Icebreaker

What's something fun you did yesterday?

SOC245: Visualizing Data
Precept 3: Understanding Spread and Counts vs Proportions

Chris Felton and Vikram Ramaswamy

Freshman Scholars Institute
Princeton University

July 14, 2022

Recall from Lecture

- Spread of data: Range, IQR, Variance.
  - How are each defined? What different information do they provide?
- Counts vs proportions:
  - What information does each give us?

Set up

- Like Tuesday, go ahead and read in the dataset `"acs.rds"`

Set up

- Like Tuesday, go ahead and read in the dataset `"acs.rds"`
- Remember to load the package `tidyverse`! (We don't need this right now, but we're going to need it soon)

Set up

```
library("tidyverse")
```

```
acsdata <- readRDS("/cloud/project/acs.rds")
```

Understanding spread of this data

- Let's first compute the mean and the median of this data.
- Recall: `summary` can help you find the information about the mean and the median. It also displays quartiles.

Computing spread: Sample variance and standard deviation

- Sample variance: $\hat{\sigma}^2 = \frac{\sum_{i=1}^{n}(\bar{X} - X_i)^2}{n-1}$

```
var(acsdata$HINCP)
```

Computing spread: Sample variance and standard deviation

- Sample variance: $\hat{\sigma}^2 = \frac{\sum_{i=1}^{n}(\bar{X} - X_i)^2}{n-1}$

```
var(acsdata$HINCP)
```

- Standard deviation: Square-root of variance

```
sd(acsdata$HINCP)
```

Computing spread: Interquartile range

- IQR: defined as the difference between the 75th and 25th percentiles.

```
IQR(acsdata$HINCP)
```

- Suppose we want to measure the central tendency and spread for different populations (for example, men and women)

- Suppose we want to measure the central tendency and spread for different populations (for example, men and women)
- We could use `filter` and create 2 new datasets...

- Suppose we want to measure the central tendency and spread for different populations (for example, men and women)
- We could use `filter` and create 2 new datasets...
- `tidyverse` lets us do something a little easier to understand: we can **group** the observations in the dataset based on certain features.

- Suppose we want to measure the central tendency and spread for different populations (for example, men and women)
- We could use `filter` and create 2 new datasets...
- `tidyverse` lets us do something a little easier to understand: we can **group** the observations in the dataset based on certain features.
- We can then **summarize** the dataset, based on the group

group_by

Syntax:

```
acs_SEX <- group_by(acsdata, SEX)
```

group_by

Syntax:

```
acs_SEX <- group_by(acsdata, SEX)
```

- View acs_SEX. What's different?

group_by

```
> acsdata
# A tibble: 2,777 × 10
     SEX  AGEP RACWHT RACBLK RACASN RACAIAN RACSOR RACHISP  SCHL HINCP
   <dbl> <dbl>  <dbl>  <dbl>  <dbl>   <dbl>  <dbl>   <int> <dbl> <dbl>
 1     2    35      1      0      0       0      0       0    20 50800
 2     2    42      0      1      0       0      0       0    16 65800
 3     1    33      0      1      0       0      0       0    21 65800
 4     2    22      0      1      0       0      0       0    16 65800
 5     1    17      0      1      0       0      0       0    15 65800
 6     1    11      0      1      0       0      0       0     9 65800
 7     1    91      0      1      0       0      0       0    14 37800
 8     2    90      0      1      0       0      0       0    18 37800
 9     1    28      1      0      0       0      0       0    21 85000
10     2    64      1      0      0       0      0       0    17 50600
# … with 2,767 more rows
```

```
> acs_SEX
# A tibble: 2,777 x 10
# Groups:   SEX [2]
     SEX  AGEP RACWHT RACBLK RACASN RACAIAN RACSOR RACHISP  SCHL HINCP
   <dbl> <dbl>  <dbl>  <dbl>  <dbl>   <dbl>  <dbl>   <int> <dbl> <dbl>
 1     2    35      1      0      0       0      0       0    20 50800
 2     2    42      0      1      0       0      0       0    16 65800
 3     1    33      0      1      0       0      0       0    21 65800
 4     2    22      0      1      0       0      0       0    16 65800
 5     1    17      0      1      0       0      0       0    15 65800
 6     1    11      0      1      0       0      0       0     9 65800
 7     1    91      0      1      0       0      0       0    14 37800
 8     2    90      0      1      0       0      0       0    18 37800
 9     1    28      1      0      0       0      0       0    21 85000
10     2    64      1      0      0       0      0       0    17 50600
# … with 2,767 more rows
```

Summarising data

Try running

```
summarize(acs_SEX, median=median(HINCP))
```

`summarize()`

- Look up `?summarize`
- `summarize` takes as input a dataset and a series of **summary functions** and returns a type of object called a **data frame**.

