Manipulating data in `R`

Homework

Prof. Bisbee

Due Date: 2024-08-29

Agenda

We're going to go quickly back over loading data and then return to the topic of filtering, selecting and arranging data. We'll then turn to some calculations using the concepts of summarizing (self explanatory) and mutating (creating new variables).

Rmarkdown

To recap, an Rmarkdown file contains two basic elements: text and code. That text and code can be combined or "knitted" into a variety of different document formats. Lets get you started by creating your own Rmarkdown file and knitting it.

Load relevant libraries

```
library(tidyverse)
```

Load The Data

Remember to load the data from GitHub (https://github.com/jbisbee1/DS1000_F2024/raw/main/data/sc_debt.Rds) and save it to the `data` folder you created. You should then open it in `R` by assigning it to an object with the `<-` command.

```
df<-read_rds("https://github.com/jbisbee1/DS1000_F2024/raw/main/data/sc_debt.Rds")
names(df)
```

```
##  [1] "unitid"         "instnm"         "stabbr"         "grad_debt_mdn"
##  [5] "control"        "region"         "preddeg"        "openadmp"
##  [9] "adm_rate"       "ccbasic"        "sat_avg"        "md_earn_wne_p6"
## [13] "ugds"           "costt4_a"       "selective"      "research_u"
```

Name	Definition
unitid	Unit ID
instnm	Institution Name
stabbr	State Abbreviation

Name	Definition
grad_debt_mdn	Median Debt of Graduates
control	Control Public or Private
region	Census Region
preddeg	Predominant Degree Offered: Associates or Bachelors
openadmp	Open Admissions Policy: 1= Yes, 2=No,3=No 1st time students
adm_rate	Admissions Rate: proportion of applications accepted
ccbasic	Type of institution– see here (https://data.ed.gov/dataset/9dc70e6b-8426-4d71-b9d5-70ce6094a3f4/resource/658b5b83-ac9f-4e41-913e-9ba9411d7967/download/collegescorecarddatadictionary_01192021.xlsx)
selective	Institution admits fewer than 10 % of applicants, 1=Yes, 0=No
research_u	Institution is a research university 1=Yes, 0=No
sat_avg	Average Sat Scores
md_earn_wne_p6	Average Earnings of Recent Graduates
ugds	Number of undergraduates

Looking at datasets

We can use “glimpse” to see what’s in a dataset. This gives a very quick rundown of the variables and the first few observations.

```
glimpse(df)
```

```
## Rows: 2,546
## Columns: 16
## $ unitid         <int> 100654, 100663, 100690, 100706, 100724, 100751, 100760,…
## $ instnm         <chr> "Alabama A & M University", "University of Alabama at B…
## $ stabbr         <chr> "AL", "AL", "AL", "AL", "AL", "AL", "AL", "AL", "AL", "…
## $ grad_debt_mdn  <int> 33375, 22500, 27334, 21607, 32000, 23250, 12500, 19500,…
## $ control        <chr> "Public", "Public", "Private", "Public", "Public", "Pub…
## $ region         <chr> "Southeast", "Southeast", "Southeast", "Southeast", "So…
## $ preddeg        <chr> "Bachelor's", "Bachelor's", "Associate", "Bachelor's", …
## $ openadmp       <int> 2, 2, 1, 2, 2, 2, 1, NA, 2, 2, 2, 1, 1, 2, 1, 1, 2, 2, …
## $ adm_rate       <dbl> 0.9175, 0.7366, NA, 0.8257, 0.9690, 0.8268, NA, NA, 0.9…
## $ ccbasic        <int> 18, 15, 20, 16, 19, 15, 2, 22, 18, 15, 21, 1, 5, 19, 7,…
## $ sat_avg        <int> 939, 1234, NA, 1319, 946, 1261, NA, NA, 1082, 1300, 123…
## $ md_earn_wne_p6 <int> 25200, 35100, 30700, 36200, 22600, 37400, 23100, 33400,…
## $ ugds           <int> 5271, 13328, 365, 7785, 3750, 31900, 1201, 2677, 4407, …
## $ costt4_a       <int> 23053, 24495, 14800, 23917, 21866, 29872, 10493, NA, 19…
## $ selective      <dbl> 0, 0, NA, 0, 0, 0, NA, NA, 0, 0, 0, NA, NA, 0, NA, NA, …
## $ research_u     <dbl> 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0…
```

Types of Variables

Notice that for each variable, it shows a different type, in angle brackets `<>`. So for instance, `instnm` has a type of `<chr>`. This is short for character– it's also called a string variable.

