Data Manipulation

Generated by Steve Cygu

November 23, 2021

Contents

List of Figures

List of Tables

1 Summary

We can create or use existing datasets, and perform various manipulations for various data types:

- numeric vectors
- factors

2 Data sets

A data frame is a table of observations. Each row contains one observation. Each observation must contain the same variables. These variables are called columns, and you can refer to them by name. You can also refer to the contents by row number and column number, just as with a matrix.

2.1 Creating datasets

- `data.frame`: Let us create two variables *age* and *gender* and combine them into a data set

```
age <- seq(50, 100, length.out=10)
gender <- c("M", "M", "F", "M", "F", "F", "M", "M", "F", "M")
age_gender_df <- data.frame(age, gender)
print(age_gender_df)
```

```
         age gender
1   50.00000      M
2   55.55556      M
3   61.11111      F
4   66.66667      M
5   72.22222      F
6   77.77778      F
7   83.33333      M
8   88.88889      M
9   94.44444      F
10 100.00000      M
```

We can change variable names in the `age_gender_df` created above:

- `names()`
- `colnames()`

```
# Create variable names
names(age_gender_df) <- c("age", "sex")
# colnames(age_gender_df) <- c("ages", "sex")
print(age_gender_df)
```

```
         age sex
1   50.00000   M
2   55.55556   M
3   61.11111   F
4   66.66667   M
5   72.22222   F
6   77.77778   F
7   83.33333   M
8   88.88889   M
9   94.44444   F
10 100.00000   M
```

Suppose we observe another variable (`edu_level`) indicating the education level of the respondent such that:

- 1 = No schooling
- 2 = Secondary
- 3 = College/University

we can created the edu_level variable with these categories and labels

- cbind.data.frame()

```
# Create factor/quantitative variable
edu_level <- c(1,1,1,2,3,3,2,4,1,1)
age_gender_edu_df <- cbind.data.frame(age_gender_df, edu_level)
age_gender_edu_df
```

```
         age sex edu_level
1   50.00000   M         1
2   55.55556   M         1
3   61.11111   F         1
4   66.66667   M         2
5   72.22222   F         3
6   77.77778   F         3
7   83.33333   M         2
8   88.88889   M         4
9   94.44444   F         1
10 100.00000   M         1
```

- Add the factor levels
  - factor()

```
age_gender_edu_df$edu_level <- factor(age_gender_edu_df$edu_level
    , levels=c(1,2,3)
    , labels=c("No schooling", "Secondary", "College/University")
)
age_gender_edu_df
```

```
         age sex          edu_level
1   50.00000   M       No schooling
2   55.55556   M       No schooling
3   61.11111   F       No schooling
4   66.66667   M          Secondary
5   72.22222   F College/University
6   77.77778   F College/University
7   83.33333   M          Secondary
8   88.88889   M               <NA>
9   94.44444   F       No schooling
10 100.00000   M       No schooling
```

2.2 Creating pipelines

This might come up later in the other chapters but it might make our life easier in handling dataframes and functions.

- We can use the pipe operator (%>%) to make workflow easier to read and write. Originally, the pipe operator %>% is from magrittr package but we are mainly going to use the tidyverse version from package dplyr, i.e., library(dplyr) in setup chunk.

```
## Print the first few observations
### Base R
head(age_gender_edu_df)
```

```
       age sex          edu_level
1 50.00000   M       No schooling
2 55.55556   M       No schooling
3 61.11111   F       No schooling
4 66.66667   M          Secondary
5 72.22222   F College/University
6 77.77778   F College/University
```

```
### Using pipe
age_gender_edu_df %>% head()
```

```
       age sex          edu_level
1 50.00000   M       No schooling
2 55.55556   M       No schooling
3 61.11111   F       No schooling
4 66.66667   M          Secondary
5 72.22222   F College/University
6 77.77778   F College/University
```

The pipe operator does not provide any new functionality to R, but it can greatly improve the readability of code. The pipe operator takes the output of the function or object on the left of the operator and passes it as the first argument of the function on the right.

