Lab 9 - Data Transformation

Snow Christensen

October 29, 2017

1. In addition to simply naming variable names in select you can also use `:` to select a range of variables and `-` to exclude some variables, similar to indexing a `data.frame` with square brackets. You can use both variable's names as well as integer indexes.

a. Use `select()` to print out a tbl that contains only the first 3 columns of your dataset, called by name.
b. Print out a `tbl` with the last 3 columns of your dataset, called by name.
c. Find the most concise way to select the first 3 columns and the last 3 columns by name.

```
#load package for reading .por file
# install.packages("haven")
library("haven")

# Read in your data with the appropriate function
data <- read_por("~/Projects/sexual_violence_college/ICPSR_03212/DS0001/03212-0001-Data.por")

#load dplyr
library("dplyr")
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
#load magrittr
library("magrittr")

#create tbl with only first three columns using select

 data_beg <- data %>%
  select(CODENUM, EDUC, MOB)

# print data_beg
 data_beg
```

```
## # A tibble: 1,580 x 3
##    CODENUM      EDUC       MOB
##      <chr> <dbl+lbl> <dbl+lbl>
##  1 90-0002        12         3
##  2 90-0003        12         6
##  3 90-0004        12         6
##  4 90-0005        12        10
##  5 90-0006        12         6
##  6 90-0007        12         1
##  7 90-0011        12         1
```

```
##  8 90-0014        12         4
##  9 90-0015        12         8
## 10 90-0016        12         1
## # ... with 1,570 more rows
```

```
#create tbl with only last three columns using select
data_end <- data %>%
  select(COOCCURA, LATERST, OVERALL)

#print data_end
data_end
```

```
## # A tibble: 1,580 x 3
##     COOCCURA   LATERST   OVERALL
##    <dbl+lbl> <dbl+lbl> <dbl+lbl>
##  1         2        NA        NA
##  2         4        NA        NA
##  3         1         1         1
##  4         2         2         3
##  5         1         1         2
##  6         3         2         4
##  7         1        NA        NA
##  8         4        NA        NA
##  9         2         1         1
## 10         3         2         3
## # ... with 1,570 more rows
```

2. `dplyr` comes with a set of helper functions that can help you select groups of variables inside a `select()` call:

- `starts_with("X")`: every name that starts with "X",
- `ends_with("X")`: every name that ends with "X",
- `contains("X")`: every name that contains "X",
- `matches("X")`: every name that matches "X", where "X" can be a regular expression,
- `num_range("x", 1:5)`: the variables named x01, x02, x03, x04 and x05,
- `one_of(x)`: every name that appears in x, which should be a character vector.

Pay attention here: When you refer to columns directly inside `select()`, you don't use quotes. If you use the helper functions, you do use quotes.

