UP431 Lab1: Exploring NHTS Data

Introduction: Using NHTS Data

Take a look at the NHTS Website. In the middle left of the page, there is an option to "Download Now!" with several file types.

Figure 1: Download Data

You could download the CSV files. But that would require a lot of manual recoding and applying factor labels. So we'll instead use the **SPSS** files that come pre-labeled. Download the zip file and extract it to your working directory.

We need to use the `haven` package to use the data in Rstudio, which allows us to read data files from non-R statistical packages. First, install and import the packages below. You will also need the `tidyverse` package to follow the practice today.

```
#install.packages("tidyverse")
#install.packages("haven")
library(tidyverse)
library(haven)
```

Then, download the SPSS files and unzip them in your **working folder** now. We'll read in the trips file to start.

```
trips <- read_sav("Data/spss (2)/trippub.sav") # put your directory here
```

When using the SPSS file, every variable comes with a label which tells you what the variable is about. Run the code below to see the labels of the trips data. Note that if you modify any of the variables, you'll lose the label. When you have to use other type of data that does not come with labels, you can always find them from the dataset codebook.

```
str(trips)
```

Also, you can see the variable names (column names) using the `names` function. Try out the code below.

```
names(trips)
```

```
##   [1] "HOUSEID"   "PERSONID"  "TDTRPNUM"  "STRTTIME"  "ENDTIME"   "TRVLCMIN"
##   [7] "TRPMILES"  "TRPTRANS"  "TRPACCMP"  "TRPHHACC"  "VEHID"     "TRWAITTM"
##  [13] "NUMTRANS"  "TRACCTM"   "DROP_PRK"  "TREGRTM"   "WHODROVE"  "WHYFROM"
##  [19] "LOOP_TRIP" "TRPHHVEH"  "HHMEMDRV"  "HH_ONTD"   "NONHHCNT"  "NUMONTRP"
##  [25] "PSGR_FLG"  "PUBTRANS"  "TRIPPURP"  "DWELTIME"  "TDWKND"    "VMT_MILE"
##  [31] "DRVR_FLG"  "WHYTRP1S"  "ONTD_P1"   "ONTD_P2"   "ONTD_P3"   "ONTD_P4"
##  [37] "ONTD_P5"   "ONTD_P6"   "ONTD_P7"   "ONTD_P8"   "ONTD_P9"   "ONTD_P10"
##  [43] "ONTD_P11"  "ONTD_P12"  "ONTD_P13"  "TDCASEID"  "TRACC_WLK" "TRACC_POV"
##  [49] "TRACC_BUS" "TRACC_CRL" "TRACC_SUB" "TRACC_OTH" "TREGR_WLK" "TREGR_POV"
##  [55] "TREGR_BUS" "TREGR_CRL" "TREGR_SUB" "TREGR_OTH" "WHYTO"     "TRAVDAY"
##  [61] "HOMEOWN"   "HHSIZE"    "HHVEHCNT"  "HHFAMINC"  "DRVRCNT"   "HHSTATE"
##  [67] "HHSTFIPS"  "NUMADLT"   "WRKCOUNT"  "TDAYDATE"  "HHRESP"    "LIF_CYC"
##  [73] "MSACAT"    "MSASIZE"   "RAIL"      "URBAN"     "URBANSIZE" "URBRUR"
##  [79] "GASPRICE"  "CENSUS_D"  "CENSUS_R"  "CDIVMSAR"  "HH_RACE"   "HH_HISP"
##  [85] "HH_CBSA"   "SMPLSRCE"  "R_AGE"     "EDUC"      "R_SEX"     "PRMACT"
##  [91] "PROXY"     "WORKER"    "DRIVER"    "WTTRDFIN"  "WHYTRP90"  "TRPMILAD"
##  [97] "R_AGE_IMP" "R_SEX_IMP" "VEHTYPE"   "OBHUR"     "DBHUR"     "OTHTNRNT"
## [103] "OTPPOPDN"  "OTRESDN"   "OTEEMPDN"  "OBHTNRNT"  "OBPPOPDN"  "OBRESDN"
## [109] "DTHTNRNT"  "DTPPOPDN"  "DTRESDN"   "DTEEMPDN"  "DBHTNRNT"  "DBPPOPDN"
## [115] "DBRESDN"
```

The first we need to do is convert all the labels into something that R recognizes: factors. Use the `as_factor` function from the `haven()` package.

```
trips <- as_factor(trips)
```

This is a large file. Just to explore a bit, let's subset only to those trips from households in the Chicago metropolitan area. The `HH_CBSA` variable contains the identifier for the Core-Based Statistical Area. (The larger version of a metroplitan area. **What do you notice about the CBSAs?**) Note that the code for Chicago is 16980, but because we ran the `as_factor` function, we must use the full name of the CBSA in our filter.

```
chi_trips <- trips %>%
  filter(HH_CBSA == "Chicago-Naperville-Elgin, IL-IN-WI")
```

Now we have a much more managable dataset: 6,955 rows.

