Data Management

CJ 702: Advanced Criminal Justice Statistics

Thomas Bryan Smith*

February 12, 2025

Contents

1 Setup Environment and Import Data

```
# As always, we want to start by loading in the package(s) we plan on using:
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.3.3

## Warning: package 'ggplot2' was built under R version 4.3.3

## Warning: package 'tidyr' was built under R version 4.3.3

## Warning: package 'readr' was built under R version 4.3.3

## Warning: package 'dplyr' was built under R version 4.3.3

## Warning: package 'stringr' was built under R version 4.3.3

## Warning: package 'lubridate' was built under R version 4.3.3
```

*University of Mississippi, tbsmit10@olemiss.edu

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
# Next, we need to read in the data we need to manage. We are going
# to work with a small subset of the National Crime Victimization
# Survey (NCVS) MSA Public-Use Data, 2000-2015. I have already
# prepared a subset of these data, but you can download the full
# data here: https://www.icpsr.umich.edu/web/NACJD/studies/38321

#   United States. Bureau of Justice Statistics.
#   National Crime Victimization Survey: MSA Public-Use Data,
#   2000-2015. Inter-university Consortium for Political and
#   Social Research [distributor], 2022-03-21.
#   https://doi.org/10.3886/ICPSR38321.v1

# In this module we are going to practice some subsetting
# and cleaning, then merge the different levels of data
# datasets so that we have something to work with when we
# reach the multilevel modelling module

# Read in the data:
household <- readRDS("./Raw_Data/household.rds")
person <- readRDS("./Raw_Data/person.rds")
incident <- readRDS("./Raw_Data/incident.rds")

# Variables
## YEAR: Year of Interview
## YEARQ: Year and Quarter of Interview
## IDPER: Person ID
## IDHH: Household ID

## Household
### V2026: Household income
### V2125: Land Use (Urban v. Rural)
### WGTHH: Household Weight

## Person
### V3014: Age
### V3018: Sex
### V3020: Educational Attainment
### WGTPER: Person Weight

## Incident
### TOC_RECODE: Violent Crime Code
### WGTVIC: Victimization Weight
### SERIESWGT: Series Weight
```

```r
# Before we move on, let's quickly just view these three datasets:
## Household
head(household)
```

```
## # A tibble: 6 x 6
##    YEAR YEARQ IDHH       V2026              V2125 WGTHH
##   <dbl> <fct> <fct>      <fct>              <fct> <dbl>
## 1  2000 001   2000993788 $75,000 and over   Urban  836.
## 2  2000 001   2000167846 $50,000 to $74,999 Urban 1052.
## 3  2000 001   2000733306 $25,000 to $29,999 Urban  836.
## 4  2000 001   2000111147 $40,000 to $49,999 Urban  840.
## 5  2000 001   2000514983 $25,000 to $29,999 Urban 1052.
## 6  2000 001   2000303653 Residue            Rural  845.
```

```r
## Person
head(person)
```

```
## # A tibble: 6 x 9
##    YEAR YEARQ IDPER      IDHH       V3014 V3018  V3020       WGTPER   YIH
##   <dbl> <fct> <fct>      <fct>      <fct> <fct>  <fct>        <dbl> <dbl>
## 1  2000 001   2000966984 2000993788 40-49 Male   College      1063.    10
## 2  2000 001   2000951294 2000993788 40-49 Female College       894.     9
## 3  2000 001   2000470356 2000993788 12-17 Male   Elementary   1317.     9
## 4  2000 001   2000205990 2000167846 35-39 Male   College      1093.     4
## 5  2000 001   2000361146 2000167846 30-34 Female College      1101.     4
## 6  2000 001   2000879996 2000733306 40-49 Male   High school  1063.     6
```

```r
## Incident
head(incident)
```

```
## # A tibble: 6 x 7
##    YEAR YEARQ IDPER      IDHH       TOC_RECODE                    WGTVIC SERIESWGT
##   <dbl> <fct> <fct>      <fct>      <fct>                          <dbl>     <dbl>
## 1  2000 001   2000361146 2000167846 Threatened assault with we~  2202.         1
## 2  2000 001   2000365150 2000514983 Attempted/completed theft    2104.         1
## 3  2000 001   2000220530 2000362293 Attempted/completed motor ~  1893.         1
## 4  2000 001   2000335184 2000591381 Burglary                     1673.         1
## 5  2000 001   2000638577 2000186105 Attempted/completed robber~  3165.         1
## 6  2000 001   2000345273 2000402319 Attempted/completed theft    1913.         1
```

2 Indexing and Recoding

```r
# First, let's extract a variable to work with:
income <- household$V2026

# Now let's familiarize ourself with the variable.
# What class is the vector?
class(income)
```

```
## [1] "factor"
```

```
# Factors are a special type of vector, typically used for labelling
# ordinal and nominal variables. When printed, factors appear
# similar to a character vector:
head(income)
```

```
## [1] $75,000 and over   $50,000 to $74,999 $25,000 to $29,999 $40,000 to $49,999
## [5] $25,000 to $29,999 Residue
## 15 Levels: Less than $5,000 $5,000 to $7,499 ... Residue
```

```
# However, you'll notice that there are 'levels':
levels(income)
```

```
##  [1] "Less than $5,000"   "$5,000 to $7,499"   "$7,500 to $9,999"
##  [4] "$10,000 to $12,499" "$12,500 to $14,999" "$15,000 to $17,499"
##  [7] "$17,500 to $19,999" "$20,000 to $24,999" "$25,000 to $29,999"
## [10] "$30,000 to $34,999" "$35,000 to $39,999" "$40,000 to $49,999"
## [13] "$50,000 to $74,999" "$75,000 and over"   "Residue"
```

```
# This is because the vector is, at it's core, numeric.
# Using the as.numeric() function will reveal the numbers
# underlying each response in the variable:
income |> as.numeric() |> head()
```

```
## [1] 14 13  9 12  9 15
```

```
# The first 6 observations are the 14th, 13th, 9th, 12th, 9th, and 15th
# level of the factor. You can use the levels() function to cross ref.

