Combining Datasets

Irfan Kanat

July 4, 2017

Sometimes the variables we want are spread between two different datasets, or observations are coming from different files. How can we get all the data together? That is what we will learn in this activity.

Introducing Data

We will use three datasets in this learning activity.

1- Codes

ISO Country Code data

```
codes <- read.csv("data/codes.csv")
```

The dataset has 3 variables Name of the Country, ISO2 Code, and ISO3 Code.

I will change the column names to ensure uniformity (and to demonstrate how its done).

```
colnames(codes) <- c("name", "iso2", "iso3")
```

2 - Users

UN Data on number of internet users per 100 people. Data comes in 2 files, one for 2006 and one for 2007

```
users2006 <- read.csv("data/Users2006.csv")
users2007 <- read.csv("data/Users2007.csv")
```

Data has 3 variables, ISO3, Year, Users.

3 - Internet Servers

Worldbank data on sercure internet servers per 1 million people

```
servers <- read.csv("data/Servers.csv")
```

Data has 3 variables, iso3, Year, Servers.

Combine Rows

When the datasets have the same structure (variables), you can just append rows of one to the next.

Consider our user data. It is currently split into two datasets, one for 2006 and one for 2007. Would it not be nice to have it in a single file?

Let us check the structure of datasets.

```
str(users2006)
```

```
## 'data.frame':    217 obs. of  3 variables:
##  $ ISO3 : Factor w/ 217 levels "ABW","ADO","AFG",..: 3 5 55 9 2 4 10 7 8 1 ...
##  $ Year : int  2006 2006 2006 2006 2006 2006 2006 2006 2006 2006 ...
##  $ Users: num  2.11 9.61 7.38 NA 48.94 ...
```

```
str(users2007)
```

```
## 'data.frame':    217 obs. of  3 variables:
##  $ ISO3 : Factor w/ 217 levels "ABW","ADO","AFG",..: 3 5 55 9 2 4 10 7 8 1 ...
##  $ Year : int  2007 2007 2007 2007 2007 2007 2007 2007 2007 2007 ...
##  $ Users: num  1.9 15.04 9.45 NA 70.87 ...
```

Seems like the variables are the same, we can simply add the datasets end to end. I will use rbind() function to bind rows together.

```
usersLong <- rbind(users2006, users2007)
View(usersLong)
```

```
## Warning in View(usersLong): X cannot set locale modifiers
```

You will note the 2007 data just got added to the end of 2006 data. This format (each pairing of group, time in separate rows) is called a long format and is used very commonly in longitudinal and time series analysis.

Combine Columns

If the data is structured accordingly, you can simply combine two datasets by bringing relevant columns together.

```
head(users2006)
```

```
##   ISO3 Year     Users
## 1  AFG 2006  2.107124
## 2  ALB 2006  9.609991
## 3  DZA 2006  7.375985
## 4  ASM 2006        NA
## 5  ADO 2006 48.936847
## 6  AGO 2006  1.907648
```

```
head(users2007)
```

```
##   ISO3 Year     Users
## 1  AFG 2007  1.900000
## 2  ALB 2007 15.036115
## 3  DZA 2007  9.451191
## 4  ASM 2007        NA
## 5  ADO 2007 70.870000
## 6  AGO 2007  3.200000
```

Same countries occur in same rows. We can create a wide dataset where each country gets a row and time observations are recorded in different variables.

Let us make some tweaks so the data will sit nicely together.

```
# Store the base dataset in a new data.frame
usersWide <- users2006
# Tweak the column names a bit
colnames(usersWide) <- c("iso3", "year", "users2006")
# Now append the relevant columns from second dataset
usersWide$users2007 <- users2007$Users
# We don't need year variable anymore since it is in column names.
usersWide <- usersWide[, -2]
# Let us look at what we created
head(usersWide)
```

```
##   iso3 users2006 users2007
```

```
## 1  AFG  2.107124  1.900000
## 2  ALB  9.609991 15.036115
## 3  DZA  7.375985  9.451191
## 4  ASM        NA        NA
## 5  ADO 48.936847 70.870000
## 6  AGO  1.907648  3.200000
```

This is called a wide format dataset. While not as common, this format can sometimes be used in longitudinal analysis as well.