`summarize()`

- Look up `?summarize`
- `summarize` takes as input a dataset and a series of **summary functions** and returns a type of object called a **data frame**.
- This is kind of like a table, but more on this in a bit!
- **summary functions** are any functions that combine the data in some way, for example, `mean()`, `median()`, `var()` are all summary functions.

summarize()

- Look up `?summarize`
- `summarize` takes as input a dataset and a series of **summary functions** and returns a type of object called a **data frame**.
- This is kind of like a table, but more on this in a bit!
- **summary functions** are any functions that combine the data in some way, for example, `mean()`, `median()`, `var()` are all summary functions.
- Names given to these functions will be the names of the columns of the returned table.

Try running

```
summarize(acs_SEX, med=median(HINCP))
```

What changes?

You try!

Can you figure out how to get the IQR of these groups as well?

Data frames in R

- **Data frames** are tables or 2-dimensional structures.

Data frames in R

- **Data frames** are tables or 2-dimensional structures.
- Each column contains one variable, and each row contains one observation, i.e, one value from each column.

Data frames in R

- **Data frames** are tables or 2-dimensional structures.
- Each column contains one variable, and each row contains one observation, i.e, one value from each column.
- Each column can store values that are numbers or strings.

Data frames in R

- **Data frames** are tables or 2-dimensional structures.
- Each column contains one variable, and each row contains one observation, i.e, one value from each column.
- Each column can store values that are numbers or strings.
- Each column must have a name, and must have the same number of items.

Data frames in R

- **Data frames** are tables or 2-dimensional structures.
- Each column contains one variable, and each row contains one observation, i.e, one value from each column.
- Each column can store values that are numbers or strings.
- Each column must have a name, and must have the same number of items.
- Can you give an example of a data frame?

Data frames in R

- **Data frames** are tables or 2-dimensional structures.
- Each column contains one variable, and each row contains one observation, i.e, one value from each column.
- Each column can store values that are numbers or strings.
- Each column must have a name, and must have the same number of items.
- Can you give an example of a data frame?
- The datasets we're working with are stored as data frames!

The pipe operator %>%

- Notice that we kept passing the result of the function into the next

```
acs_SEX <- group_by(acsdata, SEX)
summarize(acs_SEX, median=median(HINCP))
```

The pipe operator %>%

- Notice that we kept passing the result of the function into the next

```
acs_SEX <- group_by(acsdata, SEX)
summarize(acs_SEX, median=median(HINCP))
```

- The pipe operator %>% lets us do this in an easier way!

The pipe operator %>%: One pipe

- Let's see a simple example of this operator. Try running

```
acsdata %>%
    group_by(SEX)
```

The pipe operator %>%: One pipe

- Let's see a simple example of this operator. Try running

```
acsdata %>%
    group_by(SEX)
```

- This is equivalent to `group_by(acsdata, SEX)`
- The pipe operator passes `acsdata` as the first parameter of the `group_by` function

The pipe operator %>%: One pipe

- Similarly, we can write

```
acsdata %>%
    filter(SEX==1)
```

The pipe operator %>%: One pipe

- Similarly, we can write

```
acsdata %>%
    filter(SEX==1)
```

- This is equivalent to `filter(acsdata, SEX==1)`

The pipe operator %>%: Two pipes

- Note that we don't need to just have one pipe!

The pipe operator %>%: Two pipes

- Note that we don't need to just have one pipe!

```
acsdata %>%
    group_by(SEX) %>%
    summarize(median=median(HINCP))
```

- What do you think happens here?

The pipe operator %>%: Two pipes

- Note that we don't need to just have one pipe!

```
acsdata %>%
    group_by(SEX) %>%
    summarize(median=median(HINCP))
```

- What do you think happens here?
- This is equivalent to

```
summarize(group_by(acsdata, SEX),
    median = median(HINCP))
```

The pipe operator %>%: Two pipes

- Note that we don't need to just have one pipe!

```
acsdata %>%
    group_by(SEX) %>%
    summarize(median=median(HINCP))
```

- Similar to `ggplot`, %>% is designed to mimic human thought.