Here are the types of data in this dataset

- `<int>` Integer data
- `<chr>` Character or string data
- `<dbl>` Double, (double-precision floating point) or just numeric data– can be measured down to an arbitrary number of data points.

This information is useful, because we wouldn't want to try to run some kind of numeric analysis on string data. The average of institution names wouldn't make a lot of sense (but it would probably be Southeast North State University College).

We'll talk more about data types later, but we should also quickly note that there are some variables in this dataset where the numbers represent a characteristic, rather and a measurement. For instance, the variable `research_u` is set up—coded— such that a "1" indicates that the college is a research university and a "0" indicates that it is not a research university. The 1 and 0 don't measure anything, they just indicate a characteristic.

Filter, Select, Arrange

Today, we'll pick up where we left off– with the key commands of filter, select, and arrange.

In exploring data, many times we want to look at smaller parts of the dataset. There are three commands we'll use today that help with this.

- `filter` selects only those cases or rows that meet some logical criteria.

- `select` selects only those variables or columns that meet some criteria

- `arrange` arranges the rows of a dataset in the way we want.

For more on these, please see this vignette (https://cran.rstudio.com/web/packages/dplyr/vignettes/introduction.html).

We can look at the first 5 rows:

```
head(df)
```

```
## # A tibble: 6 × 16
##   unitid instnm    stabbr grad_debt_mdn control region preddeg openadmp adm_rate
##    <int> <chr>     <chr>          <int> <chr>   <chr>  <chr>      <int>    <dbl>
## 1 100654 Alabama … AL             33375 Public  South… Bachel…        2    0.918
## 2 100663 Universi… AL             22500 Public  South… Bachel…        2    0.737
## 3 100690 Amridge … AL             27334 Private South… Associ…        1   NA
## 4 100706 Universi… AL             21607 Public  South… Bachel…        2    0.826
## 5 100724 Alabama … AL             32000 Public  South… Bachel…        2    0.969
## 6 100751 The Univ… AL             23250 Public  South… Bachel…        2    0.827
## # ℹ 7 more variables: ccbasic <int>, sat_avg <int>, md_earn_wne_p6 <int>,
## #   ugds <int>, costt4_a <int>, selective <dbl>, research_u <dbl>
```

Or the last 5 rows:

```
tail(df)
```

```
## # A tibble: 6 × 16
##   unitid instnm    stabbr grad_debt_mdn control region preddeg openadmp adm_rate
##    <int> <chr>     <chr>          <int> <chr>   <chr>  <chr>      <int>    <dbl>
## 1 493716 Yeshiva … NJ                NA Private North… Associ…        2    0.477
## 2 493725 Universi… AR                NA Public  South… Bachel…        1   NA
## 3 493822 College … RI                NA Private New E… Bachel…        1   NA
## 4 494630 Christ M… TX                NA Private South… Bachel…        1   NA
## 5 494685 Urshan C… MO                NA Private Plains Bachel…        2    0.836
## 6 494737 Yeshiva … NY                NA Private North… Bachel…        1   NA
## # ℹ 7 more variables: ccbasic <int>, sat_avg <int>, md_earn_wne_p6 <int>,
## #   ugds <int>, costt4_a <int>, selective <dbl>, research_u <dbl>
```

