

Guidelines for Investigating, cleaning, and creating variables

EDUC 260A: Introduction to Programming and Data Management Using R

What we will do today

1. Introduction
2. Guidelines for exploratory data analysis (EDA)
3. Tools for exploratory data analysis (EDA)
4. Appendix: Skip patterns in survey data

1 Introduction

What is exploratory data analysis (EDA)?

The [Towards Data Science](#) website has a nice definition of EDA:

“Exploratory Data Analysis refers to the critical process of performing initial investigations on data so as to discover patterns, to spot anomalies, to test hypothesis and to check assumptions with the help of summary statistics”

This course focuses on “data management”:

- ▶ investigating and cleaning data for the purpose of creating analysis variables
- ▶ Basically, everything that happens **before** you conduct analyses

I think about “exploratory data analysis for data quality”

- ▶ Investigating values and patterns of variables from “input data”
- ▶ Identifying and cleaning errors or values that need to be changed
- ▶ Creating analysis variables
- ▶ Checking values of analysis variables against values of input variables

How we will teach exploratory data analysis

Will teach exploratory data analysis (EDA) in two sub-sections:

1. Provide “Guidelines for EDA”

- ▶ Less about coding, more about practices you should follow and mentality necessary to ensure high data quality

2. Introduce “Tools of EDA”:

- ▶ Demonstrate code to investigate variables and relationship between variables
- ▶ Most of these tools are just the application of programming skills you have already learned

Libraries

“Load” the package we will use today (output omitted)

▶ **you must run this code chunk after installing these packages**

```
library(tidyverse)
library(haven)
library(labelled)
```

If package not yet installed, then must install before you load. Install in “console” rather than .Rmd file

▶ Generic syntax: `install.packages("package_name")`

▶ Install “tidyverse”: `install.packages("tidyverse")`

Note: when we load package, name of package is not in quotes; but when we install package, name of package is in quotes:

▶ `install.packages("tidyverse")`

▶ `library(tidyverse)`

Data we will use

Use `read_dta()` function from `haven` to import Stata dataset into R

```
hsls <- read_dta(file="https://github.com/ozanj/rclass/raw/master/data/hsls/hs1
```

Let's examine the data [you **must** run this code chunk]

```
hsls %>% names()
hsls %>% names() %>% str()
hsls %>% names() %>% tolower() %>% str()

names(hs1s) <- tolower(names(hs1s)) # convert names to lowercase
names(hs1s)

str(hs1s) # ugh

str(hs1s$s3classes)
attributes(hs1s$s3classes)
typeof(hs1s$s3classes)
class(hs1s$s3classes)
```

Download the HSLS Codebook:

https://nces.ed.gov/pubs2014/2014361_AppendixI.pdf

Data we will use

First, Let's create a smaller version of the HSLs:09 dataset

```
#hsls %>% var_label()
hsls_small <- hsls %>%
  select(stu_id,x3univ1,x3sqstat,x4univ1,x4sqstat,s3classes,
         s3work,s3focus,s3clgft,s3workft,s3clgid,s3clgcntrl,
         s3clglvl,s3clgsel,s3clgstate,s3proglevel,x4evrappclg,
         x4evratndclg,x4atndclg16fb,x4ps1sector,x4ps1level,
         x4ps1ctrl,x4ps1select,x4refsector,x4reflevel,x4refctrl,
         x4refselect, x2sex,x2race,x2paredu,x2txmtscor,x4x2ses,x4x2sesq5)

names(hsls_small)
hsls_small %>% var_label()
```


2 Guidelines for exploratory data analysis (EDA)

Rule #1 for data quality: **DATA BETTER BE RIGHT**

- ▶ Cabrera, N. L., Milem, J. F., Jaquette, O., & Marx, R. (2014). Missing the (student achievement) forest for all the (political) trees: Empiricism and the Mexican American Studies controversy in Tucson. *American Educational Research Journal*, 51(6), 1084-1118.
 - ▶ Sometimes social justice is creating a GPA variable from course-level data
 - ▶ Very politically charged issue; would've been bad if we didn't get the data right

- ▶ Jaquette, O., & Parra, E. (2016). The Problem with the Delta Cost Project Database. *Research in Higher Education*, 57(5), 630-651
 - ▶ They didn't get the data right; I took them to task

- ▶ Jaquette, O. (2017). *State university no more: Out-of-state enrollment and the growing exclusion of high-achieving, low-income students at flagship public universities*. Lansdowne, VA: Jack Kent Cooke Foundation.
 - ▶ I didn't get the data right; I got taken to task

- ▶ Salazar, K. G., Jaquette, O., & Han, C. (2021). Coming Soon to a Neighborhood Near You? Off-Campus Recruiting by Public Research Universities. *American Educational Research Journal*, 58(6), 1270-1314
 - ▶ Karina/Crystal/Patricia spent thousands of hours getting the data right

