Dealing with Duplicative Data (in R)

Erin Grand @astroeringrand

New York Data Science & AI Conference - Aug 2025

Data cleaning & exploration is >80% of the work



Uniqueness

First, we think about what the data would look like if there were no duplicates

Example: By which column is this table unique?

teachers

teacher_id	name	school_id
6775709	Sally	1
6103451	Tim	2
8932487	Naveen	1
6775709	Rachel	1
5198032	Carlos	2

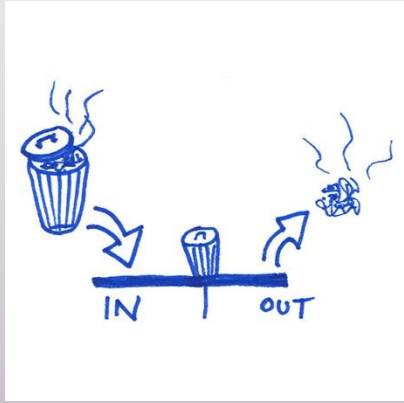
Example: By which column(s) is this table unique?

student_teacher_map

student_id	teacher_id	course	period
6775709	32432432	ELA	1
6103451	24443521	Math	4
8932487	32432432	ELA	2
6775709	24443521	Math	4
5198032	32432432	Homeroom	1

Why do we get Dupes?
(It’s not your fault)

Data Processes are most important!



If your organization doesn’t have the right data processes to start with, messy data will continue to flow no matter how much you code.

Your organization must work on the overall processes AS WELL as the individual errors.

What about organizations that are not advanced enough for a full database?

Train your people in structures and processes to reduce errors

- Ex: A student, JAMES, moved from School A to School B midyear
  - *A bad process*: School B records JAMES entering the week before, but School A doesn't record his exit until a week after he left.
  - *A better process*: School B records JAMES in their system with the correct start date, so that he is in the system to start creating class schedules. School A records JAMES as having left on his last day, and if he comes back will start a new record.

Make sure that JAMES has the same ID in School A and School B through audits.

How to fix some data organizational problems?

1: Database Engineering

- Use a database system that allows for a primary key
- Unique constraints on “surrogate” keys
- Structure tables and **documentation** so that joins between tables are clear

2: Regular data audits

- Even with identifiers, duplicates can still happen (the same person with two emails), so we use other fields to audit for duplicates
  - Names, emails, birthdays, phone numbers, and home addresses are good places to check

Remove the Duplicates from the Source Data

- Combine or delete records to fix the duplicate information

- Make sure these changes are DOCUMENTED so that there are no future questions about what happened

primary_key_1	primary_key_2	what_happened	date
18745	19745	Merged 2 into 1; Removed 2	2025-01-01
46204	42310	Removed 2	2025-01-02

Why do we get Dupes?
(This time it is your fault…)

Sometimes it’s us who creates unexpected duplicates

- *Joining* – using the incorrect fields or edits to fields needed
- *pivot_wider* – including too may columns in the select
- *pivot_longer* – including too many columns in the pivot

etc…

Joining Example

student_id	grade	subject	sc
6775709	5		
6103451			
8932			
519803			

	Math
32432432	Homeroom

What is this going to produce?

```
l                    udent_exams, student_teachers)
```

Duplicate example…

student_id	grade	subject	score	teacher_id	course
6775709	5	ELA	100	32432432	ELA
6775709	5	ELA	100	24443521	Math

```
left_join(student_exams, student_teachers,
          by = join_by("student_id"))
```

Joining Example: Is this the fix? If not what is?

student_id	grade	subject	score
6775709	5	ELA	100
6103451	7	ELA	72
8932487	5	Math	50
5198032	8	Math	97

+

student_id	teacher_id	course
6775709	32432432	ELA
6103451	24443521	Math
8932487	32432432	ELA
6775709	24443521	Math
5198032	32432432	Homeroom

```
left_join(student_exams, student_teachers,
          by = join_by(student_id,
                       subject == course))
```

Finding Duplicates (using R)

{{Janitor}} package

Janitor was built with beginning-to-intermediate R users in mind and is optimized for user-friendliness. Advanced users can already do everything covered here, but they can do it faster with janitor and save their thinking for more fun tasks. (*Sam Firke*)

If you're experienced with Tidyverse in general, you should be able to do everything inside janitor on your own, but we don't have the time to always clean up data without help.



get_dupes()

```
data %>% get_dupes(...)
```

Data frame with possible duplicates

Names of columns to check for duplicates

get_dupes with dplyr verbs

```
students %>%
  group_by(student_id) %>%
  mutate(dupe_count = n() ) %>%
  ungroup() %>%
  filter(dupe_count > 1) %>%
  select(student_id, dupe_count, everything() )
```

group_by
columns to check for duplicates

mutate
Count the # of rows in each "group"

filter
Keep only rows that have duplicates

select
Reorder the columns

get_dupes() example

```
students %>% get_dupes(student_id)
```

Data frame with possible duplicates

Names of columns to check for duplicates

student_id <chr>	dupe_count <int>	grade <dbl>	entry_date <date>	exit_date <date>
6775709	2	5	2021-02-21	2021-03-23
6775709	2	6	2021-02-21	2021-03-23

2 rows

Correcting Duplicates

Correcting Duplicates

Method 1: Correct the dupes individually with `if_else` or `case_when` and then `distinct()` to remove the straight dupes

```
students_clean <- students %>%
  mutate(grade = if_else(student_id == 6775709, 5, grade))
%>%
  distinct()
```

Method 2: Systematically remove dupes by taking the min or max of the duplicate causing column.

```
students_clean <- students %>%
  group_by(student_id) %>%
  summarize(grade = max(grade)) %>%
  ungroup()
```

Correcting Duplicates

Method 3: Output the duplicates, manually choose which version to remove, and then *anti_join* those on. Same concept as Method 3, but when there are a lot more duplicates, using a lookup table is better than using a *case_when*.

```
# Write out the dupes
get_dupes(students, student_id) %>%
  write_csv("../QC/student_id_dupes.csv", na = "")

# Open the file in Excel, add a column called remove with a 0 or flag
# to indicate whether or not you should remove it from the dataset
# Then save it as student_id_dupes_corrected.csv in the Input folder

# Read in the annotated file
# Filter to the row(s) you want to remove
dupes_remove <- read_csv("../Input/student_id_dupes_corrected.csv")
%>%
  filter(remove == 1)

# anti_join the dupes on
students_clean <- anti_join(students, dupes_remove)
```

Managing Data Changes on a Re-Run

Using `get_dupes` and `verify()` from the **assertr** package is a great way to put in checks in case the data changes (which it will).

```
check <- prac_sat_clean %>%
  get_dupes(student_id, subject) %>%
  verify(nrow(.) == 0)
```

If a student ID changes, or new duplicates occur, the code will *STOP* at this step, alerting that something is off

Thank you!

@astroeringrand