# Finally, let's tabulate the variable:
table(income)
```

```
## income
##   Less than $5,000   $5,000 to $7,499   $7,500 to $9,999 $10,000 to $12,499
##                400                232                248                354
## $12,500 to $14,999 $15,000 to $17,499 $17,500 to $19,999 $20,000 to $24,999
##                297                394                438                876
## $25,000 to $29,999 $30,000 to $34,999 $35,000 to $39,999 $40,000 to $49,999
##                978               1206               1105               1891
## $50,000 to $74,999   $75,000 and over            Residue
##               3558               5728               7263
```

```
# ============================================================================ #

# Indexing is the process by which you tell R which elements of a vector
# you want to return. For example, if you want the first element of a vector:
income[1]
```

```
## [1] $75,000 and over
## 15 Levels: Less than $5,000 $5,000 to $7,499 ... Residue
```

```r
# Or the fifty-second element of the vector:
income[52]
```

```
## [1] $35,000 to $39,999
## 15 Levels: Less than $5,000 $5,000 to $7,499 ... Residue
```

```r
# If you want to return a set of elements, you have to index with a
# vector of numbers. If you wanted the first 10, you would enter:
income[1:10]
```

```
##  [1] $75,000 and over   $50,000 to $74,999 $25,000 to $29,999 $40,000 to $49,999
##  [5] $25,000 to $29,999 Residue            $7,500 to $9,999   $15,000 to $17,499
##  [9] $7,500 to $9,999   $30,000 to $34,999
## 15 Levels: Less than $5,000 $5,000 to $7,499 ... Residue
```

```r
# Because "1:10" generates a vector of numbers from 1 to 10:
1:10
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```r
# This also works using the seq() function:
income[seq(1, 10)]
```

```
##  [1] $75,000 and over   $50,000 to $74,999 $25,000 to $29,999 $40,000 to $49,999
##  [5] $25,000 to $29,999 Residue            $7,500 to $9,999   $15,000 to $17,499
##  [9] $7,500 to $9,999   $30,000 to $34,999
## 15 Levels: Less than $5,000 $5,000 to $7,499 ... Residue
```

```r
# The index vector can be any set of numbers, as long as it
# describes a set of meaningful positions in the vector.
# For instance, you could select every other element:
(n <- length(income))
```

```
## [1] 31513
```

```r
income[seq(1, n, by = 2)] |> head()
```

```
## [1] $75,000 and over   $25,000 to $29,999 $25,000 to $29,999 $7,500 to $9,999
## [5] $7,500 to $9,999   $75,000 and over
## 15 Levels: Less than $5,000 $5,000 to $7,499 ... Residue
```

```r
# Why is this important? Because it allows you to subset your data.
# If you wanted to extract every other element in the vector,
# you could simply assign the result of the previous code to an object:
income_subset <- income[seq(1, n, by = 2)]
head(income_subset)
```

```
## [1] $75,000 and over   $25,000 to $29,999 $25,000 to $29,999 $7,500 to $9,999
## [5] $7,500 to $9,999   $75,000 and over
## 15 Levels: Less than $5,000 $5,000 to $7,499 ... Residue
```

```r
# When we tabled the income variable, you may have noticed the "Residue"
# label. This is a special code in NCVS which encodes incomplete data
# collection. These data are effectively missing, but are not encoded as
# "NA" - R's missing data code. You can use indexing to find "Residue":
income[which(income == "Residue")] |> head()
```

```
## [1] Residue Residue Residue Residue Residue Residue
## 15 Levels: Less than $5,000 $5,000 to $7,499 ... Residue
```

```r
# You can also set up the logical operation to check the number ("level")
# associated with the factor label, rather than the label for that level:
income[which(as.numeric(income) == 15)] |> head()
```

```
## [1] Residue Residue Residue Residue Residue Residue
## 15 Levels: Less than $5,000 $5,000 to $7,499 ... Residue
```

```r
# Why is this important? Because it allows you to replace non-standard
# missing codes with the "NA" special character. This is how you
# "find and replace" using R syntax.
income[which(income == "Residue")] <- NA

# I have already performed this operation for every other variable in
# the data, but let's correct the income variable (V2026) in the data
# by recoding "Residue" as "NA" (remember: the $ character allows
# you to extract or target a specific variable from a data frame):
household$V2026[which(household$V2026 == "Residue")] <- NA

# If you tabulate the income variable in the household data frame,
# you will see that there are no more "Residue" observations:
table(household$V2026)
```

```
##
##   Less than $5,000   $5,000 to $7,499   $7,500 to $9,999 $10,000 to $12,499
##                400                232                248                354
## $12,500 to $14,999 $15,000 to $17,499 $17,500 to $19,999 $20,000 to $24,999
##                297                394                438                876
## $25,000 to $29,999 $30,000 to $34,999 $35,000 to $39,999 $40,000 to $49,999
##                978               1206               1105               1891
## $50,000 to $74,999   $75,000 and over            Residue
##               3558               5728                  0
```

```r
# However, you will also see that "Residue" still appears as a label.
# You can redefine the labels of the factor. First, extract the old
# labels using the levels() function:
(labels <- levels(household$V2026))
```