a. Use `select()` and a helper function to print out a `tbl` that selects only variables that contain a specific character string.
b. Use `select()` and a helper function to print out a `tbl` that selects only variables that start with a certain letter or string of letters.

```
#using select() and starts_with() to print out a tbl with specific character string
select(data, starts_with("AUTH"))
```

```
## # A tibble: 1,580 x 10
##       AUTHSP    AUTHSI   AUTHSP2   AUTHSI2   AUTHSP3   AUTHSI3   AUTHSP4
##    <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl>
##  1         1         1         1         1         1         1         1
##  2         1         1         1         1        NA        NA        NA
##  3         1         1         1         1         1         1         1
##  4         1         1         1         1         1         1         1
##  5         1         1         2         1         1         1         1
##  6         1         1         1         1         1         1        NA
##  7         1         1         1         1        NA        NA        NA
```

```
##  8         1         1         1         1        NA        NA        NA
##  9         1         1         1         1         1         1         1
## 10         1         1         1         1         1         1         1
## # ... with 1,570 more rows, and 3 more variables: AUTHSI4 <dbl+lbl>,
## #   AUTHSP5 <dbl+lbl>, AUTHSI5 <dbl+lbl>
```

```
#using select() and containt() to print out tbl with specific letter
select(data, contains("A", "T"))
```

```
## # A tibble: 1,580 x 859
##        DAYOB      RACE   MARSTAT    RELATT   RELATT1      GRAD    CHARGE
##    <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl>
##  1        NA         1         1         5         1         1         3
##  2        NA         2         1         2         4         1         4
##  3        NA         3         1         4         2         1         3
##  4        NA         1         1         3         3         1         3
##  5        NA         1         1         2         4         1         2
##  6        NA         3         1         1         5         1         2
##  7        NA         1         1         4         3         1         2
##  8        NA         1         1         1         5         1         5
##  9        NA         1         1         2         4         2         2
## 10        NA         1         1         2         4         1         3
## # ... with 1,570 more rows, and 852 more variables: GREATEST <dbl+lbl>,
## #   BRAG <dbl+lbl>, GETMAD <dbl+lbl>, ALTRUE <dbl+lbl>,
## #   COMPLAIN <dbl+lbl>, NOASSERT <dbl+lbl>, MEAN <dbl+lbl>,
## #   APPROV <dbl+lbl>, NAG <dbl+lbl>, NOEMPATH <dbl+lbl>, EMPATH <dbl+lbl>,
## #   PLEASE <dbl+lbl>, BADNERV <dbl+lbl>, ANXIOUS <dbl+lbl>,
## #   NOCALM <dbl+lbl>, RATTLED <dbl+lbl>, SHAKEH <dbl+lbl>,
## #   RELAX <dbl+lbl>, STRAIN <dbl+lbl>, STABLE <dbl+lbl>, DEAD <dbl+lbl>,
## #   HAPPY <dbl+lbl>, SATISFID <dbl+lbl>, CALM <dbl+lbl>,
## #   ADVENTUR <dbl+lbl>, XPECTDAY <dbl+lbl>, WAKEUP <dbl+lbl>,
## #   LUVRELAT <dbl+lbl>, IARGUED <dbl+lbl>, HARGUED <dbl+lbl>,
## #   ITHREAT <dbl+lbl>, HTHREAT <dbl+lbl>, ITHRUAT <dbl+lbl>,
## #   HTHRUAT <dbl+lbl>, IHITAT <dbl+lbl>, HHITAT <dbl+lbl>,
## #   DATBEHHS <dbl+lbl>, DATFRQHS <dbl+lbl>, NUMDATES <dbl+lbl>,
## #   INJDATE <dbl+lbl>, SEXAPROV <dbl+lbl>, AUTHSP <dbl+lbl>,
## #   ATTEMPT <dbl+lbl>, DRUGATT <dbl+lbl>, AUTHSI <dbl+lbl>,
## #   SEXACTS <dbl+lbl>, KOSS91A1 <dbl+lbl>, WHEN90A1 <dbl+lbl>,
## #   INITIATE <dbl+lbl>, PAID <dbl+lbl>, CONTACT <dbl+lbl>,
## #   AGEEXP <dbl+lbl>, AGEWHO <dbl+lbl>, UNWANTSX <dbl+lbl>,
## #   RAPED <dbl+lbl>, FAMHIT <dbl+lbl>, SHOORGAN <dbl+lbl>,
## #   ATTEMPSI <dbl+lbl>, WHOSTART <dbl+lbl>, REASON <dbl+lbl>,
## #   CHIVALRY <dbl+lbl>, PASSROLE <dbl+lbl>, CHIVDMEA <dbl+lbl>,
## #   ASKFORIT <dbl+lbl>, MENASK <dbl+lbl>, HSTARTSX <dbl+lbl>,
## #   ARROGANT <dbl+lbl>, HERFAULT <dbl+lbl>, NOAGGRES <dbl+lbl>,
## #   PROGRAM <dbl+lbl>, YRADMIN <chr>, XAUTHSP <dbl+lbl>,
## #   XATTEMPT <dbl+lbl>, XDRUGATT <dbl+lbl>, XAUTHSI <dbl+lbl>,
## #   XSEXACTS <dbl+lbl>, ANXIETY <dbl+lbl>, POSAFF <dbl+lbl>,
## #   FVA <dbl+lbl>, TRADATT <dbl+lbl>, CHIVLATT <dbl+lbl>,
## #   MALEVIOL <dbl+lbl>, DISAPRN <dbl+lbl>, PRAT <dbl+lbl>, VRAT <dbl+lbl>,
## #   VAGG <dbl+lbl>, VVA <dbl+lbl>, PAGG <dbl+lbl>, VPA <dbl+lbl>,
## #   MARSTAT2 <dbl+lbl>, RELSTAT2 <dbl+lbl>, RELATT2 <dbl+lbl>,
## #   CHARGE2 <dbl+lbl>, GREAT2 <dbl+lbl>, BRAG2 <dbl+lbl>,
## #   GETMAD2 <dbl+lbl>, ALTRUE2 <dbl+lbl>, COMPLAI2 <dbl+lbl>,
## #   NOASSER2 <dbl+lbl>, MEAN2 <dbl+lbl>, ...
```

4. Are there any mutations you wish to carry out on your data (i.e. new variables you wish to create based upon the values of already existing variables)? If so, describe what they are and what you will name them.

No, I do not need to carry out any mutations.

5. You can use mutate() to add multiple variables at once. To create more than one variable, place a comma between each variable that you define inside `mutate()`.

a. Carry out any and all of the mutations you wish to perform on your dataset and print the results to the console.

I mutated the data in question 9 to be numeric so I could read the data into the summarise() function.

6. `R` comes with a set of logical operators that you can use inside `filter()`:

- `x < y`, `TRUE` if `x` is less than `y`
- `x <= y`, `TRUE` if `x` is less than or equal to `y`
- `x == y`, `TRUE` if `x` equals `y`
- `x != y`, `TRUE` if `x` does not equal `y`
- `x >= y`, `TRUE` if `x` is greater than or equal to `y`
- `x > y`, `TRUE` if `x` is greater than `y`
- `x %in% c(a, b, c)`, `TRUE` if `x` is in the vector `c(a, b, c)`

a. What are some potential subsets of your data that seem interesting and worth investigation to you?
b. Use at least two of the logical operators presented above to print these subsets of your data.