Rule #1: **DATA BETTER BE RIGHT** (for grad students)

Researchers who develop a reputation for always getting the data right are the ones who always get research funding

- ▶ Grad students working on research projects are usually the front-line of getting the data right

The virtuous circle of getting the data right, delivering on deliverables, and inter-generational grad student opportunity

- ▶ you don't pay it forward later, you pay it forward now with the quality of work you do now
- ▶ sometimes you pay it forward even when you didn't get the opportunities you deserve

Challenges that arise

- ▶ Principal investigators who don't have respect for getting the data right
- ▶ Not enough time or resources to get the data right

Guidelines for “EDA for data quality”

Assume that your goal in “EDA for data quality” is to investigate “input” data sources and create “analysis variables”

- ▶ Usually, your analysis dataset will incorporate multiple sources of input data, including data you collect (primary data) and/or data collected by others (secondary data)

While this is not a linear process, these are the broad steps I follow

1. Understand how input data sources were created
 - ▶ e.g., when working with survey data, have survey questionnaire and codebooks on hand
 - ▶ watch out for skip patterns!!!
2. For each input data source, identify the “unit of analysis” and which combination of variables uniquely identify observations
3. Investigate patterns in input variables
4. Create analysis variable from input variable(s)
5. Verify that analysis variable is created correctly through descriptive statistics that compare values of input variable(s) against values of the analysis variable

Always be aware of missing values

- ▶ They will not always be coded as `NA` in input variables

“Unit of analysis” and which variables uniquely identify observations

“Unit of analysis” refers to “what does each observation represent” in an input data source

- ▶ If each obs represents a student, you have “student level data”
- ▶ If each obs represents a student-course, you have “student-course level data”
- ▶ If each obs represents a school, you have “school-level data”
- ▶ If each obs represents a school-year, you have “school-year level data”

How to identify unit of analysis

- ▶ data documentation
- ▶ investigating the data set

We will go over syntax for identifying unit of analysis in subsequent weeks

Rules for variable creation

Rules I follow for variable creation

1. Never modify “input variable”; instead create new variable based on input variable(s)
 - ▶ Always keep input variables used to create new variables
2. Investigate input variable(s) and relationship between input variables
3. Developing a plan for creation of analysis variable
 - ▶ e.g., for each possible value of input variables, what should value of analysis variable be?
4. Write code to create analysis variable
5. Run descriptive checks to verify new variables are constructed correctly
 - ▶ Can “comment out” these checks, but don’t delete them
6. Document new variables with notes and labels

Rules for variable creation

Task:

- Create analysis for variable ses quintile called `sesq5` based on `x4x2sesq5` that converts negative values to `NA`s

[some of the code here will be covered in the next section]

```
#investigate input variable
```

```
hsls_small %>% select(x4x2sesq5) %>% var_label()  
hsls_small %>% select(x4x2sesq5) %>% val_labels()  
hsls_small %>% select(x4x2sesq5) %>% count(x4x2sesq5)  
hsls_small %>% select(x4x2sesq5) %>% count(x4x2sesq5) %>% as_factor()
```

```
#create analysis variable
```

```
hsls_small_temp <- hsls_small %>%  
  mutate(sesq5=ifelse(x4x2sesq5==-8,NA,x4x2sesq5)) # approach 1  
hsls_small_temp <- hsls_small %>%  
  mutate(sesq5=ifelse(x4x2sesq5<0,NA,x4x2sesq5)) # approach 2
```

```
#verify
```

```
hsls_small_temp %>% group_by(x4x2sesq5) %>% count(sesq5)
```

Skip patterns in survey data

Pretty easy to create an analysis variable based on a single input variable

Harder to create analysis variables based on multiple input variables

- ▶ When working with survey data, even seemingly simple analysis variables require multiple input variables due to “skip patterns”

What are “skip patterns”?

- ▶ Response on a particular survey item determines whether respondent answers some set of subsequent questions
- ▶ What are some examples of this?

Key to working with skip patterns

- ▶ Have the survey questionnaire on hand
- ▶ Sometimes it appears that analysis variable requires only one input variable, but really depends on several input variables because of skip patterns
 - ▶ Don't just blindly turn “missing” and “skips” from survey data to `NAs` in your analysis variable
 - ▶ Rather, trace why these “missing” and “skips” appear and decide how they should be coded in your analysis variable

See appendix of this lecture for more detail on working with skip patterns

Problem set due next week

Assignment:

- ▶ create GPA from postsecondary transcript student-course level data
- ▶ last week's problem set created the input var: `numgrade_v2`
- ▶ this week you are responsible for developing plan to create GPA vars and for executing plan (rather than us giving you step-by-step equations)

Why this assignment?

