Data Manipulation

Generated by Steve Cygu

November 24, 2021

Contents

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 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()`

```
         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()`

```
         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 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.

```
       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
```

```
       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.

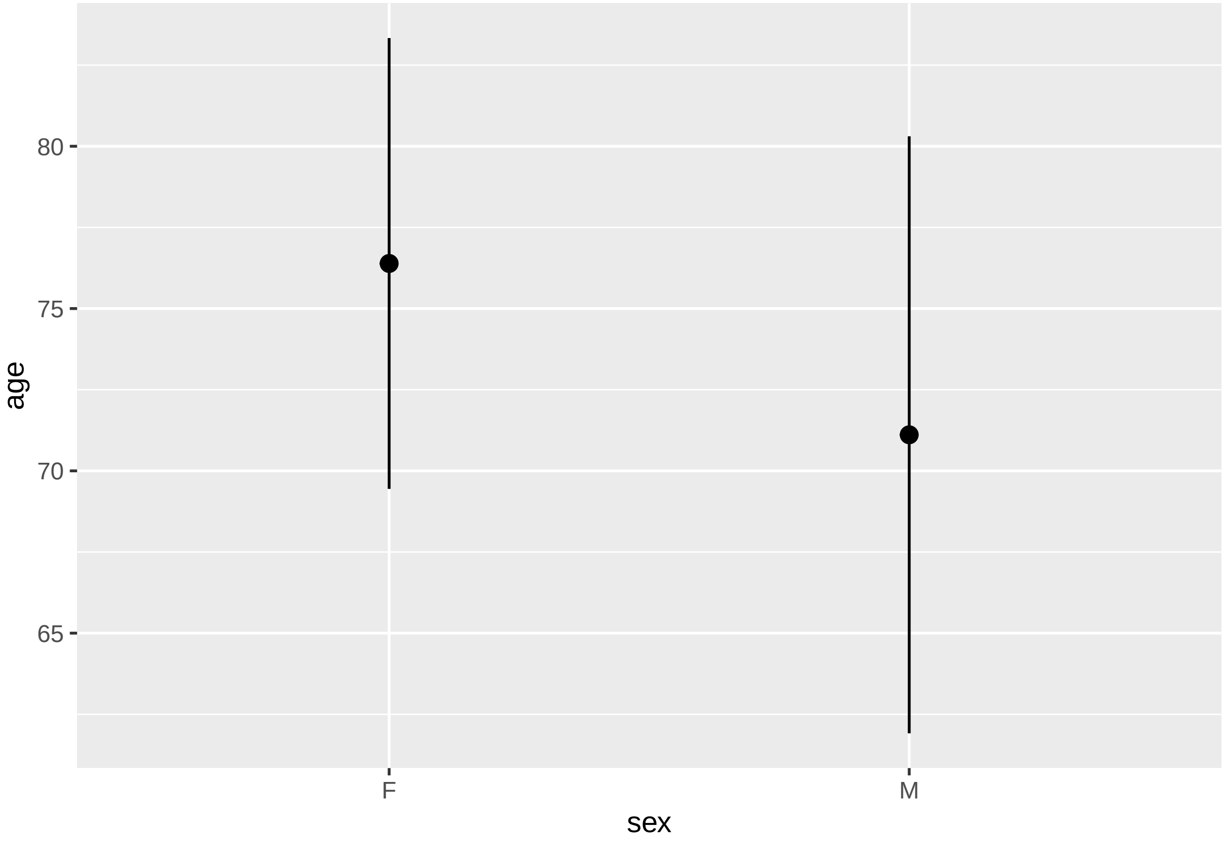
- The difference doesn't seem much in this example but with complicated examples, we may start seeing the benefits. For example, in our previous example to create `edu_level`, we use `%>%`

```
          age 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`



2.3 Reading data from other sources

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

We will use a number of dataset for illustration:

- **Contraceptive Method Choice**: This dataset is a subset of the 1987 National Indonesia Contraceptive Prevalence Survey. The samples are married women who were either not pregnant or do not know if they were at the time of interview. The problem is to predict the current contraceptive method choice (no use, long-term methods, or short-term methods) of a woman based on her demographic and socio-economic characteristics.

2.3.1 CSV and Tab-delimited files

```
'data.frame':   1473 obs. of  11 variables:
 $ X         : int  1 2 3 4 5 6 7 8 9 10 ...
 $ wife_age  : int  24 45 43 42 36 19 38 21 27 45 ...
 $ wife_edu  : int  2 1 2 3 3 4 2 3 2 1 ...
 $ hus_edu   : int  3 3 3 2 3 4 3 3 3 1 ...
 $ num_child : int  3 10 7 9 8 0 6 1 3 8 ...
 $ wife_rel  : int  1 1 1 1 1 1 1 1 1 1 ...
 $ wife_work : int  1 1 1 1 1 1 1 0 1 1 ...
 $ hus_occup : int  2 3 3 3 3 3 3 3 3 2 ...
 $ live_index: int  3 4 4 3 2 3 2 2 4 2 ...
 $ media_exp : int  0 0 0 0 0 0 0 0 0 1 ...
 $ con_method: int  1 1 1 1 1 1 1 1 1 1 ...
```

Table 1: Data description

X	Variable	Description	Type	Values
1	wife_age	Wife's age	numerical	
2	wife_edu	Wife's education	categorical	1=low, 2, 3, 4=high
3	hus_edu	Husband's education	categorical	1=low, 2, 3, 4=high
4	num_child	Number of children ever born	categorical	0, 1-2, 3-4, 5+
5	wife_rel	Wife's religion	binary	0=Non-Islam, 1=Islam
6	wife_work	Wife's now working?	binary	0=Yes, 1=No
7	hus_occup	Husband's occupation	categorical	1, 2, 3, 4
8	live_index	Standard-of-living index	categorical	1=low, 2, 3, 4=high
9	media_exp	Media exposure	binary	0=Good, 1=Not good
10	con_method	Contraceptive method used	class attribute	1=No-use , 2=Long-term, 3=Short-term