It would be interesting to see the relationship between the variables SEXACTS and PLEASE because SEXACTS is data about sexual acts because of force and PLEASE is about the resondents willingness or need to please others generally. Additionally the relationship between AUTHSI and DRUGSI would be interesting because then we could see if which factor, drugs or authority, leads to more sexual violence.

```
#only print rows where AUTHSI is greater than or equal to one and DRUGSI is less than or equal to one
filter(data, AUTHSI >= 1 & DRUGSI <= 1)
```

```
## # A tibble: 1,460 x 2,075
##    CODENUM      EDUC       MOB     DAYOB      B_YR      RACE   MARSTAT
##      <chr> <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl>
##  1 90-0002        12         3        NA      1972         1         1
##  2 90-0003        12         6        NA      1972         2         1
##  3 90-0004        12         6        NA      1973         3         1
##  4 90-0005        12        10        NA      1972         1         1
##  5 90-0006        12         6        NA      1972         1         1
##  6 90-0007        12         1        NA      1972         3         1
##  7 90-0011        12         1        NA      1972         1         1
##  8 90-0014        12         4        NA      1972         1         1
##  9 90-0015        12         8        NA      1970         1         1
## 10 90-0016        12         1        NA      1972         1         1
## # ... with 1,450 more rows, and 2068 more variables: RELINFL <dbl+lbl>,
## #   RELATT <dbl+lbl>, RELIG <dbl+lbl>, RELINFL1 <dbl+lbl>,
## #   RELATT1 <dbl+lbl>, GRAD <dbl+lbl>, CHARGE <dbl+lbl>, WHINY <dbl+lbl>,
## #   TOUGH <dbl+lbl>, GREATEST <dbl+lbl>, EMOTE <dbl+lbl>,
## #   GIVEIN <dbl+lbl>, BRAG <dbl+lbl>, GETMAD <dbl+lbl>, BUSY <dbl+lbl>,
## #   CENTER <dbl+lbl>, ALTRUE <dbl+lbl>, WHIMPY <dbl+lbl>, ROUGH <dbl+lbl>,
## #   COMPLAIN <dbl+lbl>, HELPFUL <dbl+lbl>, CONTESTS <dbl+lbl>,
## #   NOASSERT <dbl+lbl>, HOMEY <dbl+lbl>, GREED <dbl+lbl>, MEAN <dbl+lbl>,
## #   APPROV <dbl+lbl>, BOSSY <dbl+lbl>, NOHURT <dbl+lbl>, NAG <dbl+lbl>,
## #   NOEMPATH <dbl+lbl>, INDECIS <dbl+lbl>, FUSSY <dbl+lbl>,
## #   GIVEUP <dbl+lbl>, NOTRUST <dbl+lbl>, NOCRY <dbl+lbl>,
```

```
## #   CONFIDEN <dbl+lbl>, NUMONE <dbl+lbl>, BETTER <dbl+lbl>,
## #   REVENGE <dbl+lbl>, EMPATH <dbl+lbl>, FRIENDLY <dbl+lbl>,
## #   PLEASE <dbl+lbl>, NORISK <dbl+lbl>, TRUSTFUL <dbl+lbl>,
## #   FLUSTER <dbl+lbl>, NERVOUS <dbl+lbl>, BADNERV <dbl+lbl>,
## #   TENSE <dbl+lbl>, ANXIOUS <dbl+lbl>, NOCALM <dbl+lbl>, JUMPY <dbl+lbl>,
## #   RESTLESS <dbl+lbl>, RATTLED <dbl+lbl>, SHAKEH <dbl+lbl>,
## #   RELAX <dbl+lbl>, MOODY <dbl+lbl>, LOSPIRIT <dbl+lbl>, BLUE <dbl+lbl>,
## #   DEPRESS <dbl+lbl>, STRAIN <dbl+lbl>, CONTROL <dbl+lbl>,
## #   LOSEMIND <dbl+lbl>, STABLE <dbl+lbl>, NOSUCCES <dbl+lbl>,
## #   CRYING <dbl+lbl>, DEAD <dbl+lbl>, DUMPS <dbl+lbl>, SUICIDE <dbl+lbl>,
## #   NOFORWRD <dbl+lbl>, HAPPY <dbl+lbl>, SATISFID <dbl+lbl>,
## #   INTEREST <dbl+lbl>, CALM <dbl+lbl>, CHEERFUL <dbl+lbl>,
## #   ENJOY <dbl+lbl>, NOTENSE <dbl+lbl>, ADVENTUR <dbl+lbl>,
## #   XPECTDAY <dbl+lbl>, WAKEUP <dbl+lbl>, FUTRHOPE <dbl+lbl>,
## #   LOVED <dbl+lbl>, LUVRELAT <dbl+lbl>, LONELY <dbl+lbl>,
## #   IDISCUSS <dbl+lbl>, HDISCUSS <dbl+lbl>, IDISCUSD <dbl+lbl>,
## #   HDISCUSD <dbl+lbl>, IGOTINFO <dbl+lbl>, HGOTINFO <dbl+lbl>,
## #   IGOTHELP <dbl+lbl>, HGOTHELP <dbl+lbl>, IARGUED <dbl+lbl>,
## #   HARGUED <dbl+lbl>, IYELLED <dbl+lbl>, HYELLED <dbl+lbl>,
## #   ISULKED <dbl+lbl>, HSULKED <dbl+lbl>, ISTOMPED <dbl+lbl>,
## #   HSTOMPED <dbl+lbl>, ...
```

