Lecture 5 problem set

INSERT YOUR NAME HERE

November 2, 2018

Contents

Grade: /20

Required reading and instructions

Required reading before next class

- Work through slides from lecture 5 that we don't get to in class
- GW 15.1 - 15.2 (factors) [this is like 2-3 pages]
- [OPTIONAL] GW 15.3 - 15.5 (remainder of "factors" chapter)
- [OPTIONAL] GW 20.6 - 20.7 (attributes and augmented vectors)
- [OPTIONAL] GW 10 (tibbles)

General Problem Set instructions

In this homework, you will specify `pdf_document` as the output format. You must have LaTeX installed in order to create pdf documents.

If you have not yet installed MiKTeX/MacTeX, I recommend installing TinyTeX, which is much simpler to install!

- Instructions for installation of TinyTeX can be found HERE
- General Instructions for Problem Sets Here

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

General Instructions

- Don't make changes to "input" variables; instead, create a new variable

- You are responsible for deciding what data investigations to conduct (e.g., conditional statements, frequency counts, etc.)
  - Keep the data investigations you want Patricia to see; though you might want to comment out very long lists of observations
  - 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; drop these variables when you no longer need them
  - Below, you will find additional instructions/hints about making GPA variable

Load library and data

```
#install.packages("tidyverse") #uncomment if you haven't installed these packaged
#install.packages("haven")
#install.packages("labelled")
library(tidyverse)
#> -- Attaching packages ------------------------------------------------ tidyverse 1.2.1 --
#> v ggplot2 3.1.0       v purrr   0.2.5
#> v tibble  2.1.1       v dplyr   0.8.0.1
#> v tidyr   0.8.3       v stringr 1.4.0
#> v readr   1.3.1       v forcats 0.3.0
#> -- Conflicts --------------------------------------------------- tidyverse_conflicts() --
#> x dplyr::filter() masks stats::filter()
#> x dplyr::lag()    masks stats::lag()
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
```

Sort, select, and order

Here we sort the observations, remove a variable, and arrange order of variables. We also create a new object `nls_crs_v2`, which is the data frame object you will work with for the rest of the problem set. All you have to do is run this code chunk.

```
names(nls_crs)
#>  [1] "crsecip"  "gradtype" "id"       "transnum" "termnum"  "crsecred"
#>  [7] "crsgradb" "crsename" "crsgrada" "cname"

# sort observations and assign to new object nls_crs_v2, which we'll use for rest of problem set
nls_crs_v2 <- nls_crs %>% arrange(id, transnum, termnum, crsename)

# select order variables; and drop var cname, which we don't use
nls_crs_v2 <- nls_crs_v2 %>% select(id, transnum, termnum, crsecred, gradtype,
  crsgrada, crsgradb, crsecip, crsename,-cname)

#rm(nls_crs)
names(nls_crs_v2)
#> [1] "id"       "transnum" "termnum"  "crsecred" "gradtype" "crsgrada"
#> [7] "crsgradb" "crsecip"  "crsename"
```

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. Since you have never seen this dataset before, we conduct the investigation for you. all you have to do is run the code in code chunks, but spend some time thinking about how the code works and why these investigations.

- Preliminary investigation of data frame

```
names(nls_crs_v2)
glimpse(nls_crs_v2)
str(nls_crs_v2)
head(nls_crs_v2)
nls_crs_v2 %>% var_label() # view variable labels
```

- Perform one-way investigations following input variables:
  - `transnum`, `termnum`, `crsecred`, `gradtype`, `crsgrada`, `crsgradb`
- First four variables done for you. **Run one-way investigations for `crsgrada` and `crsgradb`**

