DETAILED GUIDE

TO DATA MERGING IN R

PRODUCED BY: RISHI DEY CHOWDHURY

Table of Contents

1 Data Merging in R **2**

1.1 Why do we need to merge data? 2

1.2 Discussion with an Example 2

1.3 R Instructions 3

1.3.1 Basic R 3

1.3.2 Advanced R 4

1.3.3 Even Advanced R 6

About This File

This file was created for the benefit of all teachers and students wanting to use R for data wrangling purposes.

SECTION 1

Data Merging in R

Statistical analysis of the data involves various steps in which data wrangling is an important part. In this phase of the data science workflow, we prepare the data in the proper format so that it can facilitate easier analysis in the later steps of the workflow. One such data preparatory steps is data merging.

1.1 Why do we need to merge data?

In several scenarios, where data is obtained from various sources on same or somewhat similar covariates and factors, we find the need to merge the two datasets together in an appropriate manner. This can be achieved by following the steps of data merging, which we will demonstrate in R. Now, the question may arise, why not simply copy and paste one dataset at the end of the other. This might work in some cases but in most cases it fails to properly merge by matching column names, etc.

1.2 Discussion with an Example

Suppose we have two datasets, where in each dataset is from two different labs doing an experiment on finding out the elastic nature of the string from the data collected on length and weight attached to the string. Look at Table 3 for the two datasets. There is absolutely no need that the covariates measured needs to be same for both labs or the number of observations being same for both the labs. But upon merging, we will have no way to identify that from which lab the data comes from and hence, we consider adding an extra column refering to the lab from which each observation has come. We use nominal categorical variable e.g. 1 and 2 to do so. See the merged dataset in Table 4.

Table 1: Data from Lab 1

weight	length
1.0	5.29
1.5	6.31
2.0	7.28
2.5	8.33
3.0	9.30
3.5	10.32

Table 2: Data from Lab 2

weight	length
1.2	7.60
1.5	8.11
1.8	8.88
2.1	9.40
2.1	9.39

Table 3: Datasets from two sources

weight	length	lab
1.0	5.29	1
1.5	6.31	1
2.0	7.28	1
2.5	8.33	1
3.0	9.30	1
3.5	10.32	1
1.2	7.60	2
1.5	8.11	2
1.8	8.88	2
2.1	9.40	2
2.1	9.39	2

Table 4: Merged Dataset

1.3 R Instructions

1.3.1 Basic R

Using the basic R always works but doesn't provide some better functionality that comes with some other advanced R packages.

```
# Importing the .csv files corresponding to the datasets
(lab1 <- read.csv("spring1.csv"))
  weight length
1    1.0   5.29
2    1.5   6.31
3    2.0   7.28
4    2.5   8.33
5    3.0   9.30
6    3.5  10.32
(lab2 <- read.csv("spring2.csv"))
  weight length
1    1.2   7.60
```

```
2    1.5   8.11
3    1.8   8.88
4    2.1   9.40
5    2.1   9.39
# Including a column for the lab id (i.e. 1 or 2)
(lab1m <- data.frame(lab1, lab=1))
  weight length lab
1    1.0   5.29   1
2    1.5   6.31   1
3    2.0   7.28   1
4    2.5   8.33   1
5    3.0   9.30   1
6    3.5  10.32   1
(lab2m <- data.frame(lab2, lab=2))
  weight length lab
1    1.2   7.60   2
2    1.5   8.11   2
3    1.8   8.88   2
4    2.1   9.40   2
5    2.1   9.39   2
# Merging two datasets into one (i.e. rowbinding)
alllab <- rbind(lab1m, lab2m)
    weight length lab
1      1.0   5.29   1
2      1.5   6.31   1
3      2.0   7.28   1
4      2.5   8.33   1
5      3.0   9.30   1
6      3.5  10.32   1
7      1.2   7.60   2
8      1.5   8.11   2
9      1.8   8.88   2
10     2.1   9.40   2
11     2.1   9.39   2
```

