Module 7 problem set solutions

INSERT YOUR NAME HERE

Contents

Problem Set instructions

Overview

- Using the NLS72 course-level dataset, your assigmnent is to create the following GPA variables:
  - institution-level (i.e., transcript-level) GPA variable
  - term-level GPA variable

NLS72 Codebook and Supplemental Addendum

- https://github.com/ozanj/rclass/blob/master/data/nls72/NLS72_codebook.pdf
- https://github.com/ozanj/rclass/blob/master/data/nls72/NLS72_suppaddendum.pdf

General Instructions

- Don't make changes to "input" variables; instead, create a new variable(s)
- You are responsible for deciding what data investigations to conduct (e.g., conditional statements, frequency counts, etc.)
- Keep the data investigations you want me to see; though you might want to comment out very long lists of observations
- Reference the NLS72 codebook and supplemental addendum when needed. Document your rationale for data decisions via comments and provide reference page numbers from codebook or addendum when useful.
- Whenever you create a new variable, run checks to make sure variable created correctly (e.g., counts, cross-tabulations, assertions)
- As you work towards creating the gpa variable(s) you will create several new "input" variables
- Below, you will find some "header" instructions/hints in regards to important steps you should be completing in process of creating these gpa variables
- Whenever relevant, you can insert code you developed from the previous problem set as part of your answers for this problem set

Load library and data

```
#install.packages("tidyverse") #uncomment if you haven't installed these packaged
#install.packages("haven")
#install.packages("labelled")
library(tidyverse)
#> -- 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 error
library(haven)
library(labelled)
```

Open data

Run the code chunk below

```
rm(list = ls()) # remove all objects
#getwd()
#list.files("../../../documents/rclass/data/nls72") # list files in directory w/ NLS data

#Read Stata data into R using read_data() function from haven package
nls_crs <- read_dta(file="https://github.com/ozanj/rclass/raw/master/data/nls72/nls72petscrs_v2.dta", e
```

Part I: Investigate data

First stage of creating an analysis dataset is conducting a thorough investigation of the "input" dataset(s) and an investigation of key variables. This often takes a long time.

- Preliminary investigation of data frame

```
names(nls_crs)
glimpse(nls_crs)
str(nls_crs)
head(nls_crs)
nls_crs %>% var_label() # view variable labels
```

- Perform one-way investigations following input variables:
  - `transnum`, `termnum`, `crsecred`, `gradtype`, `crsgrada`, `crsgradb`

```
#Investigate variable transnum
class(nls_crs$transnum)
nls_crs%>% select(transnum) %>% var_label() # view variable labels
nls_crs%>% count(transnum)


#Check that sum of transnum equals number of rows in dataset
nls_crs %>%
  group_by(transnum) %>% #grouping by transum
  summarise(count_transum = n()) %>% #count for each value of transum
  ungroup() %>% #ungroup
  mutate(total_obs = sum(count_transum)) #Get the sum of count to check that it equals the number of ob


#Investigate variable termnum
class(nls_crs$termnum)
nls_crs%>% select(termnum) %>% var_label() # view variable labels
nls_crs%>% count(termnum)


#Investigate course credits
#glimpse(nls_crs)
class(nls_crs$crsecred)
nls_crs%>% select(crsecred) %>% var_label() # view variable labels
options(tibble.print_min=50)
nls_crs%>% count(crsecred)
nls_crs %>% #run some descriptive stats
  summarise_at(
    .vars = vars(crsecred),
    .funs = funs(min, max, .args=list(na.rm=TRUE))
)
#> Warning: `funs()` was deprecated in dplyr 0.8.0.
#> i Please use a list of either functions or lambdas:
#>
#> # Simple named list: list(mean = mean, median = median)
#>
#> # Auto named with `tibble::lst()`: tibble::lst(mean, median)
#>
#> # Using lambdas list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
#> Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
#> generated.
#investigate high values of crsecred
nls_crs%>% filter(crsecred>=100) %>% count(crsecred) # frequency table of crsecred
nls_crs%>% filter(crsecred==999) # printing some observations for specific values of crsecred
nls_crs%>% filter(crsecred>=999) %>% count(crsecred) #


#Investigate gradtype
class(nls_crs$gradtype) # labelled
#glimpse(nls_crs)
nls_crs%>% select(gradtype) %>% var_label() # view variable labels
nls_crs%>% select(gradtype) %>% val_labels() # view value labels on variable
nls_crs %>% count(gradtype) #freq count of values
nls_crs %>% count(gradtype) %>% as_factor() #freq count with value labels


#Run one-way investigation for crsgrada
#crsgrada
#glimpse(nls_crs)
class(nls_crs$crsgrada) #character
nls_crs%>% select(crsgrada) %>% var_label() # view variable labels
nls_crs%>% count(crsgrada)
```

```r
nls_crs%>% filter(crsgrada %in% c("99", "AU", "CR", "I", "NO", "P", "S", "U", "W", "WP")) #printing som

#Run one-way investigation for crsgradb
#crsgradb
class(nls_crs$crsgradb) #numeric
nls_crs%>% select(crsgradb) %>% var_label()
nls_crs%>% count(crsgradb)
nls_crs %>% #run some descriptive stats
  summarise_at(
    .vars = vars(crsgradb),
    .funs = funs(min, max, .args=list(na.rm=TRUE))
)
#> Warning: `funs()` was deprecated in dplyr 0.8.0.
#> i Please use a list of either functions or lambdas:
#>
#> # Simple named list: list(mean = mean, median = median)
#>
#> # Auto named with `tibble::lst()`: tibble::lst(mean, median)
#>
#> # Using lambdas list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
#> Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
#> generated.
nls_crs%>% filter(crsgradb>100) %>% count(crsgradb)
```