Exploring some data structures

One of the most important variables in the dataset is `TRPTRANS`: Mode of transportation for that trip. Let's look at what's in it.

```
levels(chi_trips$TRPTRANS)
```

```
##  [1] "I prefer not to answer"
##  [2] "I don't know"
##  [3] "Not ascertained"
##  [4] "Walk"
##  [5] "Bicycle"
##  [6] "Car"
##  [7] "SUV"
##  [8] "Van"
##  [9] "Pickup truck"
## [10] "Golf cart / Segway"
## [11] "Motorcycle / Moped"
## [12] "RV (motor home, ATV, snowmobile)"
## [13] "School bus"
## [14] "Public or commuter bus"
## [15] "Paratransit / Dial-a-ride"
## [16] "Private / Charter / Tour / Shuttle bus"
## [17] "City-to-city bus (Greyhound, Megabus)"
## [18] "Amtrak / Commuter rail"
## [19] "Subway / elevated / light rail / street car"
## [20] "Taxi / limo (including Uber / Lyft)"
## [21] "Rental car (Including Zipcar / Car2Go)"
## [22] "Airplane"
## [23] "Boat / ferry / water taxi"
## [24] "Something Else"
```

There are 24 modes of transportation. That's too many! We can use R to collapse the factors into a fewer number of categories using a function from the `tidyverse` called `fct_collapse`. Let's save the collapsed variable into a new factor called `mode_short`. I'll get the code started for you, but you should finish it:

```
chi_trips <- chi_trips %>%
  mutate(mode_short = fct_collapse(TRPTRANS,
                                   Walk = "Walk",
                                   Bicycle = "Bicycle",
                                   Auto = c("Car", "SUV", "Van"))) # What else can go here?

# Hint: Look at Appendix C in the NHTS Travel Trends to determine which modes are in which broad catego
```

Descriptive Analysis of Travel Behavior

Task 1. Let's use this information to calculate the *percentage of trips made by each of the following modes of transportation: automobile, public transportation, walking, cycling, and other modes*. We want to calculate the **weighted** percentage of trips. You can use either the `tally` or `count` functions to do this.

```
## Your code goes here
```

Task 2. Create a bar chart of the data using `ggplot2`.

```
library(ggplot2)
## Your code goes here
```

You can change the order of labels by `factor()` and `labels()`.

```
## Your code goes here
```

Task 3. This time, let's use one more variable to calculate the *weighted* percentage of trips by mode *for commute trips and non-commute trips*. To do this task, you should first find and use a variable that describes the trip purpose. What is that variable? How many trip purposes are there?

```
levels(chi_trips$WHYTRP90)
```

```
##  [1] "To/From Work"                  "Work-Related Business"
##  [3] "Shopping"                      "Other Family/Personal Business"
##  [5] "School/Church"                 "Medical/Dentral"
##  [7] "Visit Friends/Relatives"       "Other Social/Recreational"
##  [9] "Other"                         "Refused / Don't Know"
```

Use `count` or `tally` to calculate the percentage of each travel mode. We should aggregate all purposes that are not related to commuting to a 'non-commute trip'. You can again use the `fct_collapse` function. I got the code started for you, but you should finish it.

```
mode_share_commute <- chi_trips %>%
  mutate(commuteTrp = fct_collapse(WHYTRP90,
                                   commute_trip = "To/From Work",
                                   non_commute_trip = c("Work-Related Business","Shopping","Other Family
                                   Missing = "Refused / Don't Know"))
                                  ## Your code goes here
```

Task 4. Create a bar chart that efficiently visualizes the data from Task 3.

```
## Your code goes here
```

Task 5. Calculate a separate percentage for carpooling and create a new bar chart. What additional variable do you need to determine whether someone carpooled?

NOTE: As the data type of the variable is 'factor', you should convert the data to a numeric type to operate mathematical comparison.You can use `as.numeric` and `as.character` to do the task.

```
as.numeric(as.character(chi_trips$NUMONTRP))
```

```
## Your code goes here
```

Now create a bar chart.This time, sort the order of bars by using `reorder`.

```
## Your code goes here
```

Saving your work

R Notebooks are set up so that you can just rerun the analysis using the original input data, so you don't have to worry about saving any other files except the RMarkdown file. However, you may wish to save time later or use the data for a different analysis. For example, converting the SPSS data to an R format isn't instantaneous, so you might like to have a native R data file.

The native R format is called an RDS (R Data Set) file. You can use the `write_rds` function from the `tidyverse` to save R objects. It's a best practice to save the files in an `output` directory or something similarly named to distinguish the raw, input data from the data you've operated on.

```
write_rds(chi_trips, "Output/chi_trips.rds")
```

Reading the RDS file back in is as simple as assigning the output of `read_rds("filename")` to a new object.