Merging Datasets

COMM 205 - Lecture 22 - R7

Hasan Cavusoglu

2020

Agenda

- `duplicated()` function
- Combining datasets
  - `inner_join()` function
  - `left_join()` function

Checking for Duplicate Observations with `duplicated()` Function

- Datasets can sometimes be erroneous. The same observation (i.e., row) can appear more than once. -This might adversely affect analysis.
- To check for duplicate observations, `duplicated()` function can be used.

`duplicated()` function

`duplicated()` function returns a logical vector indicating which rows of the data frame passed onto are duplicates of an *earlier* row.

`duplicated()` syntax:

```
duplicated(x)
```

where

- x is the name of the data object in which you want to identify duplicated rows.

An illustrative example

```
library(tidyverse)

instructors <- data_frame(
  name=c("Hasan Cavusoglu","Carson Woo", "Hasan Cavusoglu", "Adam Saunders",
         "YM Cheung", "Hasan Cavusoglu", "Adam Saunders"),
  course=c("COMM 205", "COMM 436", "COMM 205", "COMM 438", "COMM 205",
           "COMM 205", "COMM 438"))
```

- Here is the `instructors` data frame.

name	course
Hasan Cavusoglu	COMM 205
Carson Woo	COMM 436
Hasan Cavusoglu	COMM 205
Adam Saunders	COMM 438
YM Cheung	COMM 205
Hasan Cavusoglu	COMM 205
Adam Saunders	COMM 438

An illustrative example (cont'd)

Let's use `duplicated` function:

```
> duplicated(instructors)
[1] FALSE FALSE  TRUE FALSE FALSE  TRUE  TRUE
```

instructors

	name	course	
1	Hasan Cavusoglu	COMM 205	
2	Carson Woo	COMM 436	
3	Hasan Cavusoglu	COMM 205	✓
4	Adam Saunders	COMM 438	
5	YM Cheung	COMM 205	
6	Hasan Cavusoglu	COMM 205	✓
7	Adam Saunders	COMM 438	✓

- `duplicated()` creates a logical vector with the same size of the number of rows in the data frame and returns `TRUE` for each row which is the duplicate of an earlier row (i.e. a smaller index number).
- By visually inspecting the data frame, we can see that 3rd and 6th rows are the duplicate 1st row and 7th row is the duplicate of 4th row. However, `duplicated()` only turns `TRUE` for the 3rd, 6th, and 7th row.

An illustrative example (cont'd)

- We can use `duplicated()` in `filter()` to make sure that none of the observations are repeated (duplicated).

```
instructors %>% filter(!duplicated(instructors))
```

name	course
Hasan Cavusoglu	COMM 205
Carson Woo	COMM 436
Adam Saunders	COMM 438
YM Cheung	COMM 205

Unique Identifier

- In order to distinguish observations in a rectangular data object, a column or a set of columns can be used to uniquely refers to a particular observation (i.e., row).
  - Student ID is a student registration system
  - Social Insurance Number is a tax system
  - `gvkey` and `fyear` combination in `companies` data frame.
- **Not** all rectangular data objects require a unique identifier
  - if a particular observation is **not** needed to be **retrieved**, the rectangular data object does not need to have a unique identifier.
- The intent in North American Stock Market 1994-2013 Dataset is to record financial/accounting information of each public firm (i.e. gvkey) in a given year (i.e., fyear).
- If a column or a set of columns is used as unique identifier, you may want to check if any duplicates exist in the unique identifier.

Any duplicates in (unique) identifier column

Question

Are there any duplicates in the identifier (i.e., gvkey and fyear combination) of our North American Stock Market 1994-2013 Dataset.

- Load the stock market dataset and name it `companies`.

First, let's drop all observations where `fyear` is missing by using `filter()` and just select columns that should *uniquely* identify all the rows. Ideally, we should not have any duplicates in those columns.

```
identifiers <- companies %>%
  filter(!is.na(fyear)) %>%
  select(gvkey, fyear)

sum(duplicated(identifiers))
```

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

The result indicates that the [illegible] which identifies

Question (cont'd)

```
> identifiers %>% filter(duplicated(identifiers))
   gvkey fyear
1 186121  2010
```

This shows that gvkey of 186121 in fyear 2010 is a duplicate.