- Investigate the relationship between the following pairs of variables:
  - `gradtype` and `crsgrada`
  - `gradtype` and `crsgradb`
  - `crsecred` and `gradtype`

```r
options(tibble.print_min=50)
#Investigate gradtype, crsgrada, crsgradb

#some tabulations for different values of gradtype and crsgrada
nls_crs %>% group_by(gradtype) %>% count(crsgrada) # cross tab of vars gradtype & crsgrada
nls_crs %>% group_by(gradtype) %>% count(crsgrada) %>% as_factor() #cross tab this time show value labe

nls_crs%>% filter(gradtype==1) %>% count(crsgrada) # letter grade
nls_crs%>% filter(gradtype==2) %>% count(crsgrada) # numeric grade
nls_crs%>% filter(gradtype==9) %>% count(crsgrada) # missing


#some tabulations for different values of gradtype and crsgradb
nls_crs %>% group_by(gradtype) %>% count(crsgradb) # cross tab of vars gradtype & crsgradb
nls_crs %>% group_by(gradtype) %>% count(crsgradb) %>% as_factor() #cross tab this time show value labe
nls_crs %>% group_by(gradtype) %>%
  summarise_at(.vars = vars(crsgradb),
               .funs = funs(min, max, .args = list(na.rm = TRUE))) %>%
  as_factor()
#> Warning: `funs()` was deprecated in dplyr 0.8.0.
#> i Please use a list of either functions or lambdas:
#>
#> # Simple named list: list(mean = mean, median = median)
#>
#> # Auto named with `tibble::lst()`: tibble::lst(mean, median)
```

```
#>
#> # Using lambdas list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
#> Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
#> generated.

nls_crs%>% filter(gradtype==1) %>% count(crsgradb) # letter grade
nls_crs%>% filter(gradtype==2) %>% count(crsgradb) # numeric grade
nls_crs%>% filter(gradtype==9) %>% count(crsgradb) # missing


#Run tabulations for different values of gradtype and crsecred

#some tabulations for different values of gradtype and crsecred
nls_crs %>% group_by(gradtype) %>% count(crsecred) # cross tab of vars gradtype & crsecred
nls_crs %>% group_by(gradtype) %>% count(crsecred) %>% as_factor() #cross tab this time show value labe
nls_crs %>% group_by(gradtype) %>%
  summarise_at(.vars = vars(crsecred),
               .funs = funs(min, max, .args = list(na.rm = TRUE))) %>%
  as_factor()
#> Warning: `funs()` was deprecated in dplyr 0.8.0.
#> i Please use a list of either functions or lambdas:
#>
#> # Simple named list: list(mean = mean, median = median)
#>
#> # Auto named with `tibble::lst()`: tibble::lst(mean, median)
#>
#> # Using lambdas list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
#> Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
#> generated.

nls_crs%>% filter(gradtype==1) %>% count(crsecred) # letter grade
nls_crs%>% filter(gradtype==2) %>% count(crsecred) # numeric grade
nls_crs%>% filter(gradtype==9) %>% count(crsecred) # missing

nls_crs %>% group_by(gradtype) %>% count(crsecred)
```

