document.

Write functions for `fars_functions.R` (due for Homework #5)

You will write three functions to help analyze the cleaned FARS data. Details are given for each function below. You should save the code defining these functions in a file called "fars_functions.R" in the `R` subdiretory.

Confidence intervals for proportions (`perc_cis`)

The text of the results gives estimates of percentages of drivers fatally injured who tested positive for a given drug by year, along with 95% confidence intervals for these percentages.

Write a function called `perc_cis` that inputs `x` (number of drivers testing positive for a drug) and `n` (total number of non-missing observations) and outputs a character vector for the percentage of drivers testing positive and the associated 95% confidence interval.

Within the code of this function, you will first want to calculate the proportion of drivers that test positive:

$$\hat{p} = \frac{x}{n}$$

Then, you can calculate an estimate of the standard of this proportion:

$$se(\hat{p}) = \sqrt{\frac{\hat{p}(1-\hat{p})}{n}}$$

Then calculate the upper and lower 95% confidence intervals as:

$$\hat{p} \pm 1.96 * se(\hat{p})$$

Finally, convert the proportion and its 95% CI to percentages, round the values, and paste everything together to create a one-element character vector that you could include in the text of results or in a table. For

example, you would change a proporation of 0.1011 with confidence intervals of 0.0951 and 0.1070 to "10.1% (9.5%, 10.7%)".

Here is an example of running the function when 9,000 drivers test positive for a drug out of 23,000 total observations:

```
perc_cis(x = 9000, n = 23000)
```

```
## [1] "39.1% (38.5%, 39.8%)"
```

Once you've written the `perc_cis` function, you will use it to generate a table with percentages and 95% CIs by drug type for the years 1999 and 2010 (see the example "fars_analysis.pdf" document).

Testing for trend using Cochran-Armitage trend test (`test_trend_ca`)

You can use the `prop.trend.test` function, which comes with base R in the `stats` package, to conduct a Cochran-Armitage test for trend in the proportion of drivers fatally injured who tested positive for a given drug. In the results from this function, the Z^2 statistic for the Cochran-Armitage test is given as the element `statistic` and its p-value is given by `p.value`. You can get the absolute value of the Z-statistic (closer to what is reported in the Brady and Li paper) by taking the square root of the Z^2 statistic. This function requires you to input a summary of the data: `x`, a vector with the number of drivers testing positive in each year, and `n`, a vector with the total number of non-missing observations in each year.

For example, to test for a trend for alcohol, you can run:

```
to_test <- clean_fars %>%
  filter(drug_type == "Alcohol") %>%
  group_by(year) %>%
  summarize(positive = sum(positive_for_drug, na.rm = TRUE),
            trials = sum(!is.na(positive_for_drug)))
ca_alcohol <- prop.trend.test(x = to_test$positive,
                              n = to_test$trials)
sqrt(ca_alcohol$statistic)
```

```
## X-squared
##  1.059612
```

```
ca_alcohol$p.value
```

```
## [1] 0.289321
```

Write a function that will input the `clean_fars` dataset and `drug` (a character vector giving one of the drug types or "Nonalcohol" for all non-alcohol drugs). The function should output a one-row dataframe with one column for the absolute value of the Z test statistic for the Cochran-Armitage trend test over the 12 study years for that drug and one column for the associated p-value. You will need to use an if / else statement within your code to run separate summarizing code for "Nonalcohol" versus other drug types. Set `data` to have a default value of `clean_fars`.

Here are some examples of calls and output from the function you will write:

```
test_trend_ca(drug = "Stimulant", data = clean_fars)
```

```
## # A tibble: 1 × 2
##       Z p.value
##   <dbl>   <dbl>
## 1   0.6   0.531
```

```
test_trend_ca(drug = "Alcohol")
```

```
## # A tibble: 1 × 2
##       Z p.value
##   <dbl>   <dbl>
## 1   1.1   0.289
```

```
test_trend_ca(drug = "Nonalcohol")
```

```
## # A tibble: 1 × 2
##       Z p.value
##   <dbl>   <dbl>
## 1   9.9       0
```

Once you've written the function, you will use `lapply` to apply it over all the drug categories (including all non-alcohol drugs), using the code below or similar code, to generate a table of results for tests of trends over the study years. You will use this code to write one of the tables in your "fars_analysis" R Markdown document.

```
drug_list <- c("Alcohol", "Nonalcohol", "Narcotic", "Depressant",
         "Stimulant", "Cannabinoid", "Other")
drug_trend_tests_ca <- lapply(drug_list, test_trend_ca)
drug_trend_tests_ca <- dplyr::bind_rows(drug_trend_tests_ca) %>%
  dplyr::mutate(drug = drug_list) %>%
  dplyr::select(drug, Z, p.value)
drug_trend_tests_ca %>% knitr::kable()
```