/2

```
#Investigate variable transnum
class(nls_crs_v2$transnum)
nls_crs_v2%>% select(transnum) %>% var_label() # view variable labels
nls_crs_v2%>% count(transnum)


#Check that sum of transnum equals number of rows in dataset
nls_crs_v2 %>%
  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_v2$termnum)
nls_crs_v2%>% select(termnum) %>% var_label() # view variable labels
nls_crs_v2%>% count(termnum)


#Investigate course credits
#glimpse(nls_crs_v2)
class(nls_crs_v2$crsecred)
nls_crs_v2%>% select(crsecred) %>% var_label() # view variable labels
options(tibble.print_min=50)
nls_crs_v2%>% count(crsecred)
nls_crs_v2 %>% #run some descriptive stats
  summarise_at(
    .vars = vars(crsecred),
    .funs = funs(min, max, .args=list(na.rm=TRUE))
)
#> Warning: funs() is soft deprecated as of dplyr 0.8.0
#> please use list() instead
#>
#> # Before:
#> funs(name = f(.)
#>
#> # After:
#> list(name = ~f(.))
#> This warning is displayed once per session.
#investigate high values of crsecred
nls_crs_v2%>% filter(crsecred>=100) %>% count(crsecred) # frequency table of crsecred
nls_crs_v2%>% filter(crsecred==999) # printing some observations for specific values of crsecred
nls_crs_v2%>% filter(crsecred>=999) %>% count(crsecred) #


#Investigate gradtype
class(nls_crs_v2$gradtype) # labelled
#glimpse(nls_crs_v2)
nls_crs_v2%>% select(gradtype) %>% var_label() # view variable labels
nls_crs_v2%>% select(gradtype) %>% val_labels() # view value labels on variable
nls_crs_v2 %>% count(gradtype) #freq count of values
nls_crs_v2 %>% count(gradtype) %>% as_factor() #freq count with value labels


#Run one-way investigation for crsgrada
#crsgrada
#glimpse(nls_crs_v2)
class(nls_crs_v2$crsgrada) #character
nls_crs_v2%>% select(crsgrada) %>% var_label() # view variable labels
nls_crs_v2%>% count(crsgrada)
nls_crs_v2%>% filter(crsgrada %in% c("99", "AU", "CR", "I", "NO", "P", "S", "U", "W", "WP")) #printing

#Run one-way investigation for crsgradb
```

```
#crsgradb
class(nls_crs_v2$crsgradb) #numeric
nls_crs_v2%>% select(crsgradb) %>% var_label()
nls_crs_v2%>% count(crsgradb)
nls_crs_v2 %>% #run some descriptive stats
  summarise_at(
    .vars = vars(crsgradb),
    .funs = funs(min, max, .args=list(na.rm=TRUE))
)
nls_crs_v2%>% filter(crsgradb>100) %>% count(crsgradb)
```

- Investigate the relationship between the following pairs of variables:
  - gradtype and crsgrada
  - gradtype and crsgradb
  - crsecred and gradtype
- First two are done for you. **Investigate relationship between crsecred and gradtype**

/2

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

#some tabulations for different values of gradtype and crsgrada
nls_crs_v2 %>% group_by(gradtype) %>% count(crsgrada) # cross tab of vars gradtype & crsgrada
nls_crs_v2 %>% group_by(gradtype) %>% count(crsgrada) %>% as_factor() #cross tab this time show value l

nls_crs_v2%>% filter(gradtype==1) %>% count(crsgrada) # letter grade
nls_crs_v2%>% filter(gradtype==2) %>% count(crsgrada) # numeric grade
nls_crs_v2%>% filter(gradtype==9) %>% count(crsgrada) # missing


#some tabulations for different values of gradtype and crsgradb
nls_crs_v2 %>% group_by(gradtype) %>% count(crsgradb) # cross tab of vars gradtype & crsgradb
nls_crs_v2 %>% group_by(gradtype) %>% count(crsgradb) %>% as_factor() #cross tab this time show value l
nls_crs_v2 %>% group_by(gradtype) %>%
  summarise_at(.vars = vars(crsgradb),
               .funs = funs(min, max, .args = list(na.rm = TRUE))) %>%
  as_factor()

nls_crs_v2%>% filter(gradtype==1) %>% count(crsgradb) # letter grade
nls_crs_v2%>% filter(gradtype==2) %>% count(crsgradb) # numeric grade
nls_crs_v2%>% 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_v2 %>% group_by(gradtype) %>% count(crsecred) # cross tab of vars gradtype & crsecred
nls_crs_v2 %>% group_by(gradtype) %>% count(crsecred) %>% as_factor() #cross tab this time show value l
nls_crs_v2 %>% group_by(gradtype) %>%
  summarise_at(.vars = vars(crsecred),
               .funs = funs(min, max, .args = list(na.rm = TRUE))) %>%
  as_factor()

nls_crs_v2%>% filter(gradtype==1) %>% count(crsecred) # letter grade
```

```
nls_crs_v2%>% filter(gradtype==2) %>% count(crsecred) # numeric grade
nls_crs_v2%>% filter(gradtype==9) %>% count(crsecred) # missing

nls_crs_v2 %>% 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:
/3

Part III: Clean data

/5

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

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

nls_crs_v2%>% count(crsecred)

nls_crs_v2%>% filter(crsecred==999) # printing some observations for specific values of crsecred
nls_crs_v2%>% filter(crsecred==1000) # printing some observations for specific values of crsecred
nls_crs_v2%>% filter(crsecred>=999) %>% count(crsecred) # printing some observations for specific value

nls_crs_v2<- nls_crs_v2%>%
  mutate(crsecredv2= ifelse(crsecred>=900, NA, crsecred))