```r
#only print rows where PLEASE is greater than or equal to 1 and SEXACTS is greater than or equal to one
filter(data, PLEASE >= 1 & SEXACTS >= 1)
```

```
## # A tibble: 1,566 x 2,075
##    CODENUM      EDUC       MOB     DAYOB      B_YR      RACE   MARSTAT
##      <chr> <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl>
##  1 90-0002        12         3        NA      1972         1         1
##  2 90-0003        12         6        NA      1972         2         1
##  3 90-0004        12         6        NA      1973         3         1
##  4 90-0005        12        10        NA      1972         1         1
##  5 90-0006        12         6        NA      1972         1         1
##  6 90-0007        12         1        NA      1972         3         1
##  7 90-0011        12         1        NA      1972         1         1
##  8 90-0014        12         4        NA      1972         1         1
##  9 90-0015        12         8        NA      1970         1         1
## 10 90-0016        12         1        NA      1972         1         1
## # ... with 1,556 more rows, and 2068 more variables: RELINFL <dbl+lbl>,
## #   RELATT <dbl+lbl>, RELIG <dbl+lbl>, RELINFL1 <dbl+lbl>,
## #   RELATT1 <dbl+lbl>, GRAD <dbl+lbl>, CHARGE <dbl+lbl>, WHINY <dbl+lbl>,
## #   TOUGH <dbl+lbl>, GREATEST <dbl+lbl>, EMOTE <dbl+lbl>,
## #   GIVEIN <dbl+lbl>, BRAG <dbl+lbl>, GETMAD <dbl+lbl>, BUSY <dbl+lbl>,
## #   CENTER <dbl+lbl>, ALTRUE <dbl+lbl>, WHIMPY <dbl+lbl>, ROUGH <dbl+lbl>,
## #   COMPLAIN <dbl+lbl>, HELPFUL <dbl+lbl>, CONTESTS <dbl+lbl>,
## #   NOASSERT <dbl+lbl>, HOMEY <dbl+lbl>, GREED <dbl+lbl>, MEAN <dbl+lbl>,
## #   APPROV <dbl+lbl>, BOSSY <dbl+lbl>, NOHURT <dbl+lbl>, NAG <dbl+lbl>,
## #   NOEMPATH <dbl+lbl>, INDECIS <dbl+lbl>, FUSSY <dbl+lbl>,
## #   GIVEUP <dbl+lbl>, NOTRUST <dbl+lbl>, NOCRY <dbl+lbl>,
## #   CONFIDEN <dbl+lbl>, NUMONE <dbl+lbl>, BETTER <dbl+lbl>,
## #   REVENGE <dbl+lbl>, EMPATH <dbl+lbl>, FRIENDLY <dbl+lbl>,
## #   PLEASE <dbl+lbl>, NORISK <dbl+lbl>, TRUSTFUL <dbl+lbl>,
## #   FLUSTER <dbl+lbl>, NERVOUS <dbl+lbl>, BADNERV <dbl+lbl>,
## #   TENSE <dbl+lbl>, ANXIOUS <dbl+lbl>, NOCALM <dbl+lbl>, JUMPY <dbl+lbl>,
## #   RESTLESS <dbl+lbl>, RATTLED <dbl+lbl>, SHAKEH <dbl+lbl>,
```

```
## #   RELAX <dbl+lbl>, MOODY <dbl+lbl>, LOSPIRIT <dbl+lbl>, BLUE <dbl+lbl>,
## #   DEPRESS <dbl+lbl>, STRAIN <dbl+lbl>, CONTROL <dbl+lbl>,
## #   LOSEMIND <dbl+lbl>, STABLE <dbl+lbl>, NOSUCCES <dbl+lbl>,
## #   CRYING <dbl+lbl>, DEAD <dbl+lbl>, DUMPS <dbl+lbl>, SUICIDE <dbl+lbl>,
## #   NOFORWRD <dbl+lbl>, HAPPY <dbl+lbl>, SATISFID <dbl+lbl>,
## #   INTEREST <dbl+lbl>, CALM <dbl+lbl>, CHEERFUL <dbl+lbl>,
## #   ENJOY <dbl+lbl>, NOTENSE <dbl+lbl>, ADVENTUR <dbl+lbl>,
## #   XPECTDAY <dbl+lbl>, WAKEUP <dbl+lbl>, FUTRHOPE <dbl+lbl>,
## #   LOVED <dbl+lbl>, LUVRELAT <dbl+lbl>, LONELY <dbl+lbl>,
## #   IDISCUSS <dbl+lbl>, HDISCUSS <dbl+lbl>, IDISCUSD <dbl+lbl>,
## #   HDISCUSD <dbl+lbl>, IGOTINFO <dbl+lbl>, HGOTINFO <dbl+lbl>,
## #   IGOTHELP <dbl+lbl>, HGOTHELP <dbl+lbl>, IARGUED <dbl+lbl>,
## #   HARGUED <dbl+lbl>, IYELLED <dbl+lbl>, HYELLED <dbl+lbl>,
## #   ISULKED <dbl+lbl>, HSULKED <dbl+lbl>, ISTOMPED <dbl+lbl>,
## #   HSTOMPED <dbl+lbl>, ...
```