Let's check those observations in our original data frame.

```
View(companies %>% filter(gvkey=="186121", fyear==2010))
```

	gvkey	datadate	fyear	indfmt	consol	popsrc	datafmt	tic	cusip	conm	curcd	fyr	act
1	186121	2010-08-31	2010	INDL	C	D	STD	AUZ.UN	0525611...	AUSTRALIAN BANC CAP SEC TR	CAD	8	NA
2	186121	2010-12-31	2010	INDL	C	D	STD	AUZ.UN	0525611...	AUSTRALIAN BANC CAP SEC TR	CAD	12	NA

at	capx	ceq	ch	che	cogs	csho	dlc	dltt	ebit	ebitda	emp	invt	lct	lt	ni	niadj	oiadp	oibdp
NA	NA	NA	NA	NA	NA	14	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
NA	NA	NA	NA	NA	NA	14	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA

ppent	pstk	rect	sale	seq	exchg	cik	costat	fic	naicsh	sich	prcc_c	prcc_f	conml
NA	NA	NA	NA	NA	7		A	CAN	NA	NA	NA	NA	Australian Banc Capital Securities Trust
NA	NA	NA	NA	NA	7		A	CAN	NA	NA	9.6	9.6	Australian Banc Capital Securities Trust

conml	ein	loc	naics	sic
Australian Banc Capital Securities Trust		CAN		6726
Australian Banc Capital Securities Trust		CAN		6726

What to do with duplicates?

- If the entire observations (the row) in a duplicate pair are identical, you can delete one row.
- If the entire observations (the row) in a duplicate pair are not identical,
  - You can try to identify which row is the accurate one. And keep the accurate one.
  - If not practical or possible, you can delete both duplicate pair (more conservative approach).
- You can see that firm with gvkey 186121 had two observations in fyear 2010. This is because the firm reported its fiscal year-end twice in 2010: once in August and once in December (see datadate). If we are not sure the accuracy of one entry over another, we could discard both to be on the safe side.

```
companies_no_duplicate <- companies %>%
  filter(gvkey!="186121" | fyear!=2010)
```

Please note that `filter()` keeps only observations which satisfy the condition specified. If the comparison leads to NA for some observations, those observation, by definition, will not be retained. That's why we did not include `filter(!is.na(fyear))` because `fyear!=2010` will be `NA` for observations with missing fyear.

Joining datasets

- In this course, we will cover two kinds of merging datasets. We will use associated functions from `dplyr` package:
  - `inner_join()`
  - `left_join()`
- Note that there are other join functions such as `right_join()` or `full_join()`. Those are outside the scope of this course.

Joining

Joining allows us to match observations from two datasets based on matching values in a column or a set of columns. Those columns can be referred to as "joining" or "join-by" columns.

inner_join() Function

inner_join(x, y) returns all rows from x where there are matching values in y, and all columns from x and y.

- By default, the function matches observations by using columns named the same in both data frames.
- You need to specify by argument otherwise (we will see later).
- To avoid unforeseeable issues, it is recommended that the data types of the joining variable(s) in both data frames must also be the same.
  - For example, if `fyear` is classified as a numerical variable in one data frame, then it should also be classified as a numerical variable in the other data frame Otherwise, the "coercion" applied might cause problems which may not be anticipated.

An illustrative example

Question

Suppose you have the following two data frames about the IS faculty at Sauder. Combine them for professors appearing in both data frames.

```
prof <- tibble(
  emp_id = c("007","008", "009", "010", "011", "012", "013"),
  name = c( "Hasan Cavusoglu", "Carson Woo", "Adam Saunders", "Ning Nan",
            "Gene Lee", "Ron Cenfetelli", "YM Cheung"),
  office = c("HA379", "HA370", "HA673", "HA368", "HA372", "HA381", "HA675"),
  email = c("cavusoglu@sauder.ubc.ca","carson.woo@sauder.ubc.ca",
            "adam.saunders@sauder.ubc.ca", "ning.nan@sauder.ubc.ca",
            "gene.lee@sauder.ubc.ca", "cenfetelli@sauder.ubc.ca",
            "YauMan.Cheung@sauder.ubc.ca"))

background <- tibble(
  emp_id = c("007","008", "009", "010", "011", "012", "014"),
  masters = c( "UT Dallas", "Toronto", NA, "Minnesota", "UT Austin",
               "Indiana" , "UBC"),
  phd = c("UT Dallas", "Toronto", "MIT", "Michigan", "UT Austin",
          "UBC" , NA))
```

