Session 4: Tidyverse

Welcome!

TA: Jennifer Lin

MMSS 211: Institutions, Rules, & Models in Social Science

2022-04-20

Goals

1. Learn the basics of cleaning data using `dplyr`
2. Apply data cleaning skills to novel data
3. Generate summary statistics tables using `dplyr`

The Fundimentals of Data Wrangling

Data

Cooperative Elections Studies (CES) 2020 Study, which surveys respondents from every Congressional District every two years on various issues related to politics.

We are going to look at a handful of demographic variables to learn how to clean and summarize data.

```
CES20 <- read.csv("CES2020_subset.csv")
```

Things to Keep in Mind

1. Be familiar with the codes and structure of the original data
2. Pay attention to how the original data codes missing values
3. Know the kind of data structure you need for your analyses (Categorical? Numeric?)

Tidyverse Universe

Tidyverse

> The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures.

Packages

- `ggplot2`
- `dplyr`
- `stringr`
- `readr`
- `forcats`
- `purrr`
- `tidyr`
- `tibble`

The Pipe (%>%)

- The function that tells R to do one thing then something else and so on...
- Funnels each step into the next step without creating a new object

```
data %>%
  step 1 %>%
  step 2 %>%
  step 3 %>%
  steo 4
```

dplyr

From `dplyr` package documentation:

> `dplyr` is a grammar of data manipulation, providing a consistent set of verbs that help you solve the most common data manipulation challenges:
>
> - `mutate()` adds new variables that are functions of existing variables
> - `select()` picks variables based on their names.
> - `filter()` picks cases based on their values.
> - `summarise()` reduces multiple values down to a single summary.
> - `arrange()` changes the ordering of the rows.

Today, we will cover these functions (and then some)

`select()`: Picking Variables

Picking out variables from a data frame

```
select(.data, ...)

data %>%
  select(...)
```

We can select variables by name or by position

There are some shortcuts in the select variables

Operator	Description
`everything()`	everything in data
`last_col()`	last column
`starts_with()`	start with a string
`ends_with()`	ends with a string
`contains()`	contains a string

`select()`: Picking Variables

An Example

```
CES20_new <- CES20 %>%
  select(
    birthyr,
    gender,
    votereg,
    urbancity,
    pid3,
    ideo5,
    vote20
  )
```

`filter()`: Picking Cases

Putting Observations through a sifter where `var operator condition`

```
filter(.data, ...)

data %>%
  filter(...)
```

We can use conditions for variables where the variables are joined by AND (&) or OR (|) operators

Operator	Description
==	exactly equal to
<	less than
<=	less than or equal to
%in%	is in this list
!=	not equal to
is.na()	NA values

`filter()`: Picking Cases

An Example

```
CES20_new <- CES20_new %>%
  filter(!is.na(pid3))
```

`mutate()`: Creating Variables

Create new variables that are transformations of pre-existing ones.

```
data <- data %>%
  mutate(new = function(old))
```

where `function()` denotes functions for recoding like `case_when()`, `if_else()` or `recode()`. It can also be a function you wrote yourself!

So Speaking of Recoding:

```
data <- data %>%
  mutate(new = case_when(
    old == [old] ~ [new code],
    # For example:
    voterreg == "yes" ~ 1,
    # For all else not
    # previously defined
    TRUE ~ [new code]
  ))
```

`mutate()`: Creating Variables

An Example

```
CES20_new <- CES20_new %>%
  mutate(
    ideo3  = case_when(
      ideo5 %in% c(1:2) ~ "Liberal",
      ideo5  ==  3      ~ "Moderate",
      ideo5 %in% c(4:5) ~ "Conservative"),
    ideo3  = as.factor(ideo3),
    residence      = case_when(
      urbancity == 1 ~ "City",
      urbancity == 2 ~ "Suburb",
      urbancity == 3 ~ "Town",
      urbancity == 4 ~ "Rural"),
    residence = factor(
      residence,
      levels = c("City", "Suburb", "Town", "Rural"))
  )
```