1. Give you more practice investigating data, cleaning data, creating variables that require processing across rows
2. Real world example of “simple” task with complex data management needs

Data source: [National Longitudinal Study of 1972 \(NLS72\)](#)

- ▶ Follows 12th graders from 1972
 - ▶ Base year: 1972
 - ▶ Follow-up surveys in: 1973, 1974, 1976, 1979, 1986
 - ▶ Postsecondary transcripts collected in 1984
- ▶ Why use such an old survey for this assignment?
 - ▶ NLS72 predates data privacy agreements; transcript data publicly available

Recommendation for making decisions in presence of “dirty data”

- ▶ Spend some reasonable amount of time investigating data patterns
- ▶ Make decisions about what to when variable values seem problematic (e.g., a course is worth 30 credits)
 - ▶ **DON'T** make (subjective) decisions one observation at a time; rather, decide on rules that will apply across all observations
 - ▶ you can explain rationale for your decisions in comments of your code

3 Tools for exploratory data analysis (EDA)

Tools of EDA

To do EDA for data quality, must master the following tools:

- ▶ **Select, sort, filter, and print** in order to see data patterns, anomalies
 - ▶ Select and sort particular values of particular variables
 - ▶ Print particular values of particular variables
- ▶ **One-way descriptive analyses** (i.e., focus on one variable)
 - ▶ Descriptive analyses for continuous variables
 - ▶ Descriptive analyses for discrete/categorical variables
- ▶ **Two-way descriptive analyses** (relationship between two variables)
 - ▶ Categorical by categorical
 - ▶ Categorical by continuous
 - ▶ Continuous by continuous

Whenever using any of these tools, **pay close attention to missing values and how they are coded**

- ▶ Often, the “input” variables don’t code missing values as `NA`
- ▶ Especially when working with survey data, missing values coded as a negative number (e.g., `-9`, `-8`, `-4`) with different negative values representing different reasons for data being missing
- ▶ sometimes missing values coded as very high positive numbers
- ▶ Therefore, important to investigate input vars prior to creating analysis vars

Tools of EDA, caveat (using graphs to investigate variables)

Caveat on tools of EDA introduced in this lecture

- ▶ Usually you can investigate categorical variables pretty effectively by creating table of frequency counts
- ▶ More difficult to investigate continuous variables or relationship between continuous variable and another variable
 - ▶ too many different values for frequency counts
 - ▶ summary statistics (e.g., min, max) sometimes not very effective
- ▶ Graphs can be a great way to investigate continuous variables
 - ▶ also effective for investigating relationship between continuous variable and some other variable
 - ▶ but we do not introduce graphs until beginning of EDUC260B (Rclass2)

Caveat (using base R `ifelse()` rather than Tidyverse `if_else()`)

The below slides use the base R `ifelse()` function rather than Tidyverse `if_else()` function to create variables

- ▶ For example, below code uses `ifelse()` to create a version of the variable `s3classes` that replaces values less than zero with `NA`

- ▶ Note that the new variable we create `s3classes_na` is a numeric class variable, rather than labelled class, so it won't have value labels

```
hsls_small %>% mutate(s3classes_na=ifelse(s3classes<0,NA,s3classes)) %>%  
select(s3classes,s3classes_na) %>% glimpse()
```

Why not use Tidyverse `if_else()` ?

- ▶ below code won't work because `if_else(<condition>,<true>,<false>)` expects values assigned when condition is `TRUE` to have the same *class* as values assigned when condition is `FALSE`

- ▶ but there is no variation of `NA` that has the labelled class, so R throws an error

```
hsls_small %>% mutate(s3classes_na=if_else(s3classes<0,NA,s3classes))  
hsls_small %>% mutate(s3classes_na=if_else(s3classes<0,NA_real_,s3classes))
```

Caveat (using base R `ifelse()` rather than Tidyverse `if_else()`)

This “stack overflow” ([LINK](#)) post explains some solutions to get around the problem

- ▶ We don't implement any of these solutions because we want to keep the code as simple as possible

The downside of using `ifelse` is that the new variable has numeric class rather than labelled class; we lose value labels (and variable labels)

- ▶ One way around this, is to assign the `labelled()` function from “haven” package to assign “labels” attribute from the input variable

```
hsls_small_temp <- hsls_small %>% mutate(s3classes_na=ifelse(s3classes<0,NA,s3c  
attributes(hsls_small_temp$s3classes_na) # null  
attributes(hsls_small_temp$s3classes) # labelled class  
  
hsls_small_temp$s3classes_na <- labelled(  
  hsls_small_temp$s3classes_na,  
  labels = attr(x=hsls_small_temp$s3classes, which = "labels"),  
  label = attr(x=hsls_small_temp$s3classes, which = "label")  
)  
hsls_small_temp %>% select(s3classes,s3classes_na) %>% glimpse()  
hsls_small_temp %>% count(s3classes_na)  
rm(hsls_small_temp)
```