Testing for trend using logistic regression (`test_trend_log_reg`)

It turns out there are other ways to code for a trend test. Based on Agresti (*Categorical Data Analysis, Third Edition*, 2013):

> The Cochran-Armitage trend test seems unrelated to the linear logit model. However, this test statistic is equivalent to the score statistic for testing $H_0 : \beta = 0$ in that model. ... The Cochran-Armitage trend test (i.e., the score test) usually gives results similar to the Wald or likelihood-ratio test of $H_0 : \beta = 0$ in the linear logit model. The asymptotics work well even for quite small n when n_i are equal and x_i are equally spaced.

In our case, the approximation by the Wald statistic should work very well— we have a large n (total number of observations), the number of observations are fairly well distributed across the years (n_i denote the number of observations in each year, and we don't have lots in some years and few in others), and the years (x_i in this notation) are evenly spaced.

Here is a more practical comment on the choice of how to test for trend in proportions from an R help thread:

> It [i.e., the Cochran-Armitage test for trend] was taught us (epidemiologists) in the courses before we got our hands on logistic regression. ... I do not know of any advantages for the test over logistic regression or Poisson regression. You can take an ordered factor (or a non-ordered on if the levels are properly set up, coerce to numeric and do logistic regression with the numeric result and get pretty much the same result, and you would be doing so in the context of a much more flexible modeling environment. So I see it mainly as of historical interest, something to use when you only have a device that cannot run R.
>
> — David Winsemius

A linear logit (or logistic) model for the probability of a driver testing positive for a drug regressed on year is:

$$logit(\pi_i) = \alpha + \beta y_i$$

where y_i is the year of observation i and π_i is the probability a driver in that year tests positive for a given drug.

Remember that you can fit a logistic model like this in R using `glm` with `family = binomial(link = "logit")`. You can find the Wald statistic for testing $H_0 : \beta = 0$, as well as its p-value, in the `coefficients` element of the summary of the model object. For example, for alcohol, you could run:

```
to_test <- clean_fars %>%
      filter(drug_type == "Alcohol")
log_reg <- glm(positive_for_drug ~ year, data = to_test,
                 family = binomial(link = "logit"))
summary(log_reg)$coefficients
```

```
##                 Estimate  Std. Error   z value  Pr(>|z|)
## (Intercept) -8.559202791 7.673743351 -1.115388 0.2646841
## year         0.004056647 0.003828479  1.059597 0.2893278
```

The Wald statistic for a test of $H_0 : \beta = 0$ is 1.06, with a p-value of 0.289.

Write a function to fit the logistic model shown above to a specific drug category (including "Nonalcohol": all non-alcohol drugs). The function should take the same inputs as the function that you wrote for the Cochran-Armitage test (`drug` and `data`) and should create the same output (a one-row dataframe with test statistic and associated p-value), but in this case, the function should output the Wald statistic testing if the coefficient for year in the logistic model is zero ($H_0 : \beta = 0$). Set `data` to have a default value of `clean_fars`.

Here are some examples of running the function you will write:

```
test_trend_log_reg(drug = "Stimulant", data = clean_fars)
```

```
## # A tibble: 1 × 2
##       Z p.value
##   <dbl>   <dbl>
## 1  -0.6   0.531
```

```
test_trend_log_reg(drug = "Alcohol")
```

```
## # A tibble: 1 × 2
##       Z p.value
##   <dbl>   <dbl>
## 1   1.1   0.289
```

```
test_trend_log_reg(drug = "Nonalcohol")
```

```
## # A tibble: 1 × 2
##       Z p.value
##   <dbl>   <dbl>
## 1   9.9       0
```

You will use `lapply` with this function (using the code below of something similar) to create a table of test statistics and associated p-values for trend in proportions of fatally injured drivers that tested positive for specific drug categories. Again, this table will go in your "fars_analysis" document.

```
drug_list <- c("Alcohol", "Nonalcohol", "Narcotic", "Depressant",
         "Stimulant", "Cannabinoid", "Other")
drug_trend_tests_log_reg <- lapply(drug_list, test_trend_log_reg)
drug_trend_tests_log_reg <- dplyr::bind_rows(drug_trend_tests_log_reg) %>%
  dplyr::mutate(drug = drug_list) %>%
  dplyr::select(drug, Z, p.value)
drug_trend_tests_log_reg %>% knitr::kable()
```

Write "fars_analysis" document (due for Homework #5)

Create an R Markdown document in the `writing` subdirectory and call it "fars_analysis.Rmd".