#check that variables have been created correctly

  nls_crs_v2%>% filter(crsecred==999) %>% select(crsecred,crsecredv2)
  nls_crs_v2%>% filter(crsecred>999) %>% select(crsecred,crsecredv2)

  nls_crs_v2%>% filter(crsecred>900) %>% count(crsecredv2) # one-way frequency table
  nls_crs_v2%>% filter(crsecred>900) %>% group_by(crsecred) %>% count(crsecredv2) # two-way frequency t


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

  typeof(nls_crs_v2$crsgradb)
  class(nls_crs_v2$crsgradb)

  options(tibble.print_min=300)
  nls_crs_v2%>% filter(gradtype==2, (!is.na(crsecredv2))) %>% count(crsgradb)

  nls_crs_v2<- nls_crs_v2%>% mutate(crsgradbv2= ifelse(crsgradb<=4 & gradtype==2 & (!is.na(crsecredv2))
  #%>%   filter(gradtype==2)

  #check variables
    nls_crs_v2%>% filter(is.na(crsecredv2)) %>% count(crsgradbv2) # course credit is missing
    nls_crs_v2%>% filter(gradtype %in% c(1,9)) %>% count(crsgradbv2) # course credit is missing
    nls_crs_v2%>% filter(crsgradb>4) %>% count(crsgradbv2) # course credit is missing

    nls_crs_v2%>% filter(crsgradb<=4, gradtype==2, (!is.na(crsecredv2))) %>% count(crsgradbv2) # tabula
    nls_crs_v2%>% filter(crsgradb<=4, gradtype==2, (!is.na(crsecredv2))) %>% group_by(crsgradb) %>% cou
    nls_crs_v2%>% filter(crsgradb<=4, gradtype==2, (!is.na(crsecredv2))) %>% mutate(assert=crsgradb==cr

#Create "numgrade" variable that has numeric grade associated with each class
  #check inputs
  nls_crs_v2%>% count(gradtype)
  nls_crs_v2%>% count(gradtype) %>% haven::as_factor()
```

```r
  nls_crs_v2%>% filter(gradtype==1) %>% count(crsgrada)

  #options(tibble.print_min=200)


  nls_crs_v2<- nls_crs_v2%>%
    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_v2%>% count(numgrade)
  nls_crs_v2%>% filter(is.na(crsecredv2)) %>% count(numgrade) # missing when crsecredv2==NA
  nls_crs_v2%>% filter(gradtype==9) %>% count(numgrade) # missing when grade-type is not "letter"

  nls_crs_v2%>% filter(gradtype==1, (!is.na(crsecredv2))) %>% count(numgrade) # when grade-type=letter


  nls_crs_v2%>% filter(gradtype==1, (!is.na(crsecredv2))) %>% group_by(crsgrada) %>% count(numgrade)  #

#check against values of crsgradb
  nls_crs_v2%>% filter(crsgradb>4, gradtype==2, !is.na(crsecredv2)) %>% group_by(crsgradb) %>% count(nu

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

  nls_crs_v2<- nls_crs_v2%>% mutate(qualpts=numgrade*crsecredv2)

#checks
  nls_crs_v2 %>%
    count(qualpts)

  nls_crs_v2 %>%
    select(id, transnum, numgrade, crsecredv2, qualpts)

  nls_crs_v2%>% filter(is.na(numgrade)) %>% count(qualpts) # missing when numgrade==NA

  nls_crs_v2%>% group_by(numgrade, crsecredv2) %>% count(qualpts) #group by input variables and get a c
```

```r
  nls_crs_v2%>% filter(numgrade==0 & qualpts!=0) %>% select(numgrade,crsecredv2,qualpts) #If variable w

  nls_crs_v2%>% 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_v2<- nls_crs_v2%>%
  mutate(
    crsecredv3=ifelse(is.na(numgrade),NA,crsecredv2))

#check
nls_crs_v2 %>%
  count(crsecredv3)
```

Part IV: Create institution-level GPA variable

/3 Create institution-level GPA variable and save as a new object

```r
    nls_crs_trans <- nls_crs_v2%>% 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)

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

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

/1

```r
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

/3

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

```
    nls_crs_term <- nls_crs_v2%>% 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)

    nls_crs_term
```

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

/1

```
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"