```
##  [1] "Less than $5,000"   "$5,000 to $7,499"   "$7,500 to $9,999"
##  [4] "$10,000 to $12,499" "$12,500 to $14,999" "$15,000 to $17,499"
##  [7] "$17,500 to $19,999" "$20,000 to $24,999" "$25,000 to $29,999"
## [10] "$30,000 to $34,999" "$35,000 to $39,999" "$40,000 to $49,999"
## [13] "$50,000 to $74,999" "$75,000 and over"   "Residue"
```

```r
# We do not want the "Residue" label (and level), so let's extract
# every other label from this vector of labels:
(labels <- labels[which(labels != "Residue")])
```

```
##  [1] "Less than $5,000"   "$5,000 to $7,499"   "$7,500 to $9,999"
##  [4] "$10,000 to $12,499" "$12,500 to $14,999" "$15,000 to $17,499"
##  [7] "$17,500 to $19,999" "$20,000 to $24,999" "$25,000 to $29,999"
## [10] "$30,000 to $34,999" "$35,000 to $39,999" "$40,000 to $49,999"
## [13] "$50,000 to $74,999" "$75,000 and over"
```

```r
# Now we can use the factor() variable to tell R what we want
# the new labels to be:
household$V2026 <- factor(household$V2026, labels = labels)

# On an important note, you need to make sure that your labels
# are presented in the order that match the variable coding.
# If we have a variable consisting of 3 values:
(var <- c(1:3, 2:3))
```

```
## [1] 1 2 3 2 3
```

```r
# If the encoding scheme is 1 = a, 2 = b, 3 = c, then you only
# need to specify the "labels" option because the levels of the
# oridinal variable already match the order of the labels:
factor(var, labels = c("a", "b", "c"))
```

```
## [1] a b c b c
## Levels: a b c
```

```r
# If you aren't sure what "order" the values of the vector are
# assumed to be, then you can check the order that they appear
# when tabulated:
table(var)
```

```
## var
## 1 2 3
## 1 2 2
```

```r
# But if the encoding scheme is 1 = b, 2 = c, 3 = a, then you will need
# to manually define the levels of the variable (as well as the labels):
factor(var, levels = c(2, 3, 1), labels = c("b", "c", "a"))
```

```
## [1] a b c b c
## Levels: b c a
```

```r
# ============================================================================ #

# Now that you're familiar with indexing, let's quickly use it to rename
# the variables in our household data frame.

# First, print the current data frame column names:
colnames(household)
```

```
## [1] "YEAR"  "YEARQ" "IDHH"  "V2026" "V2125" "WGTHH"
```

```
# "YEAR" and "YEARQ" are self-explanatory,
# "IDHH", "MSAIND", "WGTHH" are convenient,
# but "V2026" and "V2125" are a little unhelpful.

# Which position are they in the colnames() vector?
which(colnames(household) %in% c("V2026", "V2125"))
```

```
## [1] 4 5
```

```
# Let's take the logical operation and use it to index the colnames() vector.
# Then, once indexed, we can assign the new names:
colnames(household)[which(colnames(household) %in%
                            c("V2026", "V2125"))] <- c("INCOME", "LAND_USE")
```

3 Data Manipulation

```
# While indexing is an important skill, there are more convenient ways
# of manipulating data in R. Tidyverse, and more specifically dplyr,
# offer a range of important tools for data manipulation.

# Filtering (subsetting by logical operation):
household %>%
  filter(YEAR >= 2010,
         LAND_USE == "Rural")
```

```
## # A tibble: 1,949 x 6
##     YEAR YEARQ IDHH       INCOME             LAND_USE WGTHH
##    <dbl> <fct> <fct>      <fct>              <fct>    <dbl>
##  1  2010 101   2010853697 $12,500 to $14,999 Rural    2391.
##  2  2010 101   2010563082 <NA>               Rural       0
##  3  2010 101   2010839301 <NA>               Rural    1730.
##  4  2010 101   2010291790 <NA>               Rural       0
##  5  2010 101   2010851071 $40,000 to $49,999 Rural    1136.
##  6  2010 101   2010798616 $25,000 to $29,999 Rural    1407.
##  7  2010 101   2010420458 <NA>               Rural       0
##  8  2010 101   2010639250 <NA>               Rural       0
##  9  2010 101   2010145118 $20,000 to $24,999 Rural     774.
## 10  2010 101   2010749632 $50,000 to $74,999 Rural    1030.
## # i 1,939 more rows
```

```
# Arranging (sorting the data by a variable)
household %>%
  arrange(YEAR,
          LAND_USE)
```

```
## # A tibble: 31,513 x 6
##     YEAR YEARQ IDHH       INCOME             LAND_USE WGTHH
```

```
##    <dbl> <fct> <fct>      <fct>              <fct>    <dbl>
##  1  2000 001   2000993788 $75,000 and over   Urban     836.
##  2  2000 001   2000167846 $50,000 to $74,999 Urban    1052.
##  3  2000 001   2000733306 $25,000 to $29,999 Urban     836.
##  4  2000 001   2000111147 $40,000 to $49,999 Urban     840.
##  5  2000 001   2000514983 $25,000 to $29,999 Urban    1052.
##  6  2000 001   2000665487 $30,000 to $34,999 Urban    1185.
##  7  2000 001   2000362293 $75,000 and over   Urban     946.
##  8  2000 001   2000758454 $40,000 to $49,999 Urban     828.
##  9  2000 001   2000532321 $50,000 to $74,999 Urban    1185.
## 10  2000 001   2000967150 <NA>               Urban     893.
## # i 31,503 more rows
```