Part II: Write out plan

Write a plan for how you will create institution-level (i.e., transcript-level GPA variable)

This plan should include your general conceptual definition for how to calculate GPA.

- The general definition of GPA is quality points (course credit multiplied by numerical grade value) divided by total credits.

- The plan should describe how you will apply this general definition to actual variables in the NLS course-level data

- The plan should also describe how you plan to deal with idiosyncracies in the value of "input" variables (e.g., missing values, strange values) and your rationale for treating the variable values this way.

- Note: you will almost certainly update this plan as you make progress.

Some guidelines/hints for creating gpa variable

- You will have to create a new version of course credit called `crsecredv2` that is missing (`NA`) for values of `crsecred` that you think refer to missing
- You will have to create a new course grade variable, call it `numgrade` that has numeric grade for each course
  - the primary input variables for `numgrade` will be `crsgrada`,`crsgradb`, `gradtype`, and your new course credit variable `crsecredv2`
  - Use this key to assign numeric values to letter grades from `crsgrada` - A+=4; A=4; A-=3.7; B+=3.3; B=3; B-=2.7; C+=2.3; C=2; C-=1.7; D+=1.3; D=1; D-=.7; F=0; E=0; WF=0
  - Note: WF refers to "Withdrawal with a failing grade"
  - Note: other letter grades will have missing values for numeric grade - your variable `numgrade` should be missing for observations where `crsecredv2` equals `NA` - your variable `numgrade` should be missing if `gradtype` indicates that the grade is numeric (rather than letter) but the value of the numeric grade (`crsegradb`) is greater than `4`
- After you create the variable `numgrade` you should create a new course credit variable `crsecredv3` that is missing (`NA`) for observations where `numgrade` is missing

- Calculate institutional level quality points and total credit variables by summing across observations within id and transnum.

- Finally, divide the institutional level quality points by insitutional total credits to generate the institutional level GPA.

Your Plan here:

Part III: Clean data

Clean data: create new versions of variables that will be inputs to your GPA variable

Some requirements

- Prior to creating any new variable, conduct investigations of the input variable(s)

- After creating any new variable, conduct investigations of the value of the new variable and check the value of the new variable against values of the input variable(s)