7. `R` also comes with a set of boolean operators that you can use to combine multiple logical tests into a single test. These include `&` (and), `|` (or), and `!` (not). Instead of using the `&` operator, you can also pass several logical tests to `filter()`, separated by commas. `is.na()` will also come in handy.

a. Use `R`'s logical and boolean operators to select just the rows in your data that meet a specific boolean condition.
b. Print out all of the observations in your data in which none of variables are `NA`.

```
#select rows in my data that have the response 'twice' for Forced Sexual Intercourse because of Authori
filter(data, AUTHSI == 3 | PLEASE == 4)
```

```
## # A tibble: 88 x 2,075
##    CODENUM      EDUC       MOB     DAYOB      B_YR      RACE   MARSTAT
##      <chr> <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl>
##  1 90-0029        12         7        NA      1972         1         1
##  2 90-0075        12        NA        NA      1971         1         1
##  3 90-0092        12        NA        NA      1972         2         1
##  4 90-0097        12         1        NA      1973         1         1
##  5 90-0123        12         5        NA      1972         1         1
##  6 90-0128        12        NA        NA      1971         1         1
##  7 90-0194        12         9        NA      1972         1         1
##  8 90-0295        12        11        NA      1972         1         1
##  9 90-0309        12        NA        NA      1972         2         1
## 10 90-0357        12        12        NA      1971         2         1
## # ... with 78 more rows, and 2068 more variables: RELINFL <dbl+lbl>,
## #   RELATT <dbl+lbl>, RELIG <dbl+lbl>, RELINFL1 <dbl+lbl>,
## #   RELATT1 <dbl+lbl>, GRAD <dbl+lbl>, CHARGE <dbl+lbl>, WHINY <dbl+lbl>,
## #   TOUGH <dbl+lbl>, GREATEST <dbl+lbl>, EMOTE <dbl+lbl>,
## #   GIVEIN <dbl+lbl>, BRAG <dbl+lbl>, GETMAD <dbl+lbl>, BUSY <dbl+lbl>,
## #   CENTER <dbl+lbl>, ALTRUE <dbl+lbl>, WHIMPY <dbl+lbl>, ROUGH <dbl+lbl>,
## #   COMPLAIN <dbl+lbl>, HELPFUL <dbl+lbl>, CONTESTS <dbl+lbl>,
## #   NOASSERT <dbl+lbl>, HOMEY <dbl+lbl>, GREED <dbl+lbl>, MEAN <dbl+lbl>,
## #   APPROV <dbl+lbl>, BOSSY <dbl+lbl>, NOHURT <dbl+lbl>, NAG <dbl+lbl>,
## #   NOEMPATH <dbl+lbl>, INDECIS <dbl+lbl>, FUSSY <dbl+lbl>,
## #   GIVEUP <dbl+lbl>, NOTRUST <dbl+lbl>, NOCRY <dbl+lbl>,
## #   CONFIDEN <dbl+lbl>, NUMONE <dbl+lbl>, BETTER <dbl+lbl>,
## #   REVENGE <dbl+lbl>, EMPATH <dbl+lbl>, FRIENDLY <dbl+lbl>,
## #   PLEASE <dbl+lbl>, NORISK <dbl+lbl>, TRUSTFUL <dbl+lbl>,
## #   FLUSTER <dbl+lbl>, NERVOUS <dbl+lbl>, BADNERV <dbl+lbl>,
## #   TENSE <dbl+lbl>, ANXIOUS <dbl+lbl>, NOCALM <dbl+lbl>, JUMPY <dbl+lbl>,
```

```
## #   RESTLESS <dbl+lbl>, RATTLED <dbl+lbl>, SHAKEH <dbl+lbl>,
## #   RELAX <dbl+lbl>, MOODY <dbl+lbl>, LOSPIRIT <dbl+lbl>, BLUE <dbl+lbl>,
## #   DEPRESS <dbl+lbl>, STRAIN <dbl+lbl>, CONTROL <dbl+lbl>,
## #   LOSEMIND <dbl+lbl>, STABLE <dbl+lbl>, NOSUCCES <dbl+lbl>,
## #   CRYING <dbl+lbl>, DEAD <dbl+lbl>, DUMPS <dbl+lbl>, SUICIDE <dbl+lbl>,
## #   NOFORWRD <dbl+lbl>, HAPPY <dbl+lbl>, SATISFID <dbl+lbl>,
## #   INTEREST <dbl+lbl>, CALM <dbl+lbl>, CHEERFUL <dbl+lbl>,
## #   ENJOY <dbl+lbl>, NOTENSE <dbl+lbl>, ADVENTUR <dbl+lbl>,
## #   XPECTDAY <dbl+lbl>, WAKEUP <dbl+lbl>, FUTRHOPE <dbl+lbl>,
## #   LOVED <dbl+lbl>, LUVRELAT <dbl+lbl>, LONELY <dbl+lbl>,
## #   IDISCUSS <dbl+lbl>, HDISCUSS <dbl+lbl>, IDISCUSD <dbl+lbl>,
## #   HDISCUSD <dbl+lbl>, IGOTINFO <dbl+lbl>, HGOTINFO <dbl+lbl>,
## #   IGOTHELP <dbl+lbl>, HGOTHELP <dbl+lbl>, IARGUED <dbl+lbl>,
## #   HARGUED <dbl+lbl>, IYELLED <dbl+lbl>, HYELLED <dbl+lbl>,
## #   ISULKED <dbl+lbl>, HSULKED <dbl+lbl>, ISTOMPED <dbl+lbl>,
## #   HSTOMPED <dbl+lbl>, ...
```

```
#print all observations without NAs
print(data, na.rm = TRUE)
```

```
## # A tibble: 1,580 x 2,075
##    CODENUM      EDUC       MOB     DAYOB      B_YR      RACE   MARSTAT
##      <chr> <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl>
##  1 90-0002        12         3        NA      1972         1         1
##  2 90-0003        12         6        NA      1972         2         1
##  3 90-0004        12         6        NA      1973         3         1
##  4 90-0005        12        10        NA      1972         1         1
##  5 90-0006        12         6        NA      1972         1         1
##  6 90-0007        12         1        NA      1972         3         1
##  7 90-0011        12         1        NA      1972         1         1
##  8 90-0014        12         4        NA      1972         1         1
##  9 90-0015        12         8        NA      1970         1         1
## 10 90-0016        12         1        NA      1972         1         1
## # ... with 1,570 more rows, and 2068 more variables: RELINFL <dbl+lbl>,
## #   RELATT <dbl+lbl>, RELIG <dbl+lbl>, RELINFL1 <dbl+lbl>,
## #   RELATT1 <dbl+lbl>, GRAD <dbl+lbl>, CHARGE <dbl+lbl>, WHINY <dbl+lbl>,
## #   TOUGH <dbl+lbl>, GREATEST <dbl+lbl>, EMOTE <dbl+lbl>,
## #   GIVEIN <dbl+lbl>, BRAG <dbl+lbl>, GETMAD <dbl+lbl>, BUSY <dbl+lbl>,
## #   CENTER <dbl+lbl>, ALTRUE <dbl+lbl>, WHIMPY <dbl+lbl>, ROUGH <dbl+lbl>,
## #   COMPLAIN <dbl+lbl>, HELPFUL <dbl+lbl>, CONTESTS <dbl+lbl>,
## #   NOASSERT <dbl+lbl>, HOMEY <dbl+lbl>, GREED <dbl+lbl>, MEAN <dbl+lbl>,
## #   APPROV <dbl+lbl>, BOSSY <dbl+lbl>, NOHURT <dbl+lbl>, NAG <dbl+lbl>,
## #   NOEMPATH <dbl+lbl>, INDECIS <dbl+lbl>, FUSSY <dbl+lbl>,
## #   GIVEUP <dbl+lbl>, NOTRUST <dbl+lbl>, NOCRY <dbl+lbl>,
## #   CONFIDEN <dbl+lbl>, NUMONE <dbl+lbl>, BETTER <dbl+lbl>,
## #   REVENGE <dbl+lbl>, EMPATH <dbl+lbl>, FRIENDLY <dbl+lbl>,
## #   PLEASE <dbl+lbl>, NORISK <dbl+lbl>, TRUSTFUL <dbl+lbl>,
## #   FLUSTER <dbl+lbl>, NERVOUS <dbl+lbl>, BADNERV <dbl+lbl>,
## #   TENSE <dbl+lbl>, ANXIOUS <dbl+lbl>, NOCALM <dbl+lbl>, JUMPY <dbl+lbl>,
## #   RESTLESS <dbl+lbl>, RATTLED <dbl+lbl>, SHAKEH <dbl+lbl>,
## #   RELAX <dbl+lbl>, MOODY <dbl+lbl>, LOSPIRIT <dbl+lbl>, BLUE <dbl+lbl>,
## #   DEPRESS <dbl+lbl>, STRAIN <dbl+lbl>, CONTROL <dbl+lbl>,
## #   LOSEMIND <dbl+lbl>, STABLE <dbl+lbl>, NOSUCCES <dbl+lbl>,
## #   CRYING <dbl+lbl>, DEAD <dbl+lbl>, DUMPS <dbl+lbl>, SUICIDE <dbl+lbl>,
## #   NOFORWRD <dbl+lbl>, HAPPY <dbl+lbl>, SATISFID <dbl+lbl>,
```

```
## #   INTEREST <dbl+lbl>, CALM <dbl+lbl>, CHEERFUL <dbl+lbl>,
## #   ENJOY <dbl+lbl>, NOTENSE <dbl+lbl>, ADVENTUR <dbl+lbl>,
## #   XPECTDAY <dbl+lbl>, WAKEUP <dbl+lbl>, FUTRHOPE <dbl+lbl>,
## #   LOVED <dbl+lbl>, LUVRELAT <dbl+lbl>, LONELY <dbl+lbl>,
## #   IDISCUSS <dbl+lbl>, HDISCUSS <dbl+lbl>, IDISCUSD <dbl+lbl>,
## #   HDISCUSD <dbl+lbl>, IGOTINFO <dbl+lbl>, HGOTINFO <dbl+lbl>,
## #   IGOTHELP <dbl+lbl>, HGOTHELP <dbl+lbl>, IARGUED <dbl+lbl>,
## #   HARGUED <dbl+lbl>, IYELLED <dbl+lbl>, HYELLED <dbl+lbl>,
## #   ISULKED <dbl+lbl>, HSULKED <dbl+lbl>, ISTOMPED <dbl+lbl>,
## #   HSTOMPED <dbl+lbl>, ...
```