Tools of EDA: select, sort, filter, and print

We've already know `select()` , `arrange()` , `filter()`

Select, sort, and print specific vars

#sort and print

```
hsls_small %>% arrange(desc(stu_id)) %>%  
  select(stu_id,x3univ1,x3sqstat,s3classes,s3clglvl)
```

#investigate variable attributes

```
hsls_small %>% arrange(desc(stu_id)) %>%  
  select(stu_id,x3univ1,x3sqstat,s3classes,s3clglvl) %>% str()
```

#print observations with value labels rather than variable values

```
hsls_small %>% arrange(desc(stu_id)) %>%  
  select(stu_id,x3univ1,x3sqstat,s3classes,s3clglvl) %>% as_factor()
```

Sometimes helpful to increase the number of observations printed

```
class(hsls_small) #it's a tibble, which is the "tidyverse" version of a data frame  
options(tibble.print_min=50)  
# execute this in console  
hsls_small %>% arrange(desc(stu_id)) %>%  
  select(stu_id,x3univ1,x3sqstat,s3classes,s3clglvl)  
options(tibble.print_min=10) # set default printing back to 10 lines
```

One-way descriptive stats for continuous vars, Base R approach [SKIP]

```
mean(hs1s_small$x2txmtscor)
sd(hs1s_small$x2txmtscor)

#Careful: summary stats include value of -8!
min(hs1s_small$x2txmtscor)
max(hs1s_small$x2txmtscor)
```

Be careful with NA values

```
#Create variable replacing -8 with NA
hs1s_small_temp <- hs1s_small %>%
  mutate(x2txmtscorv2=ifelse(x2txmtscor==-8,NA,x2txmtscor))
hs1s_small_temp %>% filter(is.na(x2txmtscorv2)) %>% count(x2txmtscorv2)

mean(hs1s_small_temp$x2txmtscorv2)
mean(hs1s_small_temp$x2txmtscorv2, na.rm=TRUE)
rm(hs1s_small_temp)
```


One-way descriptive stats for continuous vars, Tidyverse approach

Use `summarise_at()`, a variation of `summarise()`, to make descriptive stats

- ▶ `.args=list(na.rm=TRUE)` = a named list of additional arguments to be added to all function calls

Task:

- ▶ calculate descriptive stats for `x2txmtscor`, math test score

```
##?summarise_at
hsls_small %>% select(x2txmtscor) %>% var_label()
#> $x2txmtscor
#> [1] "X2 Mathematics standardized theta score"
hsls_small %>%
  summarise_at(
    .vars = vars(x2txmtscor),
    .funs = funs(mean, sd, min, max, .args=list(na.rm=TRUE))
  )
#> # A tibble: 1 x 4
#>   mean    sd   min   max
#>   <dbl> <dbl> <dbl> <dbl>
#> 1  44.1  21.8    -8  84.9
```

One-way descriptive stats for continuous vars, Tidyverse approach

Can calculate descriptive stats for more than one variable at a time

Task:

- ▶ calculate descriptive stats for `x2txmtscor`, math test score, and `x4x2ses`, socioeconomic index score

```
hsls_small %>% select(x2txmtscor,x4x2ses) %>% var_label()  
#> $x2txmtscor  
#> [1] "X2 Mathematics standardized theta score"  
#>  
#> $x4x2ses  
#> [1] "X4 Revised X2 Socio-economic status composite"  
  
hsls_small %>%  
  summarise_at(  
    .vars = vars(x2txmtscor,x4x2ses),  
    .funs = funs(mean, sd, min, max, .args=list(na.rm=TRUE))  
  )  
#> # A tibble: 1 x 8  
#>   x2txmtscor_mean x4x2ses_mean x2txmtscor_sd x4x2ses_sd x2txmtscor_min  
#>   <dbl>         <dbl>         <dbl>         <dbl>         <dbl>  
#> 1         44.1        -0.802         21.8         2.63         -8  
#> # ... with 3 more variables: x4x2ses_min <dbl>, x2txmtscor_max <dbl>,  
#> #   x4x2ses_max <dbl>
```

One-way descriptive stats for continuous vars, Tidyverse approach

"Input vars" in survey data often have negative values for missing/skips

```
hsls_small %>% filter(x2txmtscor<0) %>% count(x2txmtscor)
```