Combine Based on Groupings

We have the user data in usersLong data.frame, but the name of the country is not immediately apparent in this dataset. If we could match the usersLong and country codes, we could easily tell which country was which.

To combine datasets based on groups, we use the merge() function in R. Merge function is similar to a join function in SQL. If you are familiar with SQL, this will be easy to follow.

Let us review the merge() manual pages.

```
?merge
```

Merge takes a number of parameters:

1. x: first dataset*
2. y: second dataset*
3. by, by.x, by.y: Which columns to use in merging.
4. all, all.x, all.y: what to do with rows that are not matched (Left Join, Right Join, Inner Join...).

- only the parameters marked with '*' are mandatory. If other parameters are left blank, R will try to find common column names and merge based on common columns.

Let us merge codes and usersLong.

```
users <- merge(codes, usersLong, by.x = "iso3", by.y = "ISO3")
head(users)
```

```
##   iso3        name iso2 Year     Users
## 1  ABW       Aruba   AW 2006 28.000000
## 2  ABW       Aruba   AW 2007 30.900000
## 3  AFG Afghanistan   AF 2007  1.900000
## 4  AFG Afghanistan   AF 2006  2.107124
## 5  AGO      Angola   AO 2006  1.907648
## 6  AGO      Angola   AO 2007  3.200000
```

Since the two datasets did not have matching column names, I had to specify columns with by.x, and by.y parameters.

Let us review the function call:

```
users <- merge(codes, usersLong, by.x = "iso3", by.y = "ISO3")
```

1. users<-: I assign whatever is to the right of assignment operator (<-) to users
2. merge(): I call merge function with four parameters

- codes: first dataset (x)
- usersLong: second dataset (y)
- by.x: column name in first dataset to use in matching.
- by.y: column name in second dataset to use in matching.

You will notice that users has 402 observations whereas the source dataset usersLong had 434 observations. What is the source of difference?

Some rows could not be matched and they were dropped.

If you want to keep all observations from usersLong, you can use the all.y.

```
users <- merge(codes, usersLong, by.x = "iso3", by.y = "ISO3", all.y = TRUE)
nrow(users)
```

```
## [1] 434
```

There, much better.

Combine Based on Multiple Groupings

What if we have more than one variable we want to match?

Let us say we want to combine users and servers to get at an IT infrastructure dataset. There are two levels to data, countries and years... We need to match at two levels of groupings.

Let us see how it is done.

```
itInfrastructure <- merge(users, servers, by = c("iso3", "Year"))
head(itInfrastructure)
```

```
##   iso3 Year        name iso2     Users      Servers
## 1  ABW 2006       Aruba   AW 28.000000 357.03659625
## 2  ABW 2007       Aruba   AW 30.900000 385.30696121
## 3  ADO 2006        <NA> <NA> 48.936847 419.80017512
## 4  ADO 2007        <NA> <NA> 70.870000 435.91979076
## 5  AFG 2006 Afghanistan   AF  2.107124   0.07941672
## 6  AFG 2007 Afghanistan   AF  1.900000   0.23186126
```

Let us break down the command:

```
itInfrastructure<-merge(users, servers, by=c('iso3', 'Year'))
```

1. itInfrastructure<-: Assign results into itInfrastructure data.frame
2. merge(): I called merge function with 3 parameters:

- users: first dataset (x)
- servers: second dataset (y)
- by=c('iso3', 'Year'): since datasets had shared names for these variables I used a single by command and specified two variables.

```
# I will export this dataset so we can use it in later exercises.
write.csv(itInfrastructure, "data/itInfrastructure.csv", row.names = F)
```