- The difference doesn't seem much in this example but with complicated examples, we may start seeing the benefits. For example, in our previous example to create `edu_level`, we use `%>%`

```
age_gender_edu_df <- (age_gender_df
    %>% mutate(edu_level=edu_level
        , edu_level=factor(edu_level, levels=c(1,2,3)
            , labels=c("No schooling", "Secondary", "College/University")
        )
    )
)
age_gender_edu_df
```

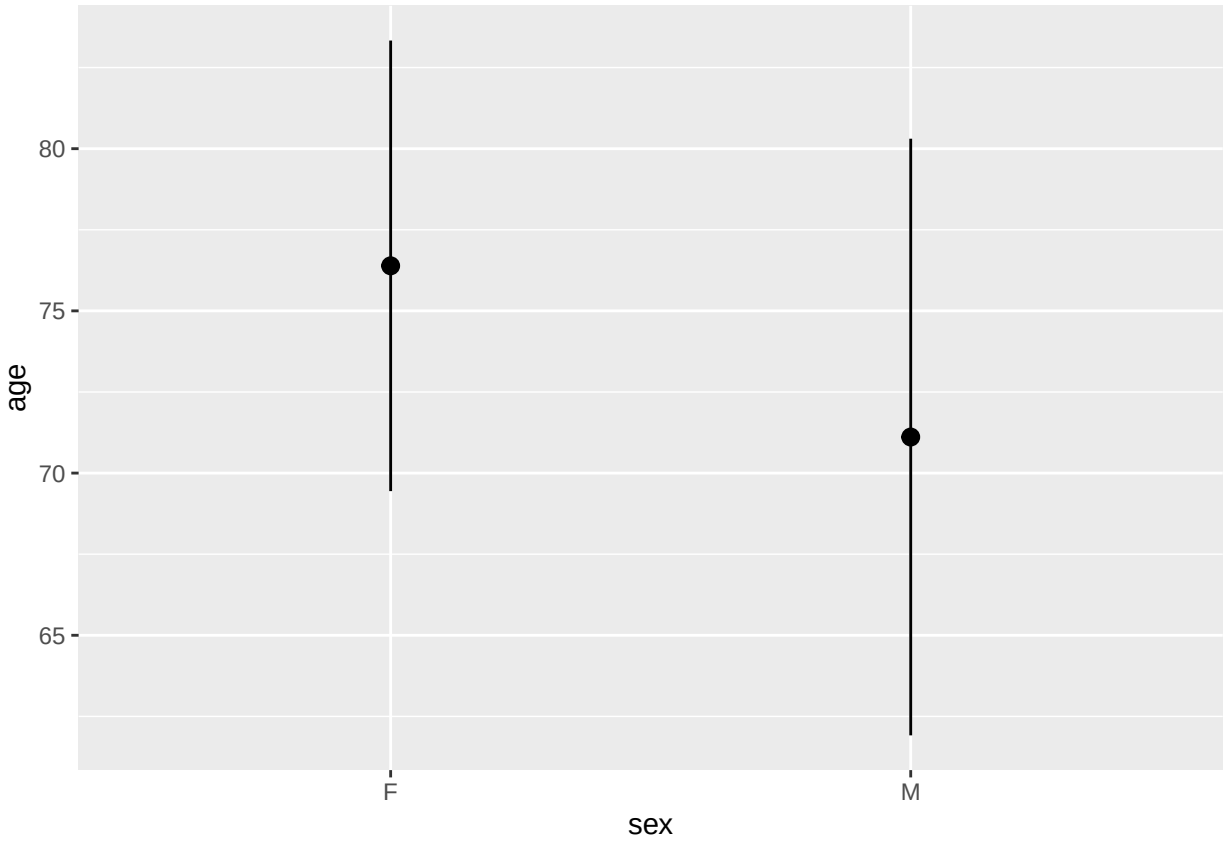
```
         age sex          edu_level
1   50.00000   M       No schooling
2   55.55556   M       No schooling
3   61.11111   F       No schooling
4   66.66667   M          Secondary
5   72.22222   F College/University
6   77.77778   F College/University
7   83.33333   M          Secondary
8   88.88889   M               <NA>
9   94.44444   F       No schooling
10 100.00000   M       No schooling
```

Now let us try to build a pipeline:

- drop observations with missind `edu_level`
- select `gender` and `age` columns
- generate a box plot of `age` and `gender` using `ggplot`

```
data_summary <- (age_gender_edu_df
    %>% filter(!is.na(edu_level))
    %>% select(age, sex)
    %>% ggplot(aes(x=sex, y=age)) + stat_summary() #+ theme_bw()
)
```

```
print(data_summary)
```



2.3 Reading data from other sources

R can read data created in various formats (SPSS, SAS, Stata, Excel, CSV, TXT, etc).