R includes those negative values when calculating stats; you don't want this

► Solution: create version of variable that replaces negative values with NA

```
hsls_small %>% mutate(x2txmtscor_na=ifelse(x2txmtscor<0,NA,x2txmtscor)) %>%  
  summarise_at(  
    .vars = vars(x2txmtscor_na),  
    .funs = funs(mean, sd, min, max, .args=list(na.rm=TRUE))  
  )  
#> # A tibble: 1 x 4  
#>   mean    sd   min   max  
#>   <dbl> <dbl> <dbl> <dbl>  
#> 1  51.5  10.2  22.2  84.9
```

What if you didn't include `.args=list(na.rm=TRUE)` ?

```
hsls_small %>% mutate(x2txmtscor_na=ifelse(x2txmtscor<0,NA,x2txmtscor)) %>%  
  summarise_at(  
    .vars = vars(x2txmtscor_na),  
    .funs = funs(mean, sd, min, max))  
#> # A tibble: 1 x 4  
#>   mean    sd   min   max  
#>   <dbl> <dbl> <dbl> <dbl>  
#> 1    NA    NA    NA    NA
```

One-way descriptive stats for continuous vars, Tidyverse approach

How to identify these missing/skip values if you don't have a codebook?

- ▶ `count()` combined with `filter()` helpful for finding extreme values of continuous vars, which are often associated with missing or skip

```
#variable x2txmtscor
hsls_small %>% filter(x2txmtscor<0) %>%
  count(x2txmtscor)
#> # A tibble: 1 x 2
#>   x2txmtscor      n
#>   <dbl> <int>
#> 1      -8  2909

#variable s3clglvl
hsls_small %>% select(s3clglvl) %>% var_label()
#> $s3clglvl
#> [1] "S3 Enrolled college IPEDS level"

hsls_small %>% filter(s3clglvl<0) %>%
  count(s3clglvl)
#> # A tibble: 3 x 2
#>   s3clglvl      n
#>   <dbl+lbl> <int>
#> 1 -9 [Missing]    487
#> 2 -8 [Unit non-response] 4945
#> 3 -7 [Item legitimate skip/NA] 5022
```

One-way descriptive stats student exercise

1. Using the object `hsls`, identify variable type, variable class, and check the variable values and value labels of `x4ps1start`
 - ▶ variable `x4ps1start` identifies month and year student first started postsecondary education
 - ▶ **Note:** This variable is a bit counterintuitive.
 - ▶ e.g., the value `201105` refers to May 2011
2. Get a frequency count of the variable `x4ps1start`
3. Get a frequency count of the variable, but this time only observations that have negative values **hint:** use `filter()`
4. Create a new version of the variable `x4ps1start_na` that replaces negative values with NAs and use `summarise_at()` to get the min and max value.

One-way descriptive stats student exercise solutions

1. Using the object `hs1s`, identify variable type, variable class, and check the variable `x4ps1start` and value labels of `x4ps1start`

```
typeof(hs1s$x4ps1start)
#> [1] "double"
class(hs1s$x4ps1start)
#> [1] "haven_labelled" "vctrs_vctr"      "double"

hs1s %>% select(x4ps1start) %>% var_label()
#> $x4ps1start
#> [1] "X4 Month and year of enrollment at first postsecondary institution"

hs1s %>% select(x4ps1start) %>% val_labels()
#> $x4ps1start
#>                                     Missing
#>                                     -9
#>                                     Unit non-response
#>                                     -8
#>                                     Item legitimate skip/NA
#>                                     -7
#>                                     Component not applicable
#>                                     -6
#> Item not administered: abbreviated interview
#>                                     -4
#>                                     Carry through missing
#>                                     -3
```

One-way descriptive stats student exercise solutions

2. Get a frequency count of the variable `x4ps1start`

```
hsls %>%  
  count(x4ps1start)  
#> # A tibble: 9 x 2  
#>           x4ps1start      n  
#>           <dbl+lbl> <int>  
#> 1      -9 [Missing]      107  
#> 2      -8 [Unit non-response] 6168  
#> 3      -7 [Item legitimate skip/NA] 4281  
#> 4 201100           57  
#> 5 201200           206  
#> 6 201300          10800  
#> 7 201400          1295  
#> 8 201500           471  
#> 9 201600           118
```

One-way descriptive stats student exercise solutions

3. Get a frequency count of the variable, but this time only observations that have negative values **hint**: use `filter()`

```
hsls %>%  
  filter(x4ps1start<0) %>%  
  count(x4ps1start)  
#> # A tibble: 3 x 2  
#>           x4ps1start      n  
#>           <dbl> <int>  
#> 1 -9 [Missing]      107  
#> 2 -8 [Unit non-response] 6168  
#> 3 -7 [Item legitimate skip/NA] 4281
```