prof and background data frames:

emp_id	name	office	email
007	Hasan Cavusoglu	HA379	cavusoglu@sauder.ubc.ca
008	Carson Woo	HA370	carson.woo@sauder.ubc.ca
009	Adam Saunders	HA673	adam.saunders@sauder.ubc.ca
010	Ning Nan	HA368	ning.nan@sauder.ubc.ca
011	Gene Lee	HA372	gene.lee@sauder.ubc.ca
012	Ron Cenfetelli	HA381	cenfetelli@sauder.ubc.ca
013	YM Cheung	HA675	YauMan.Cheung@sauder.ubc.ca

emp_id	masters	phd
007	UT Dallas	UT Dallas
008	Toronto	Toronto
009	NA	MIT
010	Minnesota	Michigan
011	UT Austin	UT Austin
012	Indiana	UBC
014	UBC	NA

An illustrative example (cont'd)

- Professors with emp_id 007 to 012 exist in both data frames. 013 exists only in `prof` data frame and 014 exists only in `background` data frame. If we want to merge datasets with the observations appearing in both datasets, we need to use `inner_join()` as follows.

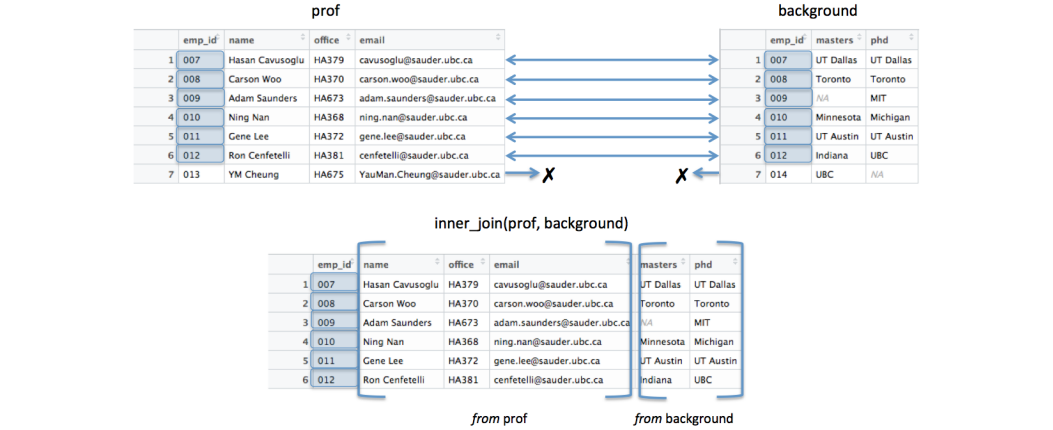
```
merged1 <- inner_join(prof,background)
```

```
## Joining, by = "emp_id"
```

- While we have not specified the column to be used to match the observations, R provided the message that `Joining, by = "emp_id"`. Here is the merged data frame (`View(merged1)`).

	emp_id	name	office	email	masters	phd
1	007	Hasan Cavusoglu	HA379	cavusoglu@sauder.ubc.ca	UT Dallas	UT Dallas
2	008	Carson Woo	HA370	carson.woo@sauder.ubc.ca	Toronto	Toronto
3	009	Adam Saunders	HA673	adam.saunders@sauder.ubc.ca	*NA*	MIT
4	010	Ning Nan	HA368	ning.nan@sauder.ubc.ca	Minnesota	Michigan
5	011	Gene Lee	HA372	gene.lee@sauder.ubc.ca	UT Austin	UT Austin
6	012	Ron Cenfetelli	HA381	cenfetelli@sauder.ubc.ca	Indiana	UBC

Pictorially how `inner_join()` works



- Essentially for matching values of the joining attribute (in this case, `emp_id`), `inner_join()` links the information appearing in both data frames. Those observations in one data frame which do not have matching observations in the other data frame are **not** included in the resultant data

left_join() Function

left_join()

left_join(x, y) returns returns all rows from x where there are matching values in y and all rows from x where there is no matching values in y; and all columns from x and y.

