Data Scientifique

GradProSkills

Data Management in R

Session 2: Merging and reshaping data

Instructor: Yara Abu Awad

Data Scientifique

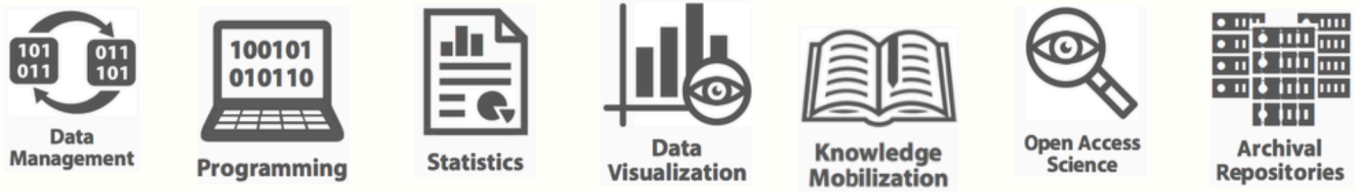
HOME

Data Scientifique fosters data science initiatives among researchers

Our mission is to help researchers develop and strengthen their analytical, computational, & programming techniques to facilitate health knowledge mobilization.

Data Scientifique offers opportunities for researchers to leverage advances in data management/integration, quantitative statistical methods, and data visualization through Educational Workshops, Enrichment Awards, and Data Clinic. Data Scientifique is a professional development catalyst for faculty, post-docs, and graduate students. Resources include tools to optimize research collaboration, archival data repositories, and open-science initiatives.

In September 2015, Data Scientifique was founded by Jennifer J McGrath, PhD MPH (PERFORM Chair, Childhood Preventive Health & Data Science) and Muhammed Idris, PhD (former PERFORM Postdoctoral Scholar, current TED Resident: www.ted.com/speakers/muhammed_idris). Housed within the PERFORM Centre, in February 2018 Data Scientifique expanded with a satellite office in the Vanier Library to make data services and workshops accessible to the wider community. Since January 2019, Data Scientifique is under the direction of Jennifer J McGrath, PhD MPH and Yara Abu Awad, ScD MS MBA (current PERFORM & Horizon Postdoctoral Scholar).



www.datascientifique.ca

Workshop Schedule

- **In this workshop, the following topics will be covered:**
  - Session I: Importing data and recoding variables
  - Session II: Merging and reshaping data

Learning Objectives

- **In this workshop participants will:**
  1. Import data from different formats including the SPSS .sav, SAS .sas7bdat and .csv
  2. Change variable format
  3. Recode variables into new categories
  4. Estimate the number of missing observations in data
  5. Merge datasets
  6. Concatenate datasets
  7. Save datasets
  8. Selecting and deleting columns
  9. Selecting and deleting rows
  10. Subsetting data

Plan for today

- I will talk for about an hour
- You will get a chance to practice afterwards with the exercises provided. I recommend that you work in pairs.
- Last 10 minutes devoted to feedback. If you need to leave early, please make sure you complete the feedback form online first! Link is on the last slide

But first: a note about conflicting functions in R -> let's open RStudio

Selecting & Deleting Columns base R

- subset function in base R to keep specific columns:

```
df2 <- subset(df1 , select = c(column1, column2, column3))
```

- to delete specific columns:

```
df2 <- subset, select = -c(column1, column2, column3))
```

or

```
df2$column1 = df2$column2 = NULL
```

Selecting & Deleting Columns dplyr

- select function in dplyr to keep specific columns:

```
df2 <- df1 %>% select(column1, column2, column3)
```

- to delete specific columns:

```
df2 <- df1 %>% select(-c(column1, column2, column3))
```

Selecting & Deleting Rows base R

- subset function in base R to select specific rows:

```
df2 <- subset(df1 , column1 < 2 & column2 == 'red'))
df2 <- subset(df1 , column1 < 2 | column2 == 'red'))
df2 <- df1[1:1000,] #keeps first 1000 rows
```

- to exclude specific rows:

```
df2 <- subset(df1, column1 != 'red')
df2 <- df1[df1$column1 != 'red',]
```

Selecting & Deleting Rows dplyr

- Filter function in dplyr to select specific rows:

```
df2 <- filter(df1, column1 == 'red' & column2 > 5)
df2 <- filter(df1, column2 > 5)
```

Doing both!

base R

```
df2 = subset(df1, column1 == 'red ', select = c(column1, column2))
```

dplyr

```
df2 = filter(df1, column1 == 'red' ) %>% select(column1, column2)
```

Merge

- Combine datasets by a column with **unique values** or multiple columns with **unique combinations** of values
- Can be thought of as adding variables (aka columns) to your dataset
- Examples of columns to merge on: unique id variable per participant, date, postal code

Example: merge by unique ID variable

df1

ID	var1	var2	var3
588	2	d	1
654	1	y	1
527	1	o	0
955	2	c	0
954	1	t	0

df2

ID	var1	var2	var3
588	290	Apples	Breakfast
654	81	Bananas	Snack
527	63	Apples	Snack
955	6	Pears	Snack
954	146	Pears	Breakfast

df3

ID	var1	var2	var3	var4	var5	var6
588	2	d	1	225	Apples	Breakfast
654	1	y	1	56	Bananas	Snack
527	1	o	0	245	Apples	Snack
955	2	c	0	46	Pears	Snack
954	1	t	0	121	Pears	Breakfast

Image source: displayr.com/what-is-data-merging/

Example: merging by zipcode

df1

ID	zipcode
588	2009
654	2037
527	2000
955	2037
954	2010

df2

zipcode	income
2000	75000
2005	82000
2009	96000
2010	88000
2037	125000

df3

ID	zipcode	income
588	2009	96000
654	2037	125000
527	2000	75000
955	2037	125000
954	2010	88000

Image source: displayr.com/what-is-data-merging/

Merge using dplyr: left_join

- The two examples we saw are left joins
- Left join keeps all rows in the left-hand data frame and only adds observations from the right-hand data frame if they match the left

```
df3 <- df1 %>% left_join(df2)
```

- This function will automatically join by all variables with common names in the two data frames. You can also specify the join column using the by = argument

```
df3 <- df1 %>% left_join(df2, by = "ID" )
```

- What if the variables have different names?

```
df3 <- df1 %>% left_join(df2,  c( "id"  = "ID" ) )
```

Merging using dplyr ctd.

- inner_join: only includes rows that match in both data sets by the common column(s)
- right_join: like left_join but keeps all observations in data set on the right
- full_join: keeps observations in both data sets and fills those not matched with NA
- semi_join: is like inner_join but only keeps columns in the left hand data set
- anti_join: drops all observations in the left-hand data set that has a match in the right-hand data set

Merging in base R

- merge function:

```
merge(x, y, by = intersect(names(x), names(y)),
      by.x = by, by.y = by, all = FALSE, all.x = all, all.y = all,
      sort = TRUE, suffixes = c(".x",".y"),
      incomparables = NULL, ...)
```

Concatenation

- To concatenate means to add rows to your data
- Useful for adding more observations
- bind_rows in dplyr is very nice for doing this
- I prefer this function to rbind in base R because it binds rows more smartly
  - It works even when columns are missing
  - It matches columns with the same names and leaves the rest as NA

Saving Data

- My two most used functions:

```
saveRDS(df, ‘nameoffile.rds’)
write.csv(df, ‘nameoffile.csv’)
```

On to the exercises!

Feedback

https://www.datascientifique.ca/feedback.html