One-way descriptive stats student exercise solutions

4. Create a new version `x4ps1start_na` of the variable `x4ps1start` that replaces negative values with NAs and use `summarise_at()` to get the min and max value.

```
hsls %>% mutate(x4ps1start_na=ifelse(x4ps1start<0,NA,x4ps1start)) %>%  
  summarise_at(  
    .vars = vars(x4ps1start_na),  
    .funs = funs(min, max, .args=list(na.rm=TRUE))  
  )  
#> # A tibble: 1 x 2  
#>       min      max  
#>   <dbl>   <dbl>  
#> 1 201100 201600
```

One-way descriptive stats for discrete/categorical vars, Tidyverse approach

Use `count()` to investigate values of discrete or categorical variables

For variables where `class==labelled`

```
class(hs1s_small$s3classes)
attributes(hs1s_small$s3classes)
#show counts of variable values
hs1s_small %>% count(s3classes) #print in console to show both
#show counts of value labels
hs1s_small %>% count(s3classes) %>% as_factor()
```

► I like `count()` because the default setting is to show `NA` values too!

```
hs1s_small %>% mutate(s3classes_na=ifelse(s3classes<0,NA,s3classes)) %>%
  count(s3classes_na)
```

Simultaneously show both values and value labels on count tables for `class==labelled` if entered into console

► This requires some concepts/functions we haven't introduced [SKIP]

```
hs1s_small %>% count(s3classes)
y <- hs1s_small %>% count(s3classes) %>% as_factor()
bind_cols(x[,1], y) #wont show in updated R
```

Relationship between variables, categorical by categorical

Two-way frequency table, called “cross tabulation”, important for data quality

- ▶ When you create categorical analysis var from single categorical “input” var
 - ▶ Two-way tables show us whether we did this correctly
- ▶ Two-way tables helpful for understanding skip patterns in surveys

key to syntax

- ▶ `df_name %>% group_by(var1) %>% count(var2)` **OR**
- ▶ `df_name %>% count(var1,var2)`
- ▶ play around with which variable is `var1` and which variable is `var2`

Relationship between variables, categorical by categorical

Task: Create a two-way table between `s3classes` and `s3clglvl`

► Investigate variables

```
hsls_small %>% select(s3classes,s3clglvl) %>% var_label()  
hsls_small %>% select(s3classes,s3clglvl) %>% val_labels()
```

► Create two-way table

```
hsls_small %>% group_by(s3classes) %>% count(s3clglvl) # show values  
hsls_small %>% count(s3classes,s3clglvl)  
#hsls_small %>% group_by(s3classes) %>% count(s3clglvl) %>% as_factor() # show v
```

► Are these objects the same?

```
hsls_small %>% group_by(s3classes) %>% count(s3clglvl) %>% glimpse()  
#> Rows: 8  
#> Columns: 3  
#> Groups: s3classes [5]  
#> $ s3classes <dbl+lbl> -9, -8, 1, 1, 1, 1, 2, 3  
#> $ s3clglvl <dbl+lbl> -9, -8, -9, 1, 2, 3, -7, -7  
#> $ n <int> 59, 4945, 428, 8894, 3929, 226, 3401, 1621  
hsls_small %>% count(s3classes,s3clglvl) %>% glimpse()  
#> Rows: 8  
#> Columns: 3  
#> $ s3classes <dbl+lbl> -9, -8, 1, 1, 1, 1, 2, 3  
#> $ s3clglvl <dbl+lbl> -9, -8, -9, 1, 2, 3, -7, -7
```

Relationship between variables, categorical by categorical

Two-way frequency table, also called “cross tabulation”

Task:

- ▶ Create a version of `s3classes` called `s3classes_na` that changes negative values to `NA`
- ▶ Create a two-way table between `s3classes_na` and `s3clglvl`

```
hsls_small %>%  
  mutate(s3classes_na=ifelse(s3classes<0,NA,s3classes)) %>%  
  group_by(s3classes_na) %>% count(s3clglvl)
```

```
hsls_small %>%  
  mutate(s3classes_na=ifelse(s3classes<0,NA,s3classes)) %>%  
  count(s3classes_na, s3clglvl)
```

#example where we create some NA obs in the second variable

```
hsls_small %>%  
  mutate(s3classes_na=ifelse(s3classes<0,NA,s3classes),  
         s3clglvl_na=ifelse(s3clglvl== -7,NA,s3clglvl)) %>%  
  group_by(s3classes_na) %>% count(s3clglvl_na)
```

```
hsls_small %>%  
  mutate(s3classes_na=ifelse(s3classes<0,NA,s3classes),  
         s3clglvl_na=ifelse(s3clglvl== -7,NA,s3clglvl)) %>%
```

Relationship between variables, categorical by categorical [SKIP]

Tables above are pretty ugly

Use the `spread()` function from `tidyr` package to create table with one variable as columns and the other variable as rows

