R Handout Slides: Introduction to tidyverse

2023-08-24

Introducing `tidyverse`

`tidyverse` package gives us tools for data cleaning and processing

- "Base `R`" (and other packages) offers other tools
- We want to use these *specific* tools for Data Analysis Activities

Things to remember:

- First time you use it on a computer, run `install.packages()`
- Include `library(tidyverse)` at start of your `R` scripts

Overview

Slides and handout cover following functions:

- `select()` - pick out specific columns or variables from your data
- `filter()` - pick out specific rows or observations from your data
- `arrange()` - reorder or sort rows of your data
- `mutate()` - create new variables as functions of existing variables
- `summarize()` - creates summary statistics (with `group_by()`)

We'll also talk about piping via %>%.

Sample Data Set

```
person.ID <- c(12, 24, 54, 65)
address   <- c("123 Main St", "274 Long St",
               "789 Right St", "467 Left St")
employed  <- c(TRUE, TRUE, FALSE, TRUE)
wage.inc  <- c(12500, 15750, 0, 14100)

sample.data <- data.frame(person.ID, address,
                          employed, wage.inc)

sample.data
```

```
##   person.ID       address employed wage.inc
## 1        12   123 Main St     TRUE    12500
## 2        24   274 Long St     TRUE    15750
## 3        54 789 Right St    FALSE        0
## 4        65   467 Left St     TRUE    14100
```

Select Columns or Variables using `select()`

```
# Select person.ID and address from sample.data,
# and store as a new data frame named ID.address.data

ID.address.data <- select(sample.data, person.ID, address)

# Print new data frame

ID.address.data
```

```
##   person.ID       address
## 1        12   123 Main St
## 2        24   274 Long St
## 3        54 789 Right St
## 4        65   467 Left St
```

Select Rows or Observations using `filter()`

```
# Keep observation with address of "123 Main St"

filter(sample.data, address == "123 Main St")
```

```
##   person.ID      address employed wage.inc
## 1        12 123 Main St     TRUE    12500
```

```
# Keep observations with wages greater than 14,000

filter(sample.data, wage.inc > 14000)
```

```
##   person.ID      address employed wage.inc
## 1        24 274 Long St     TRUE    15750
## 2        65 467 Left St     TRUE    14100
```

Select Rows or Observations using `filter()` Con't

```
# Keep observations using two criteria using the
# "&" symbol - this syntax requires that BOTH
# conditions be true

filter(sample.data, employed == TRUE & person.ID < 50)
```

```
##   person.ID      address employed wage.inc
## 1        12 123 Main St     TRUE    12500
## 2        24 274 Long St     TRUE    15750
```

```
# Alternatively, return rows that match EITHER
# condition using the | symbol (read as "OR")

filter(sample.data, employed == TRUE | person.ID < 50)
```

```
##   person.ID      address employed wage.inc
## 1        12 123 Main St     TRUE    12500
## 2        24 274 Long St     TRUE    15750
## 3        65 467 Left St     TRUE    14100
```

Sorting Data using arrange()

```
# Sort sample.data observations by wage.inc
# from smallest to largest

arrange(sample.data, wage.inc)
```

```
##   person.ID      address employed wage.inc
## 1        54 789 Right St    FALSE        0
## 2        12  123 Main St     TRUE    12500
## 3        65  467 Left St     TRUE    14100
## 4        24  274 Long St     TRUE    15750
```

```
# You can sort by multiple variables, and
# reverse sort using desc() function
```

Summarizing Data with summarize()

```
# Use summarize to return a data frame with
# average wages.

summarize(sample.data, mean.wage.inc = mean(wage.inc))
```

```
##   mean.wage.inc
## 1       10587.5
```

```
# Handles multiple summary stats easily

summarize(sample.data,
          median.wage.inc = median(wage.inc),
          pct.10.wage.inc = quantile(wage.inc, 0.1),
          pct.90.wage.inc = quantile(wage.inc, 0.9))
```

```
##   median.wage.inc pct.10.wage.inc pct.90.wage.inc
## 1           13300            3750           15255
```

Creating New Variables using `mutate()`

```
# Silly formula just to show what mutate can do

mutate(sample.data,
       new.var = log(person.ID)^2 + wage.inc/50)
```

```
##   person.ID       address employed wage.inc   new.var
## 1        12   123 Main St     TRUE    12500 256.17476
## 2        24   274 Long St     TRUE    15750 325.10003
## 3        54 789 Right St    FALSE        0  15.91199
## 4        65   467 Left St     TRUE    14100 299.42551
```

Piping with %>%

```r
# Two ways of doing the same thing - first consider:

sample.data <- mutate(sample.data,
                      avg.wage.inc = mean(wage.inc))

# Then compare with code below, which does the same
# but uses %>% to make things clearer:

sample.data <- sample.data %>%
  mutate(avg.wage.inc = mean(wage.inc))
```

Piping with %>% Continued

```r
# Let's consider another example:

summary.stats.table <- sample.data %>%
  mutate(new.var = log(person.ID)^2 + wage.inc/50) %>%
  summarize(avg.new.var = mean(new.var))
```

Grouping Data with group_by

```r
# Calculating average wages grouped by employed
# status using group_by()

group.avg.wages <- sample.data %>%
  group_by(employed) %>%
  summarize(mean.wages = mean(wage.inc))
```

Grouping Data with group_by() Continued

```
# Create a new variable using mutate

sample.data <- sample.data %>%
  group_by(employed) %>%
  mutate(avg.wage = mean(wage.inc))
```