```
# Selecting (extracting specific variables)
household %>%
  select(YEAR,
         LAND_USE)
```

```
## # A tibble: 31,513 x 2
##     YEAR LAND_USE
##    <dbl> <fct>
##  1  2000 Urban
##  2  2000 Urban
##  3  2000 Urban
##  4  2000 Urban
##  5  2000 Urban
##  6  2000 Rural
##  7  2000 Rural
##  8  2000 Rural
##  9  2000 Rural
## 10  2000 Urban
## # i 31,503 more rows
```

```
# The tidyverse pipe operator, %>%, can then be used to chain any
# combination of these functions in any order. Piping to the
# summarise() function will let you quickly calculate descriptive
# statistics for a variable (after filtering by certain conditions).
household %>%

  filter(YEAR >= 2010,
         LAND_USE == "Rural") %>%

  summarise(median = median(as.numeric(INCOME), na.rm = TRUE),
            mean = mean(as.numeric(INCOME), na.rm = TRUE),
            sd = sd(as.numeric(INCOME), na.rm = TRUE),
            min = min(as.numeric(INCOME), na.rm = TRUE),
            max = max(as.numeric(INCOME), na.rm = TRUE))
```

```
## # A tibble: 1 x 5
##   median  mean    sd   min   max
##    <dbl> <dbl> <dbl> <dbl> <dbl>
## 1     12  10.8  3.50     1    14
```

```r
# The group_by() function will let you easily calculate
# group-level descriptive statistics.
household %>%
  group_by(YEAR,
           LAND_USE)
```

```
## # A tibble: 31,513 x 6
## # Groups:   YEAR, LAND_USE [32]
##     YEAR YEARQ IDHH       INCOME             LAND_USE WGTHH
##    <dbl> <fct> <fct>      <fct>              <fct>    <dbl>
##  1  2000 001   2000993788 $75,000 and over   Urban     836.
##  2  2000 001   2000167846 $50,000 to $74,999 Urban    1052.
##  3  2000 001   2000733306 $25,000 to $29,999 Urban     836.
##  4  2000 001   2000111147 $40,000 to $49,999 Urban     840.
##  5  2000 001   2000514983 $25,000 to $29,999 Urban    1052.
##  6  2000 001   2000303653 <NA>               Rural     845.
##  7  2000 001   2000160785 $7,500 to $9,999   Rural     905.
##  8  2000 001   2000697120 $15,000 to $17,499 Rural     845.
##  9  2000 001   2000995639 $7,500 to $9,999   Rural     905.
## 10  2000 001   2000665487 $30,000 to $34,999 Urban    1185.
## # i 31,503 more rows
```

```r
# The mutate() function will let you quickly manipulate
# or update an existing variable, creating or overwriting
# the variable you manipulate. Here, we are creating
# a variable called "INC" that is a numeric vector
# version of the original "INCOME" factor vector.
(household <- household %>%
   mutate(INC = as.numeric(INCOME)))
```

```
## # A tibble: 31,513 x 7
##     YEAR YEARQ IDHH       INCOME             LAND_USE WGTHH   INC
##    <dbl> <fct> <fct>      <fct>              <fct>    <dbl> <dbl>
##  1  2000 001   2000993788 $75,000 and over   Urban     836.    14
##  2  2000 001   2000167846 $50,000 to $74,999 Urban    1052.    13
##  3  2000 001   2000733306 $25,000 to $29,999 Urban     836.     9
##  4  2000 001   2000111147 $40,000 to $49,999 Urban     840.    12
##  5  2000 001   2000514983 $25,000 to $29,999 Urban    1052.     9
##  6  2000 001   2000303653 <NA>               Rural     845.    NA
##  7  2000 001   2000160785 $7,500 to $9,999   Rural     905.     3
##  8  2000 001   2000697120 $15,000 to $17,499 Rural     845.     6
##  9  2000 001   2000995639 $7,500 to $9,999   Rural     905.     3
## 10  2000 001   2000665487 $30,000 to $34,999 Urban    1185.    10
## # i 31,503 more rows
```

```r
# You would also use the mutate() function to create
# centered or standardized variables.
## Centered:
household %>%
  mutate(INC_CEN = INC - mean(INC, na.rm = TRUE))
```

```
## # A tibble: 31,513 x 8
```

```
##     YEAR YEARQ IDHH       INCOME             LAND_USE WGTHH   INC INC_CEN
##    <dbl> <fct> <fct>      <fct>              <fct>    <dbl> <dbl>   <dbl>
##  1  2000 001   2000993788 $75,000 and over   Urban     836.    14   2.75
##  2  2000 001   2000167846 $50,000 to $74,999 Urban    1052.    13   1.75
##  3  2000 001   2000733306 $25,000 to $29,999 Urban     836.     9  -2.25
##  4  2000 001   2000111147 $40,000 to $49,999 Urban     840.    12   0.754
##  5  2000 001   2000514983 $25,000 to $29,999 Urban    1052.     9  -2.25
##  6  2000 001   2000303653 <NA>               Rural     845.    NA  NA
##  7  2000 001   2000160785 $7,500 to $9,999   Rural     905.     3  -8.25
##  8  2000 001   2000697120 $15,000 to $17,499 Rural     845.     6  -5.25
##  9  2000 001   2000995639 $7,500 to $9,999   Rural     905.     3  -8.25
## 10  2000 001   2000665487 $30,000 to $34,999 Urban    1185.    10  -1.25
## # i 31,503 more rows
```

```
## Z-Score Standardized:
household %>%
  mutate(INC_STD = (INC - mean(INC, na.rm = TRUE)) / sd(INC, na.rm = TRUE))
```