Using filter in combination with other commands

`filter` can be used with any command that retruns true or false. This can be really powerful, for instance the command `str_detect` "detects" the relevant string in the data, so we can look for any college with the word "Colorado" in its name.

```
df%>%
  filter(str_detect(instnm,"Colorado"))%>%
  select(instnm,adm_rate,sat_avg)
```

```
## # A tibble: 12 × 3
##    instnm                                                adm_rate sat_avg
##    <chr>                                                    <dbl>   <int>
##  1 University of Colorado Denver/Anschutz Medical Campus    0.673    1124
##  2 University of Colorado Colorado Springs                  0.872    1136
##  3 University of Colorado Boulder                           0.784    1276
##  4 Colorado Christian University                           NA          NA
##  5 Colorado College                                         0.135      NA
##  6 Colorado School of Mines                                 0.531    1342
##  7 Colorado State University-Fort Collins                   0.814    1204
##  8 Colorado Mesa University                                 0.782    1063
##  9 University of Northern Colorado                          0.908    1096
## 10 Colorado State University Pueblo                         0.930    1047
## 11 Western Colorado University                              0.842    1114
## 12 Colorado State University-Global Campus                  0.986    1048
```

We can combine this with the `|` operator, which remember stands for "or." Let's say we want all the institutions in Colorado OR California.

```
df%>%
  filter(str_detect(instnm,"Colorado") | str_detect(instnm,"California"))%>%
  select(instnm,adm_rate,sat_avg)
```

```
## # A tibble: 57 × 3
##    instnm                                                adm_rate sat_avg
##    <chr>                                                    <dbl>   <int>
##  1 California Institute of Integral Studies                NA          NA
##  2 California Baptist University                            0.783     1096
##  3 California College of the Arts                           0.850       NA
##  4 California Institute of Technology                       0.0642    1557
##  5 California Lutheran University                           0.714     1168
##  6 California Polytechnic State University-San Luis Obispo  0.284     1342
##  7 California State University-Bakersfield                  0.807       NA
##  8 California State University-Stanislaus                   0.893       NA
##  9 California State University-San Bernardino               0.686      985
## 10 California State Polytechnic University-Pomona           0.546     1143
## # ℹ 47 more rows
```

We can also put this together in one (notice that everything goes inside the quotes)

```
df%>%
  filter(str_detect(instnm,"Colorado|California"))%>%
  select(instnm,adm_rate,sat_avg)
```

```
## # A tibble: 57 × 3
##    instnm                                                adm_rate sat_avg
##    <chr>                                                    <dbl>   <int>
##  1 California Institute of Integral Studies                NA          NA
##  2 California Baptist University                            0.783     1096
##  3 California College of the Arts                           0.850       NA
##  4 California Institute of Technology                       0.0642    1557
##  5 California Lutheran University                           0.714     1168
##  6 California Polytechnic State University-San Luis Obispo  0.284     1342
##  7 California State University-Bakersfield                  0.807       NA
##  8 California State University-Stanislaus                   0.893       NA
##  9 California State University-San Bernardino               0.686      985
## 10 California State Polytechnic University-Pomona           0.546     1143
## # ℹ 47 more rows
```

Reminder: logical operators

Here are (many of) the logical operators that we use in R:

- `>` , `<` : greater than, less than
- `>=` , `<=` : greater than or equal to, less than or equal to
- `!` :not, as in `!=` not equal to
- `&` AND
- `|` OR

Quick Exercise Select colleges that are from Texas AND have the word “community” in their name (the name variable is `instnm`).

```
# INSERT CODE HERE
```

Extending Select

Select can also be used with other characteristics.