- The investigations that we gave you above may be useful

```r
#Create measure of course credits attempted that replaces 999 and 999.999 with missing

nls_crs%>% count(crsecred)

nls_crs%>% filter(crsecred==999) # printing some observations for specific values of crsecred
nls_crs%>% filter(crsecred==1000) # printing some observations for specific values of crsecred
nls_crs%>% filter(crsecred>=999) %>% count(crsecred) # printing some observations for specific values o

nls_crs<- nls_crs%>%
  mutate(crsecredv2= ifelse(crsecred>=900, NA, crsecred))

#check that variables have been created correctly

  nls_crs%>% filter(crsecred==999) %>% select(crsecred,crsecredv2)
  nls_crs%>% filter(crsecred>999) %>% select(crsecred,crsecredv2)

  nls_crs%>% filter(crsecred>900) %>% count(crsecredv2) # one-way frequency table
  nls_crs%>% filter(crsecred>900) %>% group_by(crsecred) %>% count(crsecredv2) # two-way frequency tabl
```

```r
#investigate values of the numeric grade when gradtype==2 and course credit not missing

  typeof(nls_crs$crsgradb)
  class(nls_crs$crsgradb)

  options(tibble.print_min=300)
  nls_crs%>% filter(gradtype==2, (!is.na(crsecredv2))) %>% count(crsgradb)

  nls_crs<- nls_crs%>% mutate(crsgradbv2= ifelse(crsgradb<=4 & gradtype==2 & (!is.na(crsecredv2)), crsg
  #%>%   filter(gradtype==2)

  #check variables
    nls_crs%>% filter(is.na(crsecredv2)) %>% count(crsgradbv2) # course credit is missing
    nls_crs%>% filter(gradtype %in% c(1,9)) %>% count(crsgradbv2) # course credit is missing
    nls_crs%>% filter(crsgradb>4) %>% count(crsgradbv2) # course credit is missing

    nls_crs%>% filter(crsgradb<=4, gradtype==2, (!is.na(crsecredv2))) %>% count(crsgradbv2) # tabulatio
    nls_crs%>% filter(crsgradb<=4, gradtype==2, (!is.na(crsecredv2))) %>% group_by(crsgradb) %>% count(
    nls_crs%>% filter(crsgradb<=4, gradtype==2, (!is.na(crsecredv2))) %>% mutate(assert=crsgradb==crsgra

#Create "numgrade" variable that has numeric grade associated with each class
  #check inputs
  nls_crs%>% count(gradtype)
  nls_crs%>% count(gradtype) %>% haven::as_factor()

  nls_crs%>% filter(gradtype==1) %>% count(crsgrada)

  #options(tibble.print_min=200)


  nls_crs<- nls_crs%>%
    mutate(
      numgrade=case_when(
        crsgrada %in% c("A+","A") & gradtype==1 & (!is.na(crsecredv2)) ~ 4,
        crsgrada=="A-" & gradtype==1 & (!is.na(crsecredv2)) ~ 3.7,
        crsgrada=="B+" & gradtype==1 & (!is.na(crsecredv2)) ~ 3.3,
        crsgrada=="B" & gradtype==1 & (!is.na(crsecredv2)) ~ 3,
        crsgrada=="B-" & gradtype==1 & (!is.na(crsecredv2)) ~ 2.7,
        crsgrada=="C+" & gradtype==1 & (!is.na(crsecredv2)) ~ 2.3,
        crsgrada=="C" & gradtype==1 & (!is.na(crsecredv2)) ~ 2,
        crsgrada=="C-" & gradtype==1 & (!is.na(crsecredv2)) ~ 1.7,
        crsgrada=="D+" & gradtype==1 & (!is.na(crsecredv2)) ~ 1.3,
        crsgrada=="D" & gradtype==1 & (!is.na(crsecredv2)) ~ 1,
        crsgrada=="D-" & gradtype==1 & (!is.na(crsecredv2)) ~ 0.7,
        crsgrada %in% c("F","E","WF") & gradtype==1 & (!is.na(crsecredv2)) ~ 0,
        crsgradb<=4 & gradtype==2 & (!is.na(crsecredv2)) ~ crsgradb # use values of numeric var crsgrad
      )
    )


#check variable created correctly
  nls_crs%>% count(numgrade)
  nls_crs%>% filter(is.na(crsecredv2)) %>% count(numgrade) # missing when crsecredv2==NA
  nls_crs%>% filter(gradtype==9) %>% count(numgrade) # missing when grade-type is not "letter"

  nls_crs%>% filter(gradtype==1, (!is.na(crsecredv2))) %>% count(numgrade) # when grade-type=letter and


  nls_crs%>% filter(gradtype==1, (!is.na(crsecredv2))) %>% group_by(crsgrada) %>% count(numgrade)  #whe

#check against values of crsgradb
  nls_crs%>% filter(crsgradb>4, gradtype==2, !is.na(crsecredv2)) %>% group_by(crsgradb) %>% count(numgra

#Create "quality points" variable, which equals credits attempted multiplied by numgrade

  nls_crs<- nls_crs%>% mutate(qualpts=numgrade*crsecredv2)

#checks
  nls_crs %>%
    count(qualpts)

  nls_crs %>%
    select(id, transnum, numgrade, crsecredv2, qualpts)

  nls_crs%>% filter(is.na(numgrade)) %>% count(qualpts) # missing when numgrade==NA

  nls_crs%>% group_by(numgrade, crsecredv2) %>% count(qualpts) #group by input variables and get a coun

  nls_crs%>% filter(numgrade==0 & qualpts!=0) %>% select(numgrade,crsecredv2,qualpts) #If variable was

  nls_crs%>% filter(crsecredv2==0 & qualpts!=0) %>% select(numgrade,crsecredv2,qualpts) #same logic



#Create measure of credits attempted that is missing if numgrade is missing
nls_crs <- nls_crs%>%
  mutate(
    crsecredv3=ifelse(is.na(numgrade),NA,crsecredv2))

#check
nls_crs %>%
  count(crsecredv3)
```

