**ECO 274 LAB: Append, Merge, and Collapsing Data Frames**

## Learning Objectives

- How to Append Data Frames in R
- Merging data frames
- Collapsing data frames

## ## Appending data frame

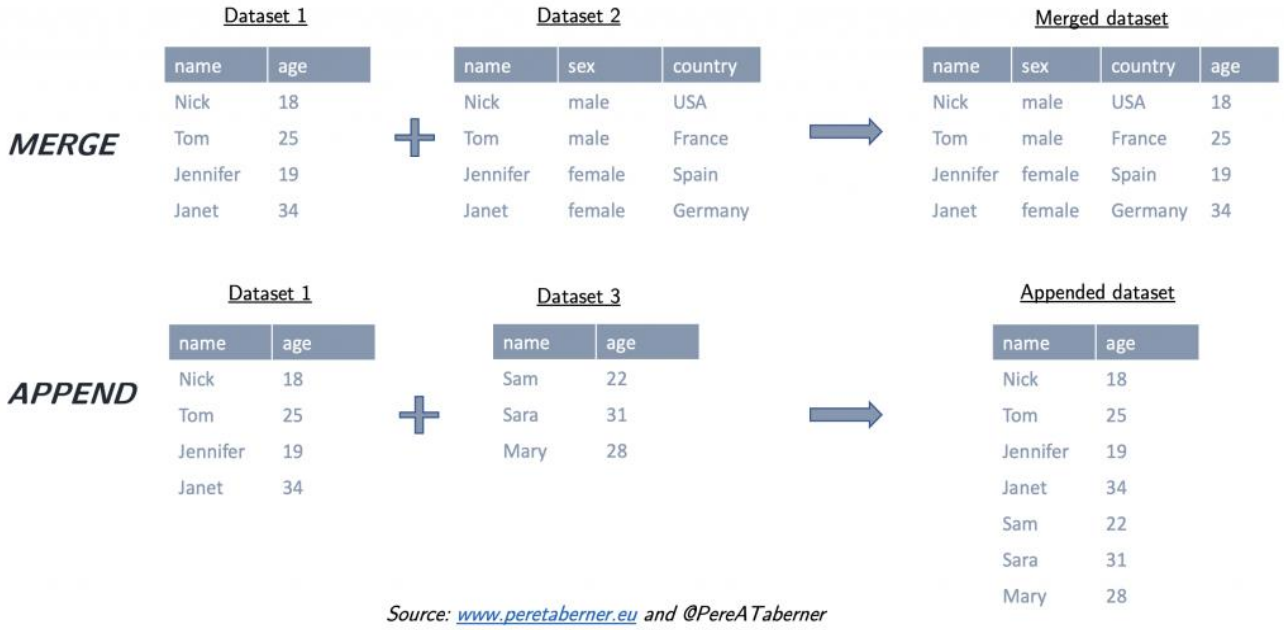
When it comes to appending data frames, the **rbind()** and **cbind()** function comes to mind because they can concatenate the data frames horizontally and vertically. In this tutorial, we will see how to use the **rbind()** function to append data frames.

```
#we will be working on cars data set

data("cars")

df1 <- head(cars,15)
df2 <- tail(cars,15)

cat("First Data Frame: ", "\n")
df1
cat("Second Data Frame: ", "\n")
df2

#To append the df2 data frame to df1, use the rbind() function.

appendedDf <- rbind(df1, df2)
cat("The appended data frame", "\n")
appendedDf



#Appending a Column to data frame


data("cars")

df <- head(cars, 10)

cat("The Data Frame: ", "\n")
df

mileage <- c(70, 80, 90, 100, 110, 120,134,137,156,160)
cat("After Appendung a new column Data Frame: ", "\n")

df$mileage <- mileage
df
```

```
#Use nrow() to append a row

#This method uses the nrow() function to append a row to the end of a given
data frame.

#define first data frame
df1 <- data.frame(var1=c(4, 13, 7, 8),
                  var2=c(15, 9, 9, 13),
                  var3=c(12, 12, 7, 5))
df1

#append row to end of data frame
df1[nrow(df1) + 1,] = c(5, 5, 3)
df1




#Using add_row() of tidyverse
#What if, instead, we want to add a new row not to the end of the DataFrame
but at some specific index of it?

#let’s create a simple DataFrame as an experiment:

super_talkers_guys <- data.frame(guys=c('funboy', 'smartGuy'),
                                 residence=c('firstHall', 'secondHall'),
                                 daily_talk_hours=c(21, 18),
                                 stringsAsFactors=FALSE)
print(super_talkers_guys)
super_talkers <- super_talkers_guys
super_talkers <- rbind(super_talkers, list('fairyLady', 'westHall', 17))
print(super_talkers)
super_talkers[nrow(super_talkers) + 1,] <- list('FoodpandaBoy', 'Surma', 16)
print(super_talkers)
#Using add_row() of tidyverse
library(tidyverse)
super_talkers <- super_talkers %>% add_row(guys='DuffGirl',
                                           residence='southHall',
                                           daily_talk_hours=20,
                                           .before=2)

print(super_talkers)
```

## Merging data frame

#how to merge datasets with R




## Merge functions

In this part, we will look at useful functions in ***dplyr*** to merge datasets and one useful function to append datasets. First, we briefly examine the six merging functions (each function joins two indicated datasets):

inner_join(): the output dataset only contains ***matched observations*** from both initial datasets.