Example

Let's merge **prof** data frame by bringing as much information from the **background** data frame Essentially, I would like to keep all the observations in the **prof** dataframe and bring new information for the matching observations in the **background** data frame.

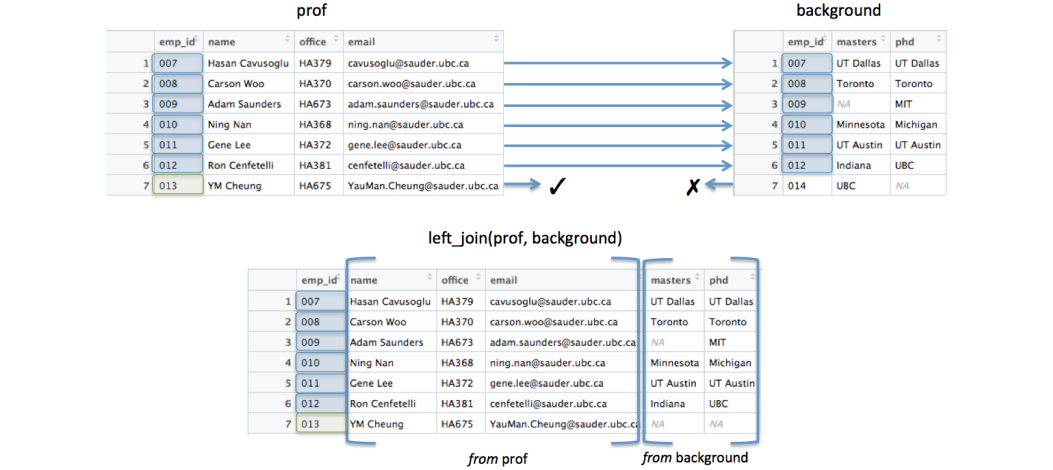
```
merged2 <- left_join(prof,background)
```

```
## Joining, by = "emp_id"
```

- Similar to `inner_join()`, while we have not specified the column to be used to match the observations, R provided the message that `Joining, by = "emp_id"`. Here is the merged data frame.

	emp_id	name	office	email	masters	phd
1	007	Hasan Cavusoglu	HA379	cavusoglu@sauder.ubc.ca	UT Dallas	UT Dallas
2	008	Carson Woo	HA370	carson.woo@sauder.ubc.ca	Toronto	Toronto
3	009	Adam Saunders	HA673	adam.saunders@sauder.ubc.ca	*NA*	MIT
4	010	Ning Nan	HA368	ning.nan@sauder.ubc.ca	Minnesota	Michigan
5	011	Gene Lee	HA372	gene.lee@sauder.ubc.ca	UT Austin	UT Austin
6	012	Ron Cenfetelli	HA381	cenfetelli@sauder.ubc.ca	Indiana	UBC
7	013	YM Cheung	HA675	YauMan.Cheung@sauder.ubc.ca	*NA*	*NA*

Pictorially how `left_join()` works.



prof

	emp_id	name	office	email
1	007	Hasan Cavusoglu	HA379	cavusoglu@sauder.ubc.ca
2	008	Carson Woo	HA370	carson.woo@sauder.ubc.ca
3	009	Adam Saunders	HA673	adam.saunders@sauder.ubc.ca
4	010	Ning Nan	HA368	ning.nan@sauder.ubc.ca
5	011	Gene Lee	HA372	gene.lee@sauder.ubc.ca
6	012	Ron Cenfetelli	HA381	cenfetelli@sauder.ubc.ca
7	013	YM Cheung	HA675	YauMan.Cheung@sauder.ubc.ca

background

	emp_id	masters	phd
1	007	UT Dallas	UT Dallas
2	008	Toronto	Toronto
3	009	*NA*	MIT
4	010	Minnesota	Michigan
5	011	UT Austin	UT Austin
6	012	Indiana	UBC
7	014	UBC	*NA*