8. `arrange()` can be used to rearrange rows according to any type of data. If you pass `arrange()` a character variable, for example, `R` will rearrange the rows in alphabetical order according to values of the variable. If you pass a factor variable, `R` will rearrange the rows according to the order of the levels in your factor (running `levels()` on the variable reveals this order).

By default, `arrange()` arranges the rows from smallest to largest. Rows with the smallest value of the variable will appear at the top of the data set. You can reverse this behavior with the `desc()` function. `arrange()` will reorder the rows from largest to smallest values of a variable if you wrap the variable name in `desc()` before passing it to `arrange()`.

a. Which variable(s) in your dataset would be logical to arrange your data on? Explain your reasoning.

It would be logical to arrange all of the variables in my dataset to go from lowest to highest because then all of the rows with the same level of response would be nearby.

b. Arrange your data by this/these variables and print the results.

```
#copying data_subset code from the last lab
data_subset <- data %>%
  select(RACE, NOASSERT, PLEASE, NORISK, TRUSTFUL, CONSENT, ATTEMPT, PRESSSI, AUTHSI, SEXACTS)

#arrange all variables with rows from least to greatest values

arrange(data_subset, RACE, NOASSERT, PLEASE, NORISK, TRUSTFUL, CONSENT, ATTEMPT, PRESSSI, AUTHSI, SEXAC
```

```
## # A tibble: 1,580 x 10
##         RACE  NOASSERT    PLEASE    NORISK  TRUSTFUL   CONSENT   ATTEMPT
##    <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl>
##  1         1         1         1         1         1         1         1
##  2         1         1         1         1         1         2         1
##  3         1         1         1         1         1         2         1
##  4         1         1         1         1         1         2         1
##  5         1         1         1         1         1         2         1
##  6         1         1         1         1         1         2         1
##  7         1         1         1         1         1         2         1
##  8         1         1         1         1         1         2         2
##  9         1         1         1         1         1         2         2
## 10         1         1         1         1         2         1         1
## # ... with 1,570 more rows, and 3 more variables: PRESSSI <dbl+lbl>,
## #   AUTHSI <dbl+lbl>, SEXACTS <dbl+lbl>
```

9. You can use any function you like in `summarise()` so long as the function can take a vector of data and return a single number. `R` contains many aggregating functions, as `dplyr` calls them:

- `min(x)` - minimum value of vector `x`.
- `max(x)` - maximum value of vector `x`.
- `mean(x)` - mean value of vector `x`.