```
## # A tibble: 31,513 x 8
##     YEAR YEARQ IDHH       INCOME             LAND_USE WGTHH   INC INC_STD
##    <dbl> <fct> <fct>      <fct>              <fct>    <dbl> <dbl>   <dbl>
##  1  2000 001   2000993788 $75,000 and over   Urban     836.    14   0.821
##  2  2000 001   2000167846 $50,000 to $74,999 Urban    1052.    13   0.523
##  3  2000 001   2000733306 $25,000 to $29,999 Urban     836.     9  -0.669
##  4  2000 001   2000111147 $40,000 to $49,999 Urban     840.    12   0.225
##  5  2000 001   2000514983 $25,000 to $29,999 Urban    1052.     9  -0.669
##  6  2000 001   2000303653 <NA>               Rural     845.    NA  NA
##  7  2000 001   2000160785 $7,500 to $9,999   Rural     905.     3  -2.46
##  8  2000 001   2000697120 $15,000 to $17,499 Rural     845.     6  -1.56
##  9  2000 001   2000995639 $7,500 to $9,999   Rural     905.     3  -2.46
## 10  2000 001   2000665487 $30,000 to $34,999 Urban    1185.    10  -0.371
## # i 31,503 more rows
```

```
household %>%
  mutate(INC_STD = scale(INC))
```

```
## # A tibble: 31,513 x 8
##     YEAR YEARQ IDHH       INCOME             LAND_USE WGTHH   INC INC_STD[,1]
##    <dbl> <fct> <fct>      <fct>              <fct>    <dbl> <dbl>       <dbl>
##  1  2000 001   2000993788 $75,000 and over   Urban     836.    14       0.821
##  2  2000 001   2000167846 $50,000 to $74,999 Urban    1052.    13       0.523
##  3  2000 001   2000733306 $25,000 to $29,999 Urban     836.     9      -0.669
##  4  2000 001   2000111147 $40,000 to $49,999 Urban     840.    12       0.225
##  5  2000 001   2000514983 $25,000 to $29,999 Urban    1052.     9      -0.669
##  6  2000 001   2000303653 <NA>               Rural     845.    NA      NA
##  7  2000 001   2000160785 $7,500 to $9,999   Rural     905.     3      -2.46
##  8  2000 001   2000697120 $15,000 to $17,499 Rural     845.     6      -1.56
##  9  2000 001   2000995639 $7,500 to $9,999   Rural     905.     3      -2.46
## 10  2000 001   2000665487 $30,000 to $34,999 Urban    1185.    10      -0.371
## # i 31,503 more rows
```

```r
# Creative combination of these functions can allow you
# to calculate a descriptive statistic for any combination
# of groups and variables. For example, we could calculate
# the mean() for every year, grouped by Urban / Rural,
# for both INCOME (INC) and Household Weight (WGTHH):
household %>%

  mutate(INC = as.numeric(INCOME)) %>%

  group_by(YEAR,
           LAND_USE) %>%

  summarise(across(c(INC, WGTHH), ~ mean(.x, na.rm = TRUE)))
```

```
## `summarise()` has grouped output by 'YEAR'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 32 x 4
## # Groups:   YEAR [16]
##     YEAR LAND_USE   INC WGTHH
##    <dbl> <fct>    <dbl> <dbl>
##  1  2000 Urban     11.2  911.
##  2  2000 Rural     11.2  821.
##  3  2001 Urban     10.9  894.
##  4  2001 Rural     11.9  834.
##  5  2002 Urban     11.4  913.
##  6  2002 Rural     11.7  869.
##  7  2003 Urban     11.3  868.
##  8  2003 Rural     11.4  922.
##  9  2004 Urban     11.3  865.
## 10  2004 Rural     11.5  895.
## # i 22 more rows
```

```r
# ============================================================================ #

# Reshaping data is unnecessarily complicated, but you can achieve it
# with the tidyr package. Any type of grouped data can be presented in a
# long format, or a wide format.

# Long format includes repeating observations for each level of the group ID.
# Wide format includes repeating variables for each level of the group ID.

# For longitudinal panel data - with repeating observations of the same people,
# states, or other unit of analysis - the group ID is the "year" variable.

# Here, I am going to demonstrate reshaping the person-level data by household.

# Long (each row is a person):
(df <- person %>%
   select(IDPER, IDHH, YEARQ, V3020))
```

```
## # A tibble: 45,776 x 4
```

```
##    IDPER      IDHH       YEARQ V3020
##    <fct>      <fct>      <fct> <fct>
##  1 2000966984 2000993788 001   College
##  2 2000951294 2000993788 001   College
##  3 2000470356 2000993788 001   Elementary
##  4 2000205990 2000167846 001   College
##  5 2000361146 2000167846 001   College
##  6 2000879996 2000733306 001   High school
##  7 2000840437 2000733306 001   High school
##  8 2000494053 2000111147 001   High school
##  9 2000833192 2000111147 001   High school
## 10 2000365150 2000514983 001   High school
## # i 45,766 more rows
```

```
# Wide (each row is a household):
(df <- df %>%

   group_by(IDHH, YEARQ) %>%        # Group by IDHH and Year/Quarter

   mutate(HHM = 1) %>%              # Create a 'ticker' that
   mutate(HHM = cumsum(HHM)) %>%    # numbers household members

   pivot_wider(id_cols = c(IDHH, YEARQ),            # Reshape the data
               names_from = HHM,                    # to wide format
               values_from = c(IDPER, V3020)))
```