Part IV: Create institution-level GPA variable

Create institution-level GPA variable and save as a new object

```
    nls_crs_trans <- nls_crs%>% group_by(id,transnum) %>%
      summarise(
        cred_trans=sum(crsecredv3, na.rm=TRUE), # sum total credits attempted where grade is known
        qualpts_trans=sum(qualpts, na.rm=TRUE) # sum of quality points where grade is known
      ) %>%
      mutate(gpa_trans=qualpts_trans/cred_trans)
#> `summarise()` has grouped output by 'id'. You can override using the `.groups`
#> argument.
```

```
    nls_crs_trans

    institutional_gpa_df <- nls_crs %>% group_by(id, transnum) %>%
summarise(cred_trans=sum(crsecredv3, na.rm=TRUE),
          qualpts_trans=sum(qualpts, na.rm=TRUE))  %>% mutate(gpa_trans=qualpts_trans/cred_trans)
#> `summarise()` has grouped output by 'id'. You can override using the `.groups`
#> argument.


    #options(tibble.print_min=400)
```

After you create the gpa variable, conduct some basic investigations/descriptive statistics to check whether it looks reasonable

```
nls_crs_trans %>%
  count(gpa_trans) #freq count of new variable

nls_crs_trans %>%
  filter(is.na(gpa_trans)) #view NA for new variable

nls_crs_trans %>%
  filter(cred_trans==0 & qualpts_trans!=0) #checking to see if cred_trans equals 0 and qualpts_trans eq

nls_crs_trans %>%
  filter(cred_trans==0 & !is.na(gpa_trans)) #checking to see if cred_trans equals 0 and gpa_trans does

nls_crs_trans %>%
  filter((qualpts_trans/cred_trans) != gpa_trans)

nls_crs_trans %>%
  count(gpa_trans) %>%
  filter(n > 1)
```

Part V: Create term-level GPA variable

Create term-level GPA variable and save as a new object

```
    nls_crs_term <- nls_crs%>% group_by(id,transnum,termnum) %>%
      summarise(
        cred_term=sum(crsecredv3, na.rm=TRUE), # sum total credits attempted where grade is known
        qualpts_term=sum(qualpts, na.rm=TRUE) # sum total credits attempted where grade is known
      ) %>%
      mutate(gpa_term=qualpts_term/cred_term)
#> `summarise()` has grouped output by 'id', 'transnum'. You can override using
#> the `.groups` argument.

    nls_crs_term
```

After you create the gpa variable, conduct some basic investigations/descriptive statistics to check whether it looks reasonable

```
nls_crs_term %>%
  count(gpa_term) #freq count of new variable
```

```
nls_crs_term %>%
  filter(is.na(gpa_term)) #view NA for new variable

nls_crs_term %>%
  filter(cred_term==0 & qualpts_term!=0) #checking to see if cred_trans equals 0 and qualpts_trans equa

nls_crs_term %>%
  filter(cred_term==0 & !is.na(gpa_term)) #checking to see if cred_trans equals 0 and gpa_trans does no

nls_crs_term %>%
  filter((qualpts_term/cred_term) != gpa_term)

nls_crs_term %>%
  count(gpa_term) %>%
  filter(n > 1)
```

Once finished, knit to (pdf) and upload both .Rmd and pdf files to class website under the week 5 tab
Remember to use this naming convention "lastname_firstname_ps5"