- `median(x)` - median value of vector `x`.
- `quantile(x, p)` - pth quantile of vector `x`.
- `sd(x)` - standard deviation of vector `x`.
- `var(x)` - variance of vector `x`.
- `IQR(x)` - Inter Quartile Range (IQR) of vector `x`.
- `diff(range(x))` - total range of vector `x`.

a. Pick at least one variable of interest to your project analysis.
b. Print out at least three summary statistics using `summarise()`.

```
#mutate variables into numeric form so they can be read into summarise

data_subset %>%
  mutate(RACE = as.numeric(RACE),
         NOASSERT = as.numeric(NOASSERT),
         PLEASE = as.numeric(PLEASE),
         NORISK = as.numeric(NORISK),
         TRUSTFUL = as.numeric(TRUSTFUL),
         CONSENT = as.numeric(CONSENT),
         ATTEMPT = as.numeric(ATTEMPT),
         PRESSSI = as.numeric(PRESSSI),
         AUTHSI = as.numeric(AUTHSI),
         SEXACTS = as.numeric(SEXACTS)
)
```

```
## # A tibble: 1,580 x 10
##     RACE NOASSERT PLEASE NORISK TRUSTFUL CONSENT ATTEMPT PRESSSI AUTHSI
##    <dbl>    <dbl>  <dbl>  <dbl>    <dbl>   <dbl>   <dbl>   <dbl>  <dbl>
##  1     1        1      2      1        2       1       1       1      1
##  2     2        2      1      4        2       2       2       2      1
##  3     3        4      1      1        1       1       1       1      1
##  4     1        1      2      3        2       2       1       1      1
##  5     1        1      2      4        2       2       1       1      1
##  6     3        3      3      2        4       1       1       1      1
##  7     1        4      2      3        2       2       1       1      1
##  8     1        1      1      4        2       2       2       2      1
##  9     1        2      3      4        2       2       1       1      1
## 10     1        1      2      4        1       2       2       1      1
## # ... with 1,570 more rows, and 1 more variables: SEXACTS <dbl>
```

```
# print summary statistics for AUTHSI
summarise(data_subset)
```

```
## data frame with 0 columns and 0 rows
```

```
#to find the mode of AUTHSI
table(data_subset$AUTHSI)
```

```
##
##    1    2
## 1545   26
```

```
#to find the mode of PLEASE
table(data_subset$PLEASE)
```

```
##
##   1   2   3   4   5
## 708 485 249  88  40
```

10. `dplyr` provides several helpful aggregate functions of its own, in addition to the ones that are already defined in `R`. These include:

- `first(x)` - The first element of vector `x`.
- `last(x)` - The last element of vector `x`.
- `nth(x, n)` - The nth element of vector `x`.
- `n()` - The number of rows in the data.frame or group of observations that `summarise()` describes.
- `n_distinct(x)` - The number of unique values in vector `x`.

Next to these `dplyr`-specific functions, you can also turn a logical test into an aggregating function with `sum()` or `mean()`. A logical test returns a vector of TRUE's and FALSE's. When you apply `sum()` or `mean()` to such a vector, `R` coerces each `TRUE` to a `1` and each `FALSE` to a `0`. `sum()` then represents the total number of observations that passed the test; `mean()` represents the proportion.

a. Print out a summary of your data using at least two of these `dplyr`-specific aggregate functions.

```
# find out how many unique values there are in the vector
n_distinct(data_subset)
```

```
## [1] 848
```

```
# take the sum of the dataset
sum(data_subset)
```

```
## [1] NA
```

b. Why did you choose the ones you did? What did you learn about your data from these summaries?

I chose them to see what they did to my data and to better understand there purpose. I learned that my data has 848 unique values and that the sum() function does not work because of what type of data it is.

11. You can also combine `group_by()` with `mutate()`. When you mutate grouped data, `mutate()` will calculate the new variables independently for each group. This is particularly useful when `mutate()` uses the `rank()` function, that calculates within-group rankings. `rank()` takes a group of values and calculates the rank of each value within the group, e.g.

```
rank(c(21, 22, 24, 23))
```

has the output

```
[1] 1 2 4 3
```

As with `arrange()`, `rank()` ranks values from the smallest to the largest.

a. Using the `%>%` operator, first group your dataset by a meaningful variable, then perform a mutation that you're interested in.
b. What do the results tell you about different groups in you data?