1.3.2 Advanced R

Using the following R package helps us deal with issues like if we had different number of variables measured by lab 2 compared to lab 1 or had different variable/column names. It fills the cells with NA wherever no value is available.

```
# Loading package
library(dplyr)
library(readr)
```

```
# Importing the .csv files corresponding to the datasets
(lab1 <- read_csv("lab1data.csv"))
Rows: 6 Columns: 2
Column specification
Delimiter: ","
dbl (2): weight, length

Use ‘spec()‘ to retrieve the full column specification for this data.
Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
# A tibble: 6 × 2
  weight length
   <dbl>  <dbl>
1    1     5.29
2    1.5   6.31
3    2     7.28
4    2.5   8.33
5    3     9.3
6    3.5  10.3
(lab2 <- read_csv("lab2data.csv"))
Rows: 5 Columns: 2
Column specification
Delimiter: ","
dbl (2): weight, length

Use ‘spec()‘ to retrieve the full column specification for this data.
Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
# A tibble: 5 × 2
  weight length
   <dbl>  <dbl>
1    1.2   7.6
2    1.5   8.11
3    1.8   8.88
4    2.1   9.4
5    2.1   9.39
# Including a column for the lab id (i.e. 1 or 2)
(lab1m <- lab1 %>% mutate(lab=1))
# A tibble: 6 × 3
  weight length   lab
   <dbl>  <dbl> <dbl>
1    1     5.29     1
2    1.5   6.31     1
3    2     7.28     1
```

```
4    2.5   8.33     1
5    3     9.3      1
6    3.5  10.3      1
(lab2m <- lab2 %>% mutate(lab=2))
# A tibble: 5 × 3
  weight length   lab
   <dbl>  <dbl> <dbl>
1    1.2   7.6      2
2    1.5   8.11     2
3    1.8   8.88     2
4    2.1   9.4      2
5    2.1   9.39     2
# Merging two datasets into one (i.e. rowbinding)
(alllab <- bind_rows(lab1m, lab2m))
# A tibble: 11 × 3
   weight length   lab
    <dbl>  <dbl> <dbl>
 1    1     5.29     1
 2    1.5   6.31     1
 3    2     7.28     1
 4    2.5   8.33     1
 5    3     9.3      1
 6    3.5  10.3      1
 7    1.2   7.6      2
 8    1.5   8.11     2
 9    1.8   8.88     2
10    2.1   9.4      2
11    2.1   9.39     2
```

1.3.3 Even Advanced R

The above operation in Advanced R actually performs a full join of the two datasets. We can instead have different kind of joins of the two datasets depending on the needs. These joins can be performed based on which columns we want to join along. There are 4 types of join we can use, which are:

- inner join: includes all rows in first and second dataset.
- left join: includes all rows in first dataset.
- right join: includes all rows in second dataset.
- full join: includes all rows in first or second dataset.

```
# Loading package
library(dplyr)
```

```
library(readr)

# Importing the .csv files corresponding to the datasets
(lab1 <- read_csv("lab1data.csv"))
Rows: 6 Columns: 2
Column specification
Delimiter: ","
dbl (2): weight, length

Use ‘spec()‘ to retrieve the full column specification for this data.
Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
# A tibble: 6 × 2
  weight length
   <dbl>  <dbl>
1    1     5.29
2    1.5   6.31
3    2     7.28
4    2.5   8.33
5    3     9.3
6    3.5  10.3
(lab2 <- read_csv("lab2data.csv"))
Rows: 5 Columns: 2
Column specification
Delimiter: ","
dbl (2): weight, length

Use ‘spec()‘ to retrieve the full column specification for this data.
Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
# A tibble: 5 × 2
  weight length
   <dbl>  <dbl>
1    1.2   7.6
2    1.5   8.11
3    1.8   8.88
4    2.1   9.4
5    2.1   9.39
# Including a column for the lab id (i.e. 1 or 2)
(lab1m <- lab1 %>% mutate(lab=1))
# A tibble: 6 × 3
  weight length   lab
   <dbl>  <dbl> <dbl>
1    1     5.29     1
2    1.5   6.31     1
```

```
3    2     7.28     1
4    2.5   8.33     1
5    3     9.3      1
6    3.5  10.3      1
(lab2m <- lab2 %>% mutate(lab=2))
# A tibble: 5 × 3
  weight length   lab
   <dbl>  <dbl> <dbl>
1    1.2   7.6      2
2    1.5   8.11     2
3    1.8   8.88     2
4    2.1   9.4      2
5    2.1   9.39     2

# Full Join, using all the common column names across the two dataset
(alllab <- full_join(lab1m, lab2m))
Joining, by = c("weight", "length", "lab")
# A tibble: 11 × 3
   weight length   lab
    <dbl>  <dbl> <dbl>
 1    1     5.29     1
 2    1.5   6.31     1
 3    2     7.28     1
 4    2.5   8.33     1
 5    3     9.3      1
 6    3.5  10.3      1
 7    1.2   7.6      2
 8    1.5   8.11     2
 9    1.8   8.88     2
10    2.1   9.4      2
11    2.1   9.39     2
```