► The variable you place in `spread()` will be columns

► We learn `spread()` function in a subsequent lecture

```
hsls_small %>% group_by(s3classes) %>% count(s3clglvl) %>%  
  spread(s3classes, n)
```

```
hsls_small %>% group_by(s3classes) %>% count(s3clglvl) %>%  
  as_factor() %>% spread(s3classes, n)
```

```
hsls_small %>% group_by(s3classes) %>% count(s3clglvl) %>%  
  as_factor() %>% spread(s3clglvl, n)
```

Relationship between variables, categorical by continuous

Investigating relationship between multiple variables is a little tougher when at least one of the variables is continuous

Conditional mean (like regression with continuous Y and one categorical X):

- ▶ Shows average values of continuous variables within groups
- ▶ Groups are defined by your categorical variable(s)

key to syntax



```
group_by(categorical_var) %>% summarise_at(.vars = vars(continuous_var))
```

Relationship between variables, categorical by continuous

Task

- Calculate mean math score, `x2txmtscor`, for each value of parental education, `x2paredu`

```
#first, investigate parental education [print in console]
hsls_small %>% count(x2paredu)
```

```
# using dplyr to get average math score by parental education level [print in console]
hsls_small %>% group_by(x2paredu) %>%
  summarise_at(.vars = vars(x2txmtscor),
               .funs = funs(mean, .args = list(na.rm = TRUE)))

#> # A tibble: 8 x 2
#>                                     x2paredu x2txmtscor
#>                                     <dbl> <dbl>
#> 1 -8 [Unit non-response]
#> 2  1 [Less than high school]
#> 3  2 [High school diploma or GED or alternative HS credential]
#> 4  3 [Certificate/diploma from school providing occupational training]
#> 5  4 [Associate's degree]
#> 6  5 [Bachelor's degree]
#> 7  6 [Master's degree]
#> 8  7 [Ph.D/M.D/Law/other high lvl prof degree]
```


Relationship between variables, categorical by continuous

Task

- Calculate mean math score, `x2txmtscor`, for each value of `x2paredu`

For checking data quality, helpful to calculate other stats besides mean

```
hsls_small %>% group_by(x2paredu) %>% #[print in console]  
  summarise_at(.vars = vars(x2txmtscor),  
               .funs = funs(mean, min, max, .args = list(na.rm = TRUE)))
```

Always Investigate presence of missing/skip values

```
hsls_small %>% filter(x2paredu<0) %>% count(x2paredu)  
hsls_small %>% filter(x2txmtscor<0) %>% count(x2txmtscor)
```

Replace `-8` with `NA` and re-calculate conditional stats

```
hsls_small %>%  
  mutate(x2paredu_na=ifelse(x2paredu<0,NA,x2paredu),  
         x2txmtscor_na=ifelse(x2txmtscor<0,NA,x2txmtscor)) %>%  
  group_by(x2paredu_na) %>%  
  summarise_at(.vars = vars(x2txmtscor_na),  
               .funs = funs(mean, min, max, .args = list(na.rm = TRUE))) %>%  
  as_factor()  
#> Warning in min(x2txmtscor_na, na.rm = TRUE): no non-missing arguments to min;  
#> returning Inf  
#> Warning in max(x2txmtscor_na, na.rm = TRUE): no non-missing arguments to max;  
#> returning -Inf
```

Student exercise

Can use same approach to calculate conditional mean by multiple `group_by()` variables

► Just add additional variables within `group_by()`

1. Calculate mean math test score (`x2txmtscor`), for each combination of parental education (`x2paredu`) and sex (`x2sex`).

Student exercise solution

1. Calculate mean math test score (`x2txmtscor`), for each combination of parental education (`x2paredu`) and sex (`x2sex`)

```
#hsls_small %>% count(x2sex)

hsls_small %>%
  group_by(x2paredu, x2sex) %>%
  summarise_at(.vars = vars(x2txmtscor),
               .funs = funs(mean, .args = list(na.rm = TRUE))) %>%
  as_factor()
```

4 Appendix: Skip patterns in survey data

What are skip patterns

Pretty easy to create an analysis variable based on a single input variable

Harder to create analysis variables based on multiple input variables

- ▶ When working with survey data, even seemingly simple analysis variables require multiple input variables due to “skip patterns”

What are “skip patterns”?

- ▶ Response on a particular survey item determines whether respondent answers some set of subsequent questions
- ▶ What are some examples of this?