The results tell show me a new variable that combine both AUTHSI and PRESSSI which allows me to look at the total amount of coerced intercourse in the data, no matter the reason. This allows me to compare this new variable directly with PLEASE or NOASSERT to see if there is a correlation.

```
library("dplyr")
# group dataset by SEXACTS variable
by_SEXACTS <- data_subset %>% group_by(SEXACTS, NOASSERT)

#mutate the data so it drops the CONSENT variable because it is not relevant
by_SEXACTS %>% mutate(RAPE = AUTHSI, PRESSSI)
```

```
## Warning in mutate_impl(.data, dots): Vectorizing 'labelled' elements may
## not preserve their attributes
```

```
## Warning in mutate_impl(.data, dots): Vectorizing 'labelled' elements may
## not preserve their attributes

## Warning in mutate_impl(.data, dots): Vectorizing 'labelled' elements may
## not preserve their attributes

## Warning in mutate_impl(.data, dots): Vectorizing 'labelled' elements may
## not preserve their attributes

## Warning in mutate_impl(.data, dots): Vectorizing 'labelled' elements may
## not preserve their attributes

## Warning in mutate_impl(.data, dots): Vectorizing 'labelled' elements may
## not preserve their attributes

## Warning in mutate_impl(.data, dots): Vectorizing 'labelled' elements may
## not preserve their attributes

## Warning in mutate_impl(.data, dots): Vectorizing 'labelled' elements may
## not preserve their attributes

## Warning in mutate_impl(.data, dots): Vectorizing 'labelled' elements may
## not preserve their attributes

## Warning in mutate_impl(.data, dots): Vectorizing 'labelled' elements may
## not preserve their attributes

## Warning in mutate_impl(.data, dots): Vectorizing 'labelled' elements may
## not preserve their attributes

## Warning in mutate_impl(.data, dots): Vectorizing 'labelled' elements may
## not preserve their attributes

## Warning in mutate_impl(.data, dots): Vectorizing 'labelled' elements may
## not preserve their attributes

## Warning in mutate_impl(.data, dots): Vectorizing 'labelled' elements may
## not preserve their attributes

## Warning in mutate_impl(.data, dots): Vectorizing 'labelled' elements may
## not preserve their attributes

## Warning in mutate_impl(.data, dots): Vectorizing 'labelled' elements may
## not preserve their attributes

## Warning in mutate_impl(.data, dots): Vectorizing 'labelled' elements may
## not preserve their attributes

## Warning in mutate_impl(.data, dots): Vectorizing 'labelled' elements may
## not preserve their attributes

## Warning in mutate_impl(.data, dots): Vectorizing 'labelled' elements may
## not preserve their attributes
```

```
## Warning in mutate_impl(.data, dots): Vectorizing 'labelled' elements may
## not preserve their attributes

## Warning in mutate_impl(.data, dots): Vectorizing 'labelled' elements may
## not preserve their attributes

## Warning in mutate_impl(.data, dots): Vectorizing 'labelled' elements may
## not preserve their attributes

## Warning in mutate_impl(.data, dots): Vectorizing 'labelled' elements may
## not preserve their attributes

## Warning in mutate_impl(.data, dots): Vectorizing 'labelled' elements may
## not preserve their attributes

## Warning in mutate_impl(.data, dots): Vectorizing 'labelled' elements may
## not preserve their attributes

## Warning in mutate_impl(.data, dots): Vectorizing 'labelled' elements may
## not preserve their attributes

## Warning in mutate_impl(.data, dots): Vectorizing 'labelled' elements may
## not preserve their attributes

## Warning in mutate_impl(.data, dots): Vectorizing 'labelled' elements may
## not preserve their attributes

## # A tibble: 1,580 x 11
## # Groups:   SEXACTS, NOASSERT [14]
##          RACE  NOASSERT    PLEASE    NORISK  TRUSTFUL   CONSENT   ATTEMPT
##     <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl>
##  1          1         1         2         1         2         1         1
##  2          2         2         1         4         2         2         2
##  3          3         4         1         1         1         1         1
##  4          1         1         2         3         2         2         1
##  5          1         1         2         4         2         2         1
##  6          3         3         3         2         4         1         1
##  7          1         4         2         3         2         2         1
##  8          1         1         1         4         2         2         2
##  9          1         2         3         4         2         2         1
## 10          1         1         2         4         1         2         2
## # ... with 1,570 more rows, and 4 more variables: PRESSSI <dbl>,
## #   AUTHSI <dbl+lbl>, SEXACTS <dbl+lbl>, RAPE <dbl>
```

```r
#arrange the grouped dataset by_SEXACTs
data_subset %>% arrange(by_SEXACTS)
```

```
## # A tibble: 10 x 10
##          RACE  NOASSERT    PLEASE    NORISK  TRUSTFUL   CONSENT   ATTEMPT
##     <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl> <dbl+lbl>
##  1          1         1         1         4         2         2         2
##  2          1         1         2         1         2         1         1
##  3          1         1         2         3         2         2         1
##  4          1         1         2         4         1         2         2
##  5          1         1         2         4         2         2         1
```

```
##  6         1         2         3         4         2         2         1
##  7         1         4         2         3         2         2         1
##  8         2         2         1         4         2         2         2
##  9         3         3         3         2         4         1         1
## 10         3         4         1         1         1         1         1
## # ... with 3 more variables: PRESSSI <dbl+lbl>, AUTHSI <dbl+lbl>,
## #   SEXACTS <dbl+lbl>
```

12. The exercises so far have tried to get you to think about how to apply the five verbs of `dplyr` to your data.

    a. Are there any specific transformations you want to make to your data? What are they and what aspect of your research question will they help you to answer?
    b. In a code chunk below, carry out all the data transformations you wish to perform on your data. Utilize the `%>%` operator to tie multiple commands together and make your code more readable and efficient. Remember to comment your code so it is clear why you doing things a certain way.

My data is already very organized so the only transformation I should need to do is create a smaller subset of my data that is easier to work with. I already created on subset called data_subset, but I am going to make one more that has fewer columns that are the most relevant to my project called data_condensed. As the project goes along if I realize I need to transform any part of the data I will execute those transformations then.

```
# create subset of data with only relevant columns
data_condensed <- data %>%
  select(RACE, NOASSERT, PLEASE, AUTHSI, PRESSSI)
```