```
## # A tibble: 24,103 x 20
## # Groups:   IDHH, YEARQ [24,103]
##    IDHH    YEARQ IDPER_1 IDPER_2 IDPER_3 IDPER_4 IDPER_5 IDPER_6 IDPER_7 IDPER_8
##    <fct>   <fct> <fct>   <fct>   <fct>   <fct>   <fct>   <fct>   <fct>   <fct>
##  1 200099~ 001   200096~ 200095~ 200047~ <NA>    <NA>    <NA>    <NA>    <NA>
##  2 200016~ 001   200020~ 200036~ <NA>    <NA>    <NA>    <NA>    <NA>    <NA>
##  3 200073~ 001   200087~ 200084~ <NA>    <NA>    <NA>    <NA>    <NA>    <NA>
##  4 200011~ 001   200049~ 200083~ <NA>    <NA>    <NA>    <NA>    <NA>    <NA>
##  5 200051~ 001   200036~ <NA>    <NA>    <NA>    <NA>    <NA>    <NA>    <NA>
##  6 200030~ 001   200051~ <NA>    <NA>    <NA>    <NA>    <NA>    <NA>    <NA>
##  7 200016~ 001   200072~ 200031~ <NA>    <NA>    <NA>    <NA>    <NA>    <NA>
##  8 200069~ 001   200057~ <NA>    <NA>    <NA>    <NA>    <NA>    <NA>    <NA>
##  9 200099~ 001   200088~ <NA>    <NA>    <NA>    <NA>    <NA>    <NA>    <NA>
## 10 200066~ 001   200086~ 200060~ <NA>    <NA>    <NA>    <NA>    <NA>    <NA>
## # i 24,093 more rows
## # i 10 more variables: IDPER_9 <fct>, V3020_1 <fct>, V3020_2 <fct>,
## #   V3020_3 <fct>, V3020_4 <fct>, V3020_5 <fct>, V3020_6 <fct>, V3020_7 <fct>,
## #   V3020_8 <fct>, V3020_9 <fct>
```

```
# Back to Long!
(df <- df %>%
   pivot_longer(cols = IDPER_1:V3020_9,
                names_to = c(".value", "HH_MEMBER"),
                names_sep = "_") %>%
   drop_na(c(IDPER, V3020)))
```

```
## # A tibble: 43,971 x 5
```

```
## # Groups:   IDHH, YEARQ [23,052]
##    IDHH       YEARQ HH_MEMBER IDPER      V3020
##    <fct>      <fct> <chr>     <fct>      <fct>
##  1 2000993788 001   1         2000966984 College
##  2 2000993788 001   2         2000951294 College
##  3 2000993788 001   3         2000470356 Elementary
##  4 2000167846 001   1         2000205990 College
##  5 2000167846 001   2         2000361146 College
##  6 2000733306 001   1         2000879996 High school
##  7 2000733306 001   2         2000840437 High school
##  8 2000111147 001   1         2000494053 High school
##  9 2000111147 001   2         2000833192 High school
## 10 2000514983 001   1         2000365150 High school
## # i 43,961 more rows
```

```
# Compare the end result to our original subset:
## Start
person %>%
  select(IDPER, IDHH, YEARQ, V3020) %>%
  head()
```

```
## # A tibble: 6 x 4
##   IDPER      IDHH       YEARQ V3020
##   <fct>      <fct>      <fct> <fct>
## 1 2000966984 2000993788 001   College
## 2 2000951294 2000993788 001   College
## 3 2000470356 2000993788 001   Elementary
## 4 2000205990 2000167846 001   College
## 5 2000361146 2000167846 001   College
## 6 2000879996 2000733306 001   High school
```

```
## End
df %>% head()
```

```
## # A tibble: 6 x 5
## # Groups:   IDHH, YEARQ [3]
##   IDHH       YEARQ HH_MEMBER IDPER      V3020
##   <fct>      <fct> <chr>     <fct>      <fct>
## 1 2000993788 001   1         2000966984 College
## 2 2000993788 001   2         2000951294 College
## 3 2000993788 001   3         2000470356 Elementary
## 4 2000167846 001   1         2000205990 College
## 5 2000167846 001   2         2000361146 College
## 6 2000733306 001   1         2000879996 High school
```

4 Putting it all together

```
# In this final section, I am going to quickly demonstrate how you
# properly apply the NCVS sampling weights and create a weighted
# violent victimization count variable at the person-level,
```

```r
# by aggregating the incident-level data to the person-level.

# This code is based on the example code chunks found in the
# NCVS MSA Public-Use Data, 2000 - 2015 codebook. It will use
# a lot of the data management skills from earlier in this script,
# and also demonstrate how you merge two datasets.

# First, let's create a binary indicator of violent victimization:
incident <- incident %>%
  mutate(VIOLENT = as.numeric(TOC_RECODE) %in% 1:11,
         NONVIOLENT = !(as.numeric(TOC_RECODE) %in% 1:11),
         TYPE = ifelse(VIOLENT, "Violent", "Nonviolent"))

# Next, we need to (a) filter the incident data to only include violent
# victimizations, (b) group by person ID (and year/quarter), and
# (c) aggregate from the incident-level to the person-level.
(vbl <- incident %>%
   filter(VIOLENT) %>%
   group_by(YEARQ, IDPER) %>%
   summarise(WGTVIC_V = mean(WGTVIC),
             VIOLENT = sum(VIOLENT * SERIESWGT)))
```