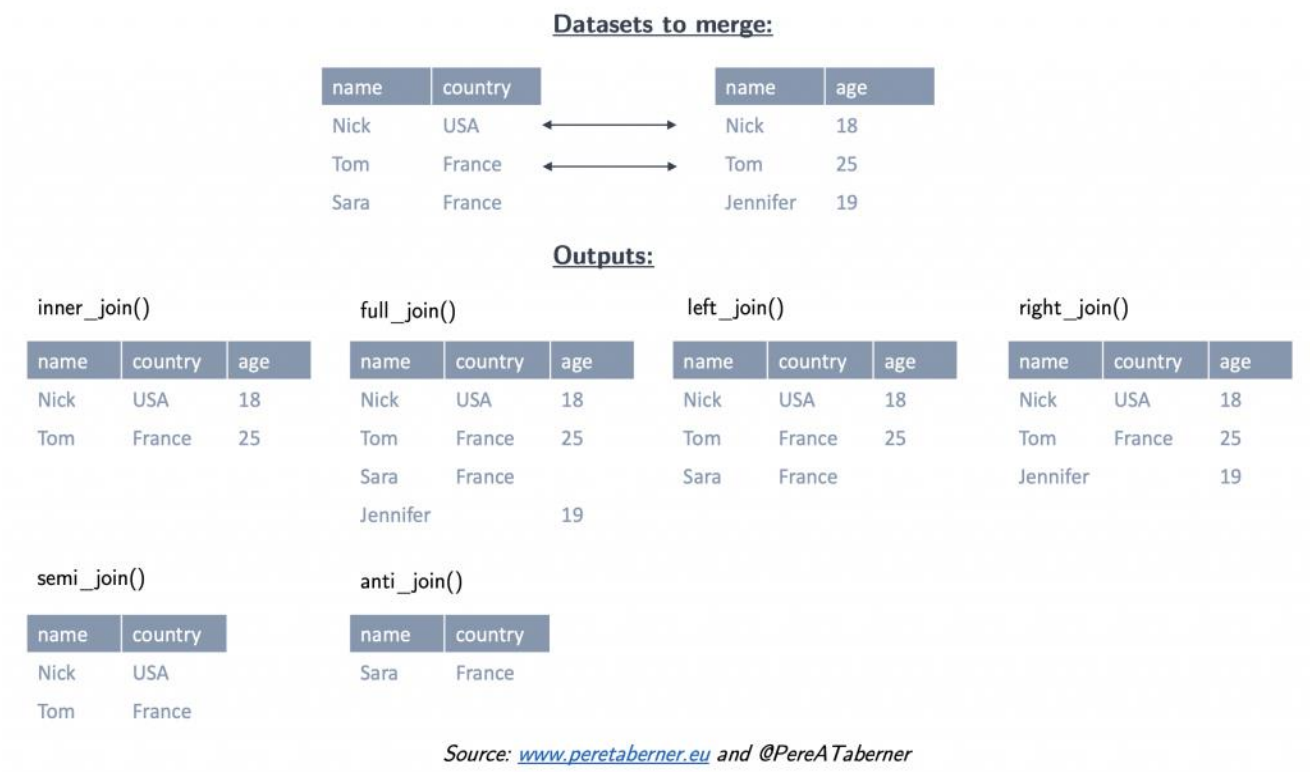
full_join(): the output dataset contains ***all observations*** from both initial datasets.

left_join(): the output dataset contains all observations from the first (or left) dataset indicated and only matched observations from the second (or right) dataset indicated.

right_join(): the output dataset contains ***only matched observations from the first*** (or left) dataset indicated and ***all observations from the second*** (or right) dataset indicated.

semi_join(): the output dataset only contains ***matched observations from the first*** (or left) dataset indicated (this includes only the variables from the left dataset).

anti_join(): the output dataset only contains ***NOT matched observations*** from the first (or left) dataset indicated (and also only the variables from the left dataset).



**Datasets to merge:**

| name | country |
|---|---|
| Nick | USA |
| Tom | France |
| Sara | France |

| name | age |
|---|---|
| Nick | 18 |
| Tom | 25 |
| Jennifer | 19 |

**Outputs:**

inner_join()

| name | country | age |
|---|---|---|
| Nick | USA | 18 |
| Tom | France | 25 |

full_join()

| name | country | age |
|---|---|---|
| Nick | USA | 18 |
| Tom | France | 25 |
| Sara | France | |
| Jennifer | | 19 |

left_join()

| name | country | age |
|---|---|---|
| Nick | USA | 18 |
| Tom | France | 25 |
| Sara | France | |

right_join()

| name | country | age |
|---|---|---|
| Nick | USA | 18 |
| Tom | France | 25 |
| Jennifer | | 19 |