You should start the document by including a code chunk that loads all required libraries and also loads the cleaned dataset you created and all of the functions you wrote in "fars_functions.R". For example, this chunk might look like:

```
library(dplyr)
library(tidyr)
library(ggplot2)

load("../data/clean_fars.RData")
source("../R/fars_functions.R")
```

The final document should look like the example "fars_analysis.pdf" document. Render to pdf if you have TeX on your computer. Otherwise, render to a Word document.

Tips for writing the function to clean yearly data

- First, read in your data and convert all the column names to lowercase (e.g., `colnames(df) <- tolower(colnames(df))`).
- Limit to only the variables you'll need to either filter the data or for the final cleaned data.
- Filter the dataset so it only includes drivers with fatal injuries. Once you have done this filtering, remove the variables you used for the filtering (e.g., `per_typ`), as you won't need these variables for anything else.
- Create a `unique_id`. You can do this by uniting the variables for `st_case`, `veh_no`, and `per_no`. To be unique when you join multiple years, `year` needs to be pasted on to this unique identifier.
- Limit to study states and then remove the `state` variable.
- Convert `sex` to a factor with levels "Male" and "Female". Make sure that you convert any codes for missing values into `NA`s.
- Use measured alcohol blood level to create `Alcohol` (logical for whether alcohol was present). Again, make sure that you convert any codes for missing values to `NA`. Use the definition from the paper to determine whether alcohol was present based on BAC results. Once you've created an `Alcohol` variable, remove the `alc_res` variable.
- Specify missing values for the lag hours and lag minutes. Because the missing value codes for lag hours changed during the study period, you'll need to use an if / else statement for this part. (As a note, there seem to be some cases from 2008 and earlier where `999` also codes for missing values for lag hours.) Limit to deaths within an hour of the accident then remove the variables for lag hours and lag minutes.
- Convert and codes for missing age to `NA`. Again, because this coding changed during the study, you'll need an if / else statement.
- Use `age` to create age categories (`agecat`) and then remove `age` variable. You may find the `cut` function useful for this.
- Gather all the columns with different drug listings (i.e., `drugres1`, `drugres2`, `drugres3`). Convert from the numeric code listings to drug categories (use the FARS documentation to figure out categories from codes). If you get *really* stuck on this part, I've included some example code below for what I've used in my function from this part through the end of the function.
- Filter out any observations in this "gathered" dataset where both alcohol and drug data are missing.
- Create a subset of the data (save as a different object rather than writing over your old object) with only individuals with at least one non-missing listing for drugs. Group by person and drug type, add a `has_drug` column that is always `TRUE`, and then spread this subset out so each drug has a separate column. Use `fill = FALSE` when you spread. Before you spread, you will need to add unique row numbers (`mutate(row_num = 1:n())`) for the spread to work properly, but then you can remove this `row_num` variable as soon as you make the spread.

- Join this spread dataset back into the full dataset using a full join and remove all `drugres` columns from the dataset. Remove the "None" column. Gather all drug categories (including `Alcohol`), with key `drug_type` and value `positive_for_drug`. Convert `drug_type` to a factor.

```
# 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 %>%
  tidyr::gather(drug_number, drug_type_raw, contains("drugres")) %>%
  dplyr::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)) %>%
  dplyr::select(-drug_type_raw, -drug_number) %>%

  # Filter out any observations where both alcohol and drug data is missing
  dplyr::filter(!(is.na(Alcohol) & is.na(drug_type)))

# Create a subset with only individuals with at least one non-missing
# listing for drugs
non_missing_drugs <- gathered_df %>%
  filter(!is.na(drug_type)) %>%
  group_by(unique_id, drug_type) %>%
  summarize(has_drug = TRUE) %>%
  ungroup() %>%
  mutate(row_num = 1:n()) %>%
  spread(drug_type, has_drug, fill = FALSE) %>%
  select(-row_num)

# Join this back into the full dataset
df <- df %>%
  dplyr::select(-contains("drugres")) %>%
  dplyr::full_join(non_missing_drugs, by = "unique_id") %>%
  dplyr::select(-None) %>%
  tidyr::gather(drug_type, positive_for_drug, Alcohol, Cannabinoid,
                Depressant, Narcotic, Other, Stimulant) %>%
  dplyr::mutate(drug_type = factor(drug_type))
```