`mutate()`: Creating Variables

An Example

```
CES20_new <- CES20_new %>%
  mutate(
    age     = 2020 - birthyr,
    AGE     = case_when(
      age %in% c(18:24) ~ "18 - 24",
      age %in% c(25:44) ~ "25 - 44",
      age %in% c(45:64) ~ "45 - 64",
      age %in% c(65:80) ~ "65+"),
    TRUMP   = case_when(
      vote20 == 1 ~ TRUE,
      TRUE ~ FALSE),
    pres_vote = case_when(
      vote20 == 1 ~ "Donald J. Trump",
      vote20 == 2 ~ "Joseph R. Biden"),
  )
```

`summarise()`: Get to the Point

Reduce observations to simple statistics.

Is usually preceded by `group_by()` which groups the data by a grouping variable

```
data %>%
  group_by(grouping_variable) %>%
  summarise(
    newvar_name1 = function(variable),
    newvar_name2 = function(variable),
    .groups = 'keep)
```

where `function()` is something like `mean()`, `sd()`, `min()`, `n()`, `max()` or any other summary statistics function

Add `.groups = 'keep'` to maintain the groups or `.groups = 'drop'` to remove all groupings

`summarise()`: Get to the Point

An Example

```
n_by_residence <- CES20_new %>%
  group_by(residence) %>%
  summarise(
    n = n(),
    avg_age = mean(age, na.rm = TRUE),
    .groups = 'keep'
  ) %>%
  filter(!is.na(residence))
```

residence	n	avg_age
City	16668	44.90659
Suburb	23482	49.26667
Town	8809	49.04949
Rural	11693	51.14479

`summarise()`: Get to the Point

An Example

```
vote_by_ideo <- CES20_new %>%
  group_by(ideo3, pres_vote) %>%
  summarise(
    n = n(),
    avg_age = mean(age, na.rm = TRUE),
    .groups = 'keep') %>%
  filter(!is.na(ideo3) & !is.na(pres_vote))
```

ideo3	pres_vote	n	avg_age
Conservative	Donald J. Trump	1772	59.31151
Conservative	Joseph R. Biden	364	52.93681
Liberal	Donald J. Trump	66	44.10606
Liberal	Joseph R. Biden	5430	48.16096
Moderate	Donald J. Trump	603	51.81758
Moderate	Joseph R. Biden	2923	54.25830

`arrange()`: Put in Order

Put variables in alphabetical or numeric order

```
arrange(
  data, variable,
  grouping_variable)

data %>%
  arrange(
    variable,
    grouping_variable)
```

You can also put values in reverse order using `desc()`

```
arrange(
  data,
  desc(variable),
  grouping_variable)

data %>%
  arrange(
    desc(variable),
    grouping_variable)
```

`arrange()`: Put in Order

An Example

```
n_by_residence <- n_by_residenc
  arrange(avg_age)
```

residence	n	avg_age
City	16668	44.90659
Town	8809	49.04949
Suburb	23482	49.26667
Rural	11693	51.14479

```
vote_by_ideo <- vote_by_ideo %>
  arrange(desc(avg_age))
```

ideo3	pres_vote	n	avg_age
Conservative	Donald J. Trump	1772	59.31151
Moderate	Joseph R. Biden	2923	54.25830
Conservative	Joseph R. Biden	364	52.93681
Moderate	Donald J. Trump	603	51.81758
Liberal	Joseph R. Biden	5430	48.16096
Liberal	Donald J. Trump	66	44.10606

Exercises

1. Using `mutate()` and `case_when()`, recode the `votereg` and `gender` variables such that:
    1. `gender` is coded as 1 is Male and 2 is Female and is a factor variable
    2. `votereg` is coded as 1 is yes and 2 is no
2. Pick any combination of 2 variables, either from the demonstration or your recodes in Part 1. Create a data frame that satisfies the following:
    1. The dataframe ONLY includes both of these variables
    2. There are no NA values on both of these variables
    3. The dataset is arranged in alphabetical order on one of these variables
3. From your outcome in Part 2, generate some sort of summary statistics table. Use reasonable grouping mechanisms.