```
## `summarise()` has grouped output by 'YEARQ'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 262 x 4
## # Groups:   YEARQ [63]
##    YEARQ IDPER      WGTVIC_V VIOLENT
##    <fct> <fct>         <dbl>   <dbl>
##  1 001   2000203259    1634.       1
##  2 001   2000324505    1886.       1
##  3 001   2000361146    2202.       1
##  4 001   2000486531    1694.       1
##  5 001   2000585923    3415.       1
##  6 001   2000638577    3165.       1
##  7 001   2000640200    3011.       1
##  8 001   2000664422    1626.       1
##  9 001   2000713056    2494.       1
## 10 001   2000778432    2423.       1
## # i 252 more rows
```

```r
(nvbl <- incident %>%
   filter(NONVIOLENT) %>%
   group_by(YEARQ, IDPER) %>%
   summarise(WGTVIC_NV = mean(WGTVIC),
             NONVIOLENT = sum(NONVIOLENT * SERIESWGT)))
```

```
## `summarise()` has grouped output by 'YEARQ'. You can override using the
## `.groups` argument.
```

```
## # A tibble: 1,376 x 4
## # Groups:   YEARQ [64]
```

```
##    YEARQ IDPER      WGTVIC_NV NONVIOLENT
##    <fct> <fct>          <dbl>      <dbl>
##  1 001   2000126695     1975.          1
##  2 001   2000220530     1893.          1
##  3 001   2000264479     3186.          1
##  4 001   2000276410     1612.          2
##  5 001   2000292627     1590.          1
##  6 001   2000335184     1673.          1
##  7 001   2000339862     1629.          2
##  8 001   2000345273     1913.          1
##  9 001   2000365150     2104.          1
## 10 001   2000387989     3395.          1
## # i 1,366 more rows
```

```
# Now, let's merge this violent victimization variable onto the
# person-level data. To do this, you use left_join() and specify
# the variables you want to match with the "by" option:
(person <- person %>%
   left_join(vbl, by = c("YEARQ", "IDPER")) %>%
   left_join(nvbl, by = c("YEARQ", "IDPER")) %>%
   mutate(VIOLENT = if_else(is.na(VIOLENT), 0, VIOLENT),
          NONVIOLENT = if_else(is.na(NONVIOLENT), 0, NONVIOLENT)))
```

```
## # A tibble: 45,776 x 13
##     YEAR YEARQ IDPER      IDHH   V3014 V3018 V3020 WGTPER   YIH WGTVIC_V VIOLENT
##    <dbl> <fct> <fct>      <fct>  <fct> <fct> <fct>  <dbl> <dbl>    <dbl>   <dbl>
##  1  2000 001   2000966984 20009~ 40-49 Male  Coll~  1063.    10       NA       0
##  2  2000 001   2000951294 20009~ 40-49 Fema~ Coll~   894.     9       NA       0
##  3  2000 001   2000470356 20009~ 12-17 Male  Elem~  1317.     9       NA       0
##  4  2000 001   2000205990 20001~ 35-39 Male  Coll~  1093.     4       NA       0
##  5  2000 001   2000361146 20001~ 30-34 Fema~ Coll~  1101.     4     2202.      1
##  6  2000 001   2000879996 20007~ 40-49 Male  High~  1063.     6       NA       0
##  7  2000 001   2000840437 20007~ 40-49 Fema~ High~   894.     6       NA       0
##  8  2000 001   2000494053 20001~ 35-39 Male  High~  1098.    11       NA       0
##  9  2000 001   2000833192 20001~ 40-49 Fema~ High~   899.    11       NA       0
## 10  2000 001   2000365150 20005~ 30-34 Fema~ High~  1101.     8       NA       0
## # i 45,766 more rows
## # i 2 more variables: WGTVIC_NV <dbl>, NONVIOLENT <dbl>
```

```
# Calculate the victimization adjustment factor (weights)
# per the NCVS codebook. Multiply this adjustment factor by
# the respective victimization variable (VIOLENT and NONVIOLENT)
# to create a weighted violent victimization variable ([N]VLNT_WGT):
(person <- person %>%
   mutate(ADJINC_WT_V = if_else(!is.na(WGTVIC_V), WGTVIC_V / WGTPER, 0),
          VLNT_WGT = VIOLENT * ADJINC_WT_V,
          ADJINC_WT_NV = if_else(!is.na(WGTVIC_NV), WGTVIC_NV / WGTPER, 0),
          NVLNT_WGT = NONVIOLENT * ADJINC_WT_NV))
```

```
## # A tibble: 45,776 x 17
##     YEAR YEARQ IDPER      IDHH   V3014 V3018 V3020 WGTPER   YIH WGTVIC_V VIOLENT
##    <dbl> <fct> <fct>      <fct>  <fct> <fct> <fct>  <dbl> <dbl>    <dbl>   <dbl>
##  1  2000 001   2000966984 20009~ 40-49 Male  Coll~  1063.    10       NA       0
```

```
##  2  2000 001   2000951294 20009~ 40-49 Fema~ Coll~   894.     9      NA        0
##  3  2000 001   2000470356 20009~ 12-17 Male  Elem~  1317.     9      NA        0
##  4  2000 001   2000205990 20001~ 35-39 Male  Coll~  1093.     4      NA        0
##  5  2000 001   2000361146 20001~ 30-34 Fema~ Coll~  1101.     4    2202.       1
##  6  2000 001   2000879996 20007~ 40-49 Male  High~  1063.     6      NA        0
##  7  2000 001   2000840437 20007~ 40-49 Fema~ High~   894.     6      NA        0
##  8  2000 001   2000494053 20001~ 35-39 Male  High~  1098.    11      NA        0
##  9  2000 001   2000833192 20001~ 40-49 Fema~ High~   899.    11      NA        0
## 10  2000 001   2000365150 20005~ 30-34 Fema~ High~  1101.     8      NA        0
## # i 45,766 more rows
## # i 6 more variables: WGTVIC_NV <dbl>, NONVIOLENT <dbl>, ADJINC_WT_V <dbl>,
## #   VLNT_WGT <dbl>, ADJINC_WT_NV <dbl>, NVLNT_WGT <dbl>
```