Key to working with skip patterns

- ▶ Have the survey questionnaire on hand
- ▶ Sometimes it appears that analysis variable requires only one input variable, but really depends on several input variables because of skip patterns
 - ▶ Don't just blindly turn “missing” and “skips” from survey data to `NAs` in your analysis variable
 - ▶ Rather, trace why these “missing” and “skips” appear and decide how they should be coded in your analysis variable

Creating analysis variables in the presence of skip patterns

Task: Create a measure of “level” of postsecondary institution attended in 2013 from HSLS:09 survey data

- ▶ “level” is highest award-level of the postsecondary institution
 - ▶ e.g., if highest award is associate’s degree (a two-year degree), then ‘level==2’
- ▶ The measure, `pselev2013`, should have following [non-missing] values:
 1. Not attending postsecondary education institution
 2. Attending a 2-year or less-than-2-year institution
 3. Attending 4-year or greater-than-4year institution

Background info:

- ▶ In “2013 Update” of HSLS:09, students asked about college attendance
 - ▶ Variables from student responses to “2013 Update” have prefix `s3`
- ▶ Survey questionnaire for 2013 update can be found [HERE](#)
- ▶ The “online codebook” website [HERE](#) has info about specific variables
- ▶ Measure has 3 input variables [usually must figure this out yourself]:
 1. `x3sqstat` : “X3 Student questionnaire status”
 2. `s3clclasses` : “S3 B01A Taking postsecondary classes as of Nov 1 2013”
 3. `s3clglvl` : “S3 Enrolled college IPEDS level”

```
hsls_small %>% select(x3sqstat,s3clclasses,s3clglvl) %>% var_label()
```

You won’t have time to complete this task, but develop a plan for the task and get as far as you can

Creating analysis variables in the presence of skip patterns

Step 1a: Investigate each input variable separately

```
#variable labels
```

```
hsls_small %>% select(x3sqstat,s3classes,s3clglvl) %>% var_label()
```

```
hsls_small %>% count(x3sqstat)
```

```
hsls_small %>% count(x3sqstat) %>% as_factor()
```

```
hsls_small %>% count(s3classes)
```

```
hsls_small %>% count(s3classes) %>% as_factor()
```

```
hsls_small %>% count(s3clglvl)
```

```
hsls_small %>% count(s3clglvl) %>% as_factor()
```

Creating analysis variables in the presence of skip patterns

Step 1b: Investigate relationship between input variables

#x3sqstate and s3classes

```
hsls_small %>% group_by(x3sqstat) %>% count(s3classes)
hsls_small %>% group_by(x3sqstat) %>% count(s3classes) %>% as_factor()

hsls_small %>% filter(x3sqstat==8) %>% count(s3classes)
hsls_small %>% filter(x3sqstat==8) %>% count(s3classes==8)
hsls_small %>% filter(x3sqstat !=8) %>% count(s3classes)
```

#x3sqstate, s3classes and s3clglvl

```
hsls_small %>% group_by(s3classes) %>% count(s3clglvl)
hsls_small %>% group_by(s3classes) %>% count(s3clglvl) %>% as_factor()
```

#add filter for whether student did not respond to X3 questionnaire

```
hsls_small %>% filter(x3sqstat==8) %>% group_by(s3classes) %>% count(s3clglvl)
hsls_small %>% filter(x3sqstat !=8) %>% group_by(s3classes) %>% count(s3clglvl)
```

#continued on the next page

Creating analysis variables in the presence of skip patterns

Step 1b: Investigate relationship between input variables continued...

```
#add filter for s3classes is "missing" [-9]
hsls_small %>% filter(x3sqstat !=8,s3classes== -9) %>% group_by(s3classes) %>%
  count(s3clglvl)
#> # A tibble: 1 x 3
#> # Groups:   s3classes [1]
#>   s3classes s3clglvl     n
#>   <dbl+lbl> <dbl+lbl> <int>
#> 1 -9 [Missing] -9 [Missing]    59
hsls_small %>% filter(x3sqstat !=8,s3classes!= -9) %>% group_by(s3classes) %>%
  count(s3clglvl)
#> # A tibble: 6 x 3
#> # Groups:   s3classes [3]
#>   s3classes s3clglvl     n
#>   <dbl+lbl> <dbl+lbl> <int>
#> 1 1 [Yes]      -9 [Missing]    428
#> 2 1 [Yes]       1 [4 or more years]    8894
#> 3 1 [Yes]       2 [At least 2 but less than 4 years]    3929
#> 4 1 [Yes]       3 [Less than 2 years (below associate)]    226
#> 5 2 [No]       -7 [Item legitimate skip/NA]    3401
#> 6 3 [Don't know] -7 [Item legitimate skip/NA]    1621

#add filter for s3classes equal to "no" or "don't know"
hsls_small %>% filter(x3sqstat !=8,s3classes!= -9, s3classes %in% c(2,3)) %>%
  group_by(s3classes) %>% count(s3clglvl)
#> # A tibble: 2 x 3
```