semi_join()

| name | country |
|---|---|
| Nick | USA |
| Tom | France |

anti_join()

| name | country |
|---|---|
| Sara | France |

*Source: www.peretaberner.eu and @PereATaberner*

```
#merging data frames

std_id <- c(1:5)
names <- c("Paul", "Sara", "John", "Julia", "Laura")
years <- c("24", "30", "45", "29", "18")

stdlist1 <- data.frame(std_id, names)
stdlist2 <- data.frame(std_id, years)

#merge student list 1 and 2 datasets with inner_join():

# Merge the two datasets by "id" variable
output_dataset <- inner_join(stdlist1, stdlist2, by = "std_id")

stdlist1
stdlist2

output_dataset

#the two initial datasets might have two different variables but with the
same name.
#we can label each of these variable to know where it comes from with suffix
=:

# New variable in each dataset with the same name but different values to
show this example:
```

```
random1 <- c("a", "2", "c", "8", "a")
random2 <- c("d", "23", "c", "8a", "a")
stdlist1 <- data.frame(std_id, names, random1)
stdlist1
stdlist1 <- rename(stdlist1, random=random1)
stdlist1

# Prepare `std_list2` dataset for this example
stdlist2 <- data.frame(std_id, years, random2)
stdlist2
stdlist2 <- rename(stdlist2, random=random2)
stdlist2

# Then, merge these two datasets
output_dataset <- inner_join(stdlist1, stdlist2, by = "std_id",
suffix=c("_stdlist1", "_stdlist2"))
output_dataset

#Note that in case of wanting to join more than two datasets,
#we must repeat the function as many times as datatsets we want to join. For
example, to merge three datasets:


# Create a third dataset:
state <- c("Illinois", "Florida", "Nevada", "NewYork", "Georgia")
state <- data.frame(std_id, state)
state
output_dataset <- stdlist1 %>%
  inner_join(stdlist2, by = "std_id") %>%
  inner_join(state, by = "std_id")

output_dataset

## For all other merge functions, I strongly suggest you see the following


#https://statisticsglobe.com/r-dplyr-join-inner-left-right-full-semi-anti
```

## Collapsing Data

In this part, we will see how to collapse data in R. Collapsing means using one or several grouping variables to find summary statistics — mean, median, etc. — for different categories in your data. For example, if you have yearly income data for the 50 U.S. states over a 10-year period (i.e., you have 500 data points), you may want to know what the mean income was in each state (collapsing the data to 50 data points) or in each year (10 data points). Or you may want to collapse the data by year and U.S. region, say, South v. non-South (20 data points).

```
## Collapsing data set

grades <- data.frame(
  student = c("Julia", "Marry", "Cindy", "Ian", "Ella", "Frank", "Gina",
"Henry"),
  school = c(rep("Stanford", 4), rep("Harvard", 4)),
  sat_score = c(750, 730, 690, 800, 780, 720, 730, 700)
)

print(grades)

grades %>%
  group_by(school) %>%
  summarize(mean(sat_score))

grades %>%
  group_by(school) %>%
  summarize(mean_SAT = mean(sat_score))

#Several grouping variables
#Collapsing can also be done using several grouping variables.
#Let's modify the grades data frame to illustrate:

grades <- data.frame(
  student = c("Julia", "Marry", "Cindy", "Ian", "Ella", "Frank", "Gina",
"Henry"),
  school = c(rep("Stanford", 4), rep("Harvard", 4)),
  classof = rep(c(2017, 2017, 2018, 2018), 2),
  sat_score = c(750, 730, 690, 800, 780, 720, 730, 700)
)


#We now have two grouping variables: school and classof. The latter specifies
the expected graduation year for each student.

#Collapsing by these two grouping variables follows the same logic as above.
#Just specify the variables to collapse by inside group_by().

grades %>%
  group_by(school, classof) %>%
  summarize(mean_sat = mean(sat_score))


#One nice thing about using dplyr functions for collapsing data is that you
can combine them with other data manipulation functions
```

```
#an example in which we are filtering the grades data frame to class of 2017
and then collapsing:

grades %>%
  filter(classof == 2017) %>%
  group_by(school) %>%
  summarize(mean_sat = mean(sat_score))



#Here's an example that adds a variable after the collapse
#(rescaling the mean SAT scores to be between 0 and 100, assuming 800 is the
maximum possible score):

grades %>%
  group_by(school) %>%
  summarize(mean_sat = mean(sat_score)) %>%
  mutate(mean_sat_stdrzd = (mean_sat / 800) * 100)



#Importing quiz data set from my Github page:

library (readr)

urlfile="https://raw.githubusercontent.com/masud-alam/ECO274LAB/main/world-
small.csv"

world_small_data<-read_csv(url(urlfile))
View(world_small_data)
```

**Acknowledgement**

1. Simon Ejdemyr (2015), https://sejdemyr.github.io/r-tutorials

2. https://statisticsglobe.com

3. Harvard Chan Bioinformatics Core, R for data science lab