For quick guide on this: https://dplyr.tidyverse.org/reference/select.html (https://dplyr.tidyverse.org/reference/select.html)

For example, we can select just variables that contain the word “region”

```
df%>%
  select(contains("region"))
```

```
## # A tibble: 2,546 × 1
##    region
##    <chr>
##  1 Southeast
##  2 Southeast
##  3 Southeast
##  4 Southeast
##  5 Southeast
##  6 Southeast
##  7 Southeast
##  8 Southeast
##  9 Southeast
## 10 Southeast
## # ℹ 2,536 more rows
```

`contains()` and `matches()` are equivalent functions

```
df %>%
  select(matches('region'))
```

```
## # A tibble: 2,546 × 1
##    region
##    <chr>
##  1 Southeast
##  2 Southeast
##  3 Southeast
##  4 Southeast
##  5 Southeast
##  6 Southeast
##  7 Southeast
##  8 Southeast
##  9 Southeast
## 10 Southeast
## # ℹ 2,536 more rows
```

We can augment these with the logical operators listed above

```
# Removes columns with "inst" in their names
df %>%
  select(!matches('inst'))
```

```
## # A tibble: 2,546 × 15
##    unitid stabbr grad_debt_mdn control region  preddeg openadmp adm_rate ccbasic
##     <int> <chr>          <int> <chr>   <chr>   <chr>      <int>    <dbl>   <int>
##  1 100654 AL             33375 Public  Southe… Bachel…        2    0.918      18
##  2 100663 AL             22500 Public  Southe… Bachel…        2    0.737      15
##  3 100690 AL             27334 Private Southe… Associ…        1   NA          20
##  4 100706 AL             21607 Public  Southe… Bachel…        2    0.826      16
##  5 100724 AL             32000 Public  Southe… Bachel…        2    0.969      19
##  6 100751 AL             23250 Public  Southe… Bachel…        2    0.827      15
##  7 100760 AL             12500 Public  Southe… Associ…        1   NA           2
##  8 100812 AL             19500 Public  Southe… Bachel…       NA   NA          22
##  9 100830 AL             24826 Public  Southe… Bachel…        2    0.904      18
## 10 100858 AL             21281 Public  Southe… Bachel…        2    0.807      15
## # ℹ 2,536 more rows
## # ℹ 6 more variables: sat_avg <int>, md_earn_wne_p6 <int>, ugds <int>,
## #   costt4_a <int>, selective <dbl>, research_u <dbl>
```

```
# Selects columns with either "inst" or an underline in their names
df %>%
  select(matches('inst|_'))
```

```
## # A tibble: 2,546 × 7
##    instnm      grad_debt_mdn adm_rate sat_avg md_earn_wne_p6 costt4_a research_u
##    <chr>               <int>    <dbl>   <int>          <int>    <int>      <dbl>
##  1 Alabama A …         33375    0.918     939          25200    23053          0
##  2 University…         22500    0.737    1234          35100    24495          0
##  3 Amridge Un…         27334   NA          NA          30700    14800          0
##  4 University…         21607    0.826    1319          36200    23917          1
##  5 Alabama St…         32000    0.969     946          22600    21866          0
##  6 The Univer…         23250    0.827    1261          37400    29872          0
##  7 Central Al…         12500   NA          NA          23100    10493          0
##  8 Athens Sta…         19500   NA          NA          33400       NA          0
##  9 Auburn Uni…         24826    0.904    1082          30100    19849          0
## 10 Auburn Uni…         21281    0.807    1300          39500    31590          0
## # ℹ 2,536 more rows
```

We can also select just variables by their type using `where()`

```
# Select only numeric variables
df%>%
  select(where(is.numeric))
```

```
## # A tibble: 2,546 × 11
##    unitid grad_debt_mdn openadmp adm_rate ccbasic sat_avg md_earn_wne_p6  ugds
##     <int>         <int>    <int>    <dbl>   <int>   <int>          <int> <int>
##  1 100654         33375        2    0.918      18     939          25200  5271
##  2 100663         22500        2    0.737      15    1234          35100 13328
##  3 100690         27334        1   NA          20      NA          30700   365
##  4 100706         21607        2    0.826      16    1319          36200  7785
##  5 100724         32000        2    0.969      19     946          22600  3750
##  6 100751         23250        2    0.827      15    1261          37400 31900
##  7 100760         12500        1   NA           2      NA          23100  1201
##  8 100812         19500       NA   NA          22      NA          33400  2677
##  9 100830         24826        2    0.904      18    1082          30100  4407
## 10 100858         21281        2    0.807      15    1300          39500 24209
## # ℹ 2,536 more rows
## # ℹ 3 more variables: costt4_a <int>, selective <dbl>, research_u <dbl>
```

Quick Exercise Use the same setup to select only character variables (`is.character`)

```
# INSERT CODE HERE
```

Summarizing Data

To summarize data, we use the `summarize` command. Inside that command, we tell R two things: what to call the new object (a data frame, really) that we're creating, and what numerical summary we would like. The code below summarizes median debt for the colleges in the dataset by calculating the average of median debt for all institutions.