The pipe operator %>%: Two pipes

- Note that we don't need to just have one pipe!

```
acsdata %>%
    group_by(SEX) %>%
    summarize(median=median(HINCP))
```

- Similar to `ggplot`, %>% is designed to mimic human thought.
- "I need to first group this data by SEX, and then summarize it and then . . ."

The pipe operator %>%: More pipes

- What does this code do?

```
acsdata %>%
    filter(RACBLK==1) %>%
    group_by(AGEP<=25) %>%
    summarize(mean=mean(HINCP))
```

You try!

Can you rewrite the code you used to get the median and IQR of the household income grouped by sex using the %>% operator?

Motivation

For the rest of today, we'll consider the following question:

What are the counts and fraction of individuals live below the poverty line, and how does this vary by race?

Set up

Let's start by creating 2 datasets: one of people who are white, one of people who are Black.
Hint: Use the `filter` command

Set up

Let's start by creating 2 datasets: one of people who are white, one of people who are Black.
Hint: Use the `filter` command

```
acs_WHT <- filter(acsdata, RACWHT==1)
acs_BLK <- filter(acsdata, RACBLK==1)
```

Set up

Let's start by creating 2 datasets: one of people who are white, one of people who are Black.
Hint: Use the `filter` command

```
acs_WHT <- filter(acsdata, RACWHT==1)
acs_BLK <- filter(acsdata, RACBLK==1)
```

How would you write this using the %>% operator?

Grouping by poverty level

- For each of these datasets, let's group them by the federal poverty line.
- We're going to use the poverty guideline from the government[1] for households with 3 people, and assume the poverty line is $21,330.

[1] https://aspe.hhs.gov/topics/poverty-economic-mobility/poverty-guidelines/prior-hhs-poverty-guidelines-federal-register-references/2019-poverty-guidelines

You try!

Getting the counts

- For the counts, we want to count the number of observations in each group.
- `n()` counts the values within each group.

```
acs_BLK %>%
    group_by(HINCP <= 21330)%>%
    summarize(n=n())
```

Getting the proportions

- To get the proportions, we're going to introduce a new variable into our summarized table.
- We can add new columns into a table using `mutate`.

mutate()

`mutate()`

- Look up `?mutate`

`mutate()`

- Look up `?mutate`
- We can name and define new columns within `mutate`!

`mutate()`

- Look up `?mutate`
- We can name and define new columns within `mutate`!
- Here, we want to divide the new column named `n`, by the sum of this column, to find out the fraction of individuals living in poverty.

mutate()

- Look up ?mutate
- We can name and define new columns within mutate!
- Here, we want to divide the new column named n, by the sum of this column, to find out the fraction of individuals living in poverty.

```
acs_BLK %>%
    group_by(HINCP <= 21330) %>%
    summarize(n = n()) %>%
    mutate(frac = n/sum(n))
```

You try!

Redo the previous command for the dataset comprising of only white individuals (`acs_WHT`). What does this tell you?

More practice

Rather than computing the fraction of people above and below the poverty line, and how this varies by race, can you compute the fraction of the population that's below 25?

Where we are and where we're going

What we learned:

- Computing spread in different ways (IQR, variance, standard deviation)

Where we are and where we're going

What we learned:

- Computing spread in different ways (IQR, variance, standard deviation)
- Other operations in the `tidyverse`: `group_by`, `summarize`, `%>%`, ...

Where we are and where we're going

What we learned:

- Computing spread in different ways (IQR, variance, standard deviation)
- Other operations in the `tidyverse`: `group_by`, `summarize`, `%>%`, ...

What's next?

- Tomorrow, we'll learn how to embed our work (code, output and plots) into a nice PDF using `RMarkdown`

Where we are and where we're going

What we learned:

- Computing spread in different ways (IQR, variance, standard deviation)
- Other operations in the `tidyverse`: `group_by`, `summarize`, `%>%`, ...

What's next?

- Tomorrow, we'll learn how to embed our work (code, output and plots) into a nice PDF using `RMarkdown`
- We'll also talk about debugging your code and how to (hopefully!) quickly figure out what's going wrong.

Where we are and where we're going

What we learned:

- Computing spread in different ways (IQR, variance, standard deviation)
- Other operations in the `tidyverse`: `group_by`, `summarize`, %>%, ...

What's next?

- Tomorrow, we'll learn how to embed our work (code, output and plots) into a nice PDF using `RMarkdown`
- We'll also talk about debugging your code and how to (hopefully!) quickly figure out what's going wrong.

Homework 2 will be available on Canvas today. Due on Tuesday July 19th at 1:29 pm