```
# Now we can use the VLNT_WGT to calculate a weighted average
# of the victimization count, or to ensure that

# We might want to quickly recode a variable so that it is more
# in line with our own operationalization for a project, or
# satisfies more statistical assumptions (e.g., normality):
(person <- person %>%
  mutate(EDUC = case_when(
    V3020 %in% c("Nev/kindergarten",
                 "Elementary") ~ "NHSE",

    V3020 %in% c("High school",
                 "12th grade (no diploma)",
                 "High school graduate (diploma or equivalent)") ~ "HSE",

    V3020 %in% c("Some college (no degree)",
                 "College",
                 "Bachelor degree") ~ "FE",

    V3020 %in% c("Master degree",
                 "Prof school degree") ~ "MA",

    V3020 %in% c("Doctorate degree") ~ "PHD",
  )))
```

```
## # A tibble: 45,776 x 18
##     YEAR YEARQ IDPER      IDHH   V3014 V3018 V3020 WGTPER   YIH WGTVIC_V VIOLENT
##    <dbl> <fct> <fct>      <fct>  <fct> <fct> <fct>  <dbl> <dbl>    <dbl>   <dbl>
##  1  2000 001   2000966984 20009~ 40-49 Male  Coll~  1063.    10      NA        0
##  2  2000 001   2000951294 20009~ 40-49 Fema~ Coll~   894.     9      NA        0
##  3  2000 001   2000470356 20009~ 12-17 Male  Elem~  1317.     9      NA        0
##  4  2000 001   2000205990 20001~ 35-39 Male  Coll~  1093.     4      NA        0
##  5  2000 001   2000361146 20001~ 30-34 Fema~ Coll~  1101.     4    2202.       1
##  6  2000 001   2000879996 20007~ 40-49 Male  High~  1063.     6      NA        0
##  7  2000 001   2000840437 20007~ 40-49 Fema~ High~   894.     6      NA        0
##  8  2000 001   2000494053 20001~ 35-39 Male  High~  1098.    11      NA        0
##  9  2000 001   2000833192 20001~ 40-49 Fema~ High~   899.    11      NA        0
## 10  2000 001   2000365150 20005~ 30-34 Fema~ High~  1101.     8      NA        0
## # i 45,766 more rows
## # i 7 more variables: WGTVIC_NV <dbl>, NONVIOLENT <dbl>, ADJINC_WT_V <dbl>,
## #   VLNT_WGT <dbl>, ADJINC_WT_NV <dbl>, NVLNT_WGT <dbl>, EDUC <chr>
```

```
person$EDUC <- factor(person$EDUC, levels = c("NHSE",
                                              "HSE",
                                              "FE",
                                              "MA",
                                              "PHD"))

# Finally, lets just rename some of the other variables we want to work with.
## First, we look at the data frame. We can see that AGE is the 5th variable,
## and SEX is the 6th variable.
person
```

```
## # A tibble: 45,776 x 18
##     YEAR YEARQ IDPER      IDHH   V3014 V3018 V3020 WGTPER   YIH WGTVIC_V VIOLENT
##    <dbl> <fct> <fct>      <fct>  <fct> <fct> <fct>  <dbl> <dbl>    <dbl>   <dbl>
##  1  2000 001   2000966984 20009~ 40-49 Male  Coll~  1063.    10       NA       0
##  2  2000 001   2000951294 20009~ 40-49 Fema~ Coll~   894.     9       NA       0
##  3  2000 001   2000470356 20009~ 12-17 Male  Elem~  1317.     9       NA       0
##  4  2000 001   2000205990 20001~ 35-39 Male  Coll~  1093.     4       NA       0
##  5  2000 001   2000361146 20001~ 30-34 Fema~ Coll~  1101.     4     2202.      1
##  6  2000 001   2000879996 20007~ 40-49 Male  High~  1063.     6       NA       0
##  7  2000 001   2000840437 20007~ 40-49 Fema~ High~   894.     6       NA       0
##  8  2000 001   2000494053 20001~ 35-39 Male  High~  1098.    11       NA       0
##  9  2000 001   2000833192 20001~ 40-49 Fema~ High~   899.    11       NA       0
## 10  2000 001   2000365150 20005~ 30-34 Fema~ High~  1101.     8       NA       0
## # i 45,766 more rows
## # i 7 more variables: WGTVIC_NV <dbl>, NONVIOLENT <dbl>, ADJINC_WT_V <dbl>,
## #   VLNT_WGT <dbl>, ADJINC_WT_NV <dbl>, NVLNT_WGT <dbl>, EDUC <fct>
```

```
## If you have too many variables to check manually, you can run this code:
which(colnames(person) == "V3014")  # AGE
```

```
## [1] 5
```

```
which(colnames(person) == "V3018")  # SEX
```

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

```
## Now you index colnames(person) by 5 and 6:
colnames(person)[5:6]
```

```
## [1] "V3014" "V3018"
```

```
## Then overwrite those variable names with simpler names:
colnames(person)[5:6] <- c("AGE", "SEX")

# Now, I'll output a sample of this data frame into the next module's Data
# folder, so we can use it when practicing Descriptive Statistics and Graphics:
saveRDS(incident, "../Data/incident.rds")
saveRDS(person, "../Data/person.rds")
saveRDS(household, "../Data/household.rds")
```