Notice that inside the `mean` command

```
df%>%
  summarize(mean_debt=mean(grad_debt_mdn,na.rm=TRUE))
```

```
## # A tibble: 1 × 1
##   mean_debt
##       <dbl>
## 1    19646.
```

Quick Exercise Summarize the average entering SAT scores in this dataset.

```
# INSERT CODE HERE
```

Combining Commands

We can also combine commands, so that summaries are done on only a part of the dataset. Below, we summarize median debt for selective schools, and not very selective schools.

```
df%>%
  filter(stabbr=="CA")%>%
  summarize(mean_adm_rate=mean(adm_rate,na.rm=TRUE))
```

```
## # A tibble: 1 × 1
##   mean_adm_rate
##           <dbl>
## 1         0.592
```

Quick Exercise Calculate average earnings for schools where SAT>1200 & the admissions rate is between 10 and 20 percent.

```
# INSERT CODE HERE
```

Mutate

`mutate` is the verb for changing variables in R. Let's say we want to create a variable that's set to 1 if the college admits less than 10 percent of the students who apply.

```
df<-df%>%
  mutate(selective=ifelse(adm_rate<=.1,1,0))
```

The `ifelse()` function is powerful. It allows us to create one value if a logical expression is `TRUE`, and another value if the logical expression is `FALSE`. The inputs are: `ifelse([LOGIC],[VALUE IF TRUE],[VALUE IF FALSE])`. In this example, the "logical expression" is `adm_rate <= 0.1`. For every row where this is `TRUE`, we get the value `1`. For every row where this is `FALSE`, we get the value `0`.

Quick Exercise Create a new variable that's set to 1 if the college has more than 10,000 undergraduate students

```
# INSERT CODE HERE
```

Or what if we want to create another new variable that changes the admissions rate from its current proportion to a percent?

```
df<-df%>%
  mutate(adm_rate_pct=adm_rate*100)
```

To figure out if that worked we can use `summarize`

```
df%>%
  summarize(mean_adm_rate_pct=mean(adm_rate_pct,na.rm=TRUE))
```

```
## # A tibble: 1 × 1
##   mean_adm_rate_pct
##               <dbl>
## 1              67.9
```

Grouping

Above, we calculated the `mean_adm_rate` for schools in California by combining a `filter()` command with a `summarise()` command. Let's use the same approach to calculate the average SAT score for schools that are selective and for those that aren't.

```
# Mean SAT for selective schools
df %>%
  filter(selective == 1) %>%
  summarise(SATavg = mean(sat_avg,na.rm=T))
```

```
## # A tibble: 1 × 1
##   SATavg
##    <dbl>
## 1  1510.
```

```
# Mean SAT for non-selective schools
df %>%
  filter(selective == 0) %>%
  summarise(SATavg = mean(sat_avg,na.rm=T))
```

```
## # A tibble: 1 × 1
##   SATavg
##    <dbl>
## 1  1135.
```

This works, but requires two separate chunks of code. We can streamline this analysis with the `group_by()` function, which tells `R` to run a command on each group separately. Thus:

```
df %>%
  group_by(selective) %>%
  summarise(SATavg = mean(sat_avg,na.rm=T))
```

```
## # A tibble: 3 × 2
##   selective SATavg
##       <dbl>  <dbl>
## 1         0  1135.
## 2         1  1510.
## 3        NA   NaN
```

Quick Exercise Do the same, but calculate the average SAT score for each state, using `group_by()` .

```
# INSERT CODE HERE
```