r_int_day_3_data_transformation

Nicholus Tint Zaw

2022-11-10

Content

This lecture focus on the implementation of data wrangling using the `dplyer` package and will cover some exercise using the popular functions from `dplyer` package.

1. filter
2. arrange
3. select
4. mutate
5. summarise
6. group_by

Some useful commands for exploring dataset

```
# load dataset

df <- polls_us_election_2016 # from dslabs package

head(df) # print first 6 obs
tail(df) # print last 6 obs

names(df) # variable/column names

unique(df$grade) # inspect unique values in a specific variable/column

typeof(df$grade) # inspect the type of variable
class(df$grade) # inspect the type of variable

table(df$grade) # frequency table
```

`filter` : select the sub-set of the dataset

```
filter(df, grade == "D")
filter(df, state == "Ohio")

# filter with multiple conditions
```

```r
interested_grades <- c("D", "B")

filter(df, grade %in% interested_grades) # one line

filter(df, grade == "D" | grade == "B")
```

Using `filter` with pipe (`%>%`) function.

```r
df %>%
  filter(state == "Illinois")
```

`arrange` : order the dataset by given variable

```r
head(arrange(df, samplesize)) # ascending order - default

head(arrange(df,desc(population))) # descending order


head(arrange(df, population))
head(arrange(df, desc(population)))
```

use with pipe function,

```r
df %>% arrange(population)

df %>% arrange(desc(population))
```

`select`: keep only variables require for data processing

```r
head(select(df, state, samplesize, population))


head(select(df, -(c(state, population, samplesize))))
```

```r
# starts_with("")

df %>%
  select(starts_with("adj"))

# ends_with("")
df %>%
  select(ends_with("date"))

# contains("")
df %>%
  select(contains("po"))

# matches("(.)\\1") - regular expression
```

```r
df %>%
  select(matches("raw")) # check the result.

# what is the different between matches and contains?

df1 <- data.frame(colnm = 1:5, col1 = 24, col2 = 46)
df1 %>%
    select(contains("col"))

df1 %>%
  select(matches("col\\d+"))

# num_range("x", 1:3)
# pls check at help file - type ?dplyr::select in console.
```

Wanna study more about regular expression, check here.

`mutate`: adding new column to existing one

```r
df %>%
  mutate(ss_small = ifelse(samplesize < 1000, 1, 0)) %>%
  select(samplesize, ss_small)

# if want to add into existing dataframe, override the existing dataframe with resulted dataframe.
df <- df %>%
  mutate(ss_small = ifelse(samplesize < 1000, 1, 0)) # check the variable names and numbers
```

`summarise`: perform sumstat functions

```r
summarise(df, ss_mean = mean(samplesize)) # what happened!

summarize(df, ss_mean = mean(samplesize, na.rm = TRUE)) # Missing values are contagious

# Missing values are contagious
NA + 1

10 * 100 * NA

100/NA


# sumstat table for samplesize var
df %>%
  summarise(man = mean(samplesize, na.rm = TRUE),
            sd = sd(samplesize, na.rm = TRUE),
            median = median(samplesize, na.rm = TRUE),
            min = min(samplesize, na.rm = TRUE),
            max = max(samplesize, na.rm = TRUE))
```

`group_by`: manipulation at group level, sub-group level

```
# get the sample size mean value per state
df %>%
  group_by(state) %>%
  summarise(mean = mean(samplesize, na.rm = TRUE))


df %>%
  group_by(population) %>%
  summarise(mean = mean(samplesize, na.rm = TRUE))


df %>%
  group_by(state) %>%
  count()
```

Exercise

- Use `iris` dataset and calculate sumstat by `Species`.
- Select the only observation from `versicolor` and `virginica` species.