left_join(prof, background)

	emp_id	name	office	email	masters	phd
1	007	Hasan Cavusoglu	HA379	cavusoglu@sauder.ubc.ca	UT Dallas	UT Dallas
2	008	Carson Woo	HA370	carson.woo@sauder.ubc.ca	Toronto	Toronto
3	009	Adam Saunders	HA673	adam.saunders@sauder.ubc.ca	*NA*	MIT
4	010	Ning Nan	HA368	ning.nan@sauder.ubc.ca	Minnesota	Michigan
5	011	Gene Lee	HA372	gene.lee@sauder.ubc.ca	UT Austin	UT Austin
6	012	Ron Cenfetelli	HA381	cenfetelli@sauder.ubc.ca	Indiana	UBC
7	013	YM Cheung	HA675	YauMan.Cheung@sauder.ubc.ca	*NA*	*NA*

- Essentially for matching values of the joining attribute (in this case, `emp_id`), `left_join()` links the information appearing in both data frames. Those observations in the *"left"* data frame which do not have matching observations in the other data frame will be retained in the

One-to-One Matching versus Non One-to-One Matching

One-to-One Matching: If unique identifiers are the same in both data frames, we you merge them (using their unique identifiers as join-by column(s)), each row in one data frame would match with at most one row in the other data frame.

Non-one-to-one Matching: Note that the join-by column(s) used as a basis of matching the observations in two data frames do not have to be unique identifiers of their corresponding data frames. In such situations, an observation in one data frame can match with multiple observations in the other data frame.

To illustrate non-one-to-one matching, let's consider another small data frame called `course` which has information about courses taught by IS faculty at Sauder.

```
course <- data_frame(
  employee_id = c("007", "007", "007", "008", "008", "011", "015"),
  course_no = c("205.101", "205.102", "205.103", "436.101", "436.102","337.101", "433.101"),
  capacity = c(150, 140, 140, 60, 60, 55, 50)
)
```

As you can see, the key for this dataset is `course_no`. Also note that `employee_id` is the identifier of a professor. But, the attribute name is not `emp_id` as `prof` dataset has.

Non-One-to-One Matching Example

Question

Suppose you want to create a new dataset which is an improved version of the ‘course‘ dataset by adding professor information from ‘prof‘ dataset.

```
merged3 <- left_join(course,prof, by = c("employee_id" = "emp_id"))
```

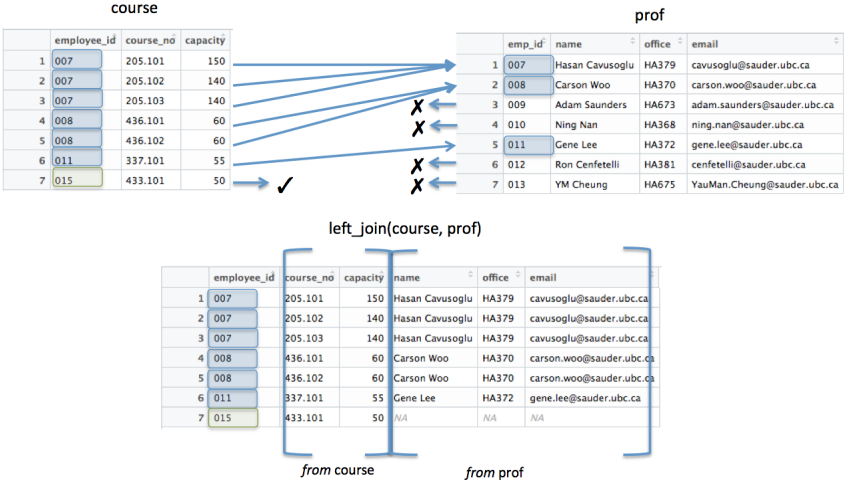
Note that join-by columns are **not** the same, we should explicitly state the join-by variables. You could confirm the output by comparing with the one below.

	employee_id	course_no	capacity	name	office	email
1	007	205.101	150	Hasan Cavusoglu	HA379	cavusoglu@sauder.ubc.ca
2	007	205.102	140	Hasan Cavusoglu	HA379	cavusoglu@sauder.ubc.ca
3	007	205.103	140	Hasan Cavusoglu	HA379	cavusoglu@sauder.ubc.ca
4	008	436.101	60	Carson Woo	HA370	carson.woo@sauder.ubc.ca
5	008	436.102	60	Carson Woo	HA370	carson.woo@sauder.ubc.ca
6	011	337.101	55	Gene Lee	HA372	gene.lee@sauder.ubc.ca
7	015	433.101	50	*NA*	*NA*	*NA*

How to specify join-by column with different names

$$\text{by} = \text{c}\ (\text{"JOIN-BY}_X\text{"} = \text{"JOIN-BY}_Y\text{"})$$

Pictorially, how `left_join()` works when the match is **not** one-to-one.



Multiple columns as matching variables to merge datasets

- We can use multiple attributes as our matching variables when merging data frames.
  - When the names of the matching attributes are the same across the two data frames, we do not need to specify the matching attributes; R will figure out the matching attributes.
  - Otherwise, we need to *explicitly* indicate which columns to use in each data frame.

Question

Suppose that you are asked to create a data frame by retaining the matching observations from the datasets provided (data1.rds and data2.rds).

- The two datasets are subsets of the North American Stock Market 1994-2013 dataset:
  - **data1.rds** contains `cnumber`, `fyear`, `cname`, `loc`, `at`, and `ch` for all observations from `fyear==2010` to `fyear==2011` inclusive and whose `loc` is either `USA` or `CAN`.
  - **data2.rds** contains `gvkey`, `fyear`, `cname`, and `emp` for all observations from `fyear==2010` to `fyear==2011` inclusive and whose `loc` is `CAN`.

Example (cont'd)

- Warnings:
  - `cnumber` in data1.rds contains `gvkey` values of the corresponding firms
  - `cname` in data1.rds contains `conm` values of the corresponding firms
  - `cname` in data2.rds contains `conml` values of the corresponding firms
- Hence, we should not let R to decide which `join-by` variables to use as R would use `fyear` and `cname`.
- Because `cname` in both datasets come from different columns of the North American Stock Market 1994-2013 dataset, we need to explicitly indicate what `join-by` variables to be used.

How to specify join-by columns with different names

$$\text{by} = \text{c}\ (\text{"JOIN-BY}_{1,X}\text{"} = \text{"JOIN-BY}_{1,Y}\text{"},\ \text{"JOIN-BY}_{2,X}\text{"} = \text{"JOIN-BY}_{2,Y}\text{"})$$

Example of merging when attribute names do not match

```
df1 <- readRDS("data/data1.rds")
df2 <- readRDS("data/data2.rds")
merged4 <- inner_join(df1, df2,
                      by = c("fyear" = "fyear",
                             "cnumber" = "gvkey"))
```

As explained in the previous slide, we specify by argument if we do not want R to decide what to use to merge data frames.

	cnumber	fyear	cname.x	loc	at	ch	cname.y	emp
1	001096	2010	MORGUARD CORP	CAN	2057.911	27.535	Morguard Corp	1.300
2	001096	2011	MORGUARD CORP	CAN	3467.210	28.755	Morguard Corp	*NA*
3	001186	2010	AGNICO EAGLE MINES LTD	CAN	5500.351	95.560	Agnico Eagle Mines Ltd	4.782
4	001186	2011	AGNICO EAGLE MINES LTD	CAN	5034.262	179.447	Agnico Eagle Mines Ltd	5.106
5	001262	2010	ALGOMA CENTRAL CORP	CAN	741.450	45.537	Algoma Central Corp	1.500
6	001262	2011	ALGOMA CENTRAL CORP	CAN	874.397	132.316	Algoma Central Corp	2.000
7	001263	2010	ESSAR STEEL ALGOMA INC	CAN	1923.200	5.700	Essar Steel Algoma Inc	3.200
8	001263	2011	ESSAR STEEL ALGOMA INC	CAN	1906.400	25.700	Essar Steel Algoma Inc	3.075

Showing 1 to 9 of 4,195 entries

The End

Thanks for watching

See you in next time!