Tidy Data and Joins Activity KEY

Question 1: Assess Tidyness per the tidy data principles:

- P1: Every column is a variable
- P2: Every row is an observation
- P3: Every row is a single value

Survey dataframe

survey_id	site_id	survey_date	time_start	time_end	observer	wind_speed	air_temp
23	AT-N	2000-12-29	9:10:00	9:25:00	J. Lemmer	0	50
87	AT-S	2000-12-28	9:05:00	9:20:00	B. Rambo	0	45
370	AT-E	2001-03-09	8:45:00	9:00:00	J. Lemmer	0	54
760	AT-C	2001-06-27	8:50:00	9:05:00	D. Stuart	10	85
938	AT-W	2001-09-16	6:25:00	6:40:00	B. Rambo	0	75
total_parks	**total_sites**	**total_surveys**	**total_time**	**total_observers**	**avg_air_temp**		
1	5	5	15 minutes	3	61.8		

- Does it adhere to tidy data principles?
    - P1: No. The last row makes each column have multiple variables.
    - P2: No. Except for the last two rows, all rows have a single observation.
    - P3: No: the cells in the second-to-last row have headers instead of values.
- How to tidy: The easiest step would be to remove the last two rows of the dataframe. We can always recalculate these summary statistics.
- Tidied Survey DF:

survey_id	site_id	survey_date	observer	air_temp
87	AT-S	2000-12-28	B.R.	45
370	AT-E	2001-03-09	J.L.	54
936	AT-E	2001-09-16	B.R.	75

Taxonomy dataframe

species_id	common_name	asu_itis
EUST	European Starling	211002
MODO	Mourning Dove	162560

- Does it adhere to tidy data principles?
    - This one satisfies all three tidy data principles.

Bird dataframe

survey_id	site_id	bird_count_EUST	distance_EUST	direction_EUST	bird_count_MODO	distance_MODO	direction_MODO
760	AT-C	*NA*	*NA*	*NA*	1	0-5	E
936	AT-E	1	FT	N	*NA*	*NA*	*NA*
370	AT-E	4	20-40	SE	*NA*	*NA*	*NA*
23	AT-N	5	20-40	E	*NA*	*NA*	*NA*
87	AT-S	4	FT	W	2	FT	W
938	AT-W	1	>40	SW	*NA*	*NA*	*NA*

- Does it adhere to tidy data principles?
    - P1: No. There are variables spread across two columns. For example, two columns with values for distance.
    - P2: This dataframe mixes observations of two different bird species. Notice also that if we want to add a new observed species, we would need to add two columns instead of one row.
    - P3: Yes.

survey_id	site_id	bird_count_EUST	direction_EUST	bird_count_MODO	direction_MODO
936	AT-E	1	N	1	S
370	AT-E	4	SE	3	NE
87	AT-S	4	W	2	W

multiple observations of EUST multiple observations of MODO

- How to tidy: Transform the data from a wide format to a long format, where you keep the columns for survey_id and site_id and add columns for species, bird_count, and direction.

Tidied Birds DF:

survey_id	site_id	species_id	bird_count	direction
936	AT-E	EUST	1	N
370	AT-E	EUST	4	SE
87	AT-S	EUST	4	W
936	AT-E	MODO	1	S
370	AT-E	MODO	3	NE
87	AT-S	MODO	2	W

Site dataframe

site_id	park_code	park_district	park_name
AT-C	AT	NE	Altadena
AT-E	AT	NE	Altadena
AT-N	AT	NE	Altadena
AT-S	AT	NE	Altadena
AT-use	AT	NE	Altadena
AT-W	AT	NE	Altadena

- Does it adhere to tidy data principles?
    - This is a *tidy* data frame but not *normalized*. You can create a separate table with park code, park district and park name. And then have a site table with site id and park code.

Tidied Site table

site_id	park_code
AT-E	AT
AT-N	AT
AT-S	AT

Tidied Parks table

park_code	park_district	park_name
AT	NE	Altadena

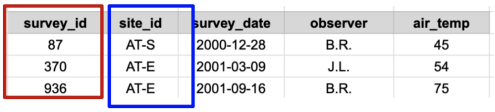
Question 2: Identify keys

Remember that:

- For tidy data, where variables and columns are equivalent, a column is a key if it has a different value in each row.
- **Primary Key:** chosen key for a table, uniquely identifies each observation in the table
- **Foreign Key:** reference to a primary key in another table

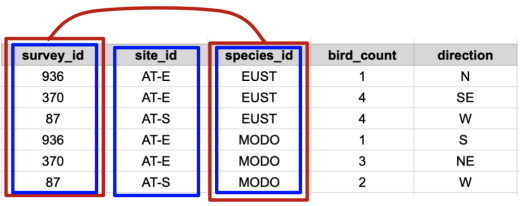
Table	Primary Key	Foreign Keys
surveys	survey_id	site_id
taxonomy	species_id	-
bird_records	survey_id + species_id (compound key)	survey_id, site_id, species_id
sites	site_id	park_code
parks	park_code	-

Primary keys are boxed in red and foreign keys are in blue.

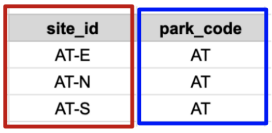


survey_id	site_id	survey_date	observer	air_temp
87	AT-S	2000-12-28	B.R.	45
370	AT-E	2001-03-09	J.L.	54
936	AT-E	2001-09-16	B.R.	75

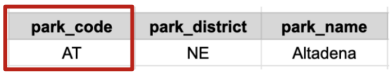
species_id	common_name	asu_itis
EUST	European Starling	211002
MODO	Mourning Dove	162560



survey_id	site_id	species_id	bird_count	direction
936	AT-E	EUST	1	N
370	AT-E	EUST	4	SE
87	AT-S	EUST	4	W
936	AT-E	MODO	1	S
370	AT-E	MODO	3	NE
87	AT-S	MODO	2	W



site_id	park_code
AT-E	AT
AT-N	AT
AT-S	AT



park_code	park_district	park_name
AT	NE	Altadena

Question 3: Join the data tables

There are many ways to join all tables to form a single one. This is one example.

Step 1:

- Do a left join of the sites dataframe (on the left) with parks dataframe (on the right), join by **park_code**. This gets us back to our denormalized dataframe. Let's call the resulting dataframe DF1.

sites

site_id	park_code
AT-E	AT
AT-N	AT
AT-S	AT

parks

park_code	park_district	park_name
AT	NE	Altadena

Result: `df1 = left_join(sites, parks, by = "park_code")` (note, any join will work here but **left_join** is a good default option

site_id	park_code	park_district	park_name
AT-E	AT	NE	Altadena
AT-N	AT	NE	Altadena
AT-S	AT	NE	Altadena

Step 2:

- Join **df1** with **surveys** data frame by using the **site_id** key.
- Notice that not all elements of **site_id** in **df1** are used in the **site_id** column of the **surveys** data frame. This means we need to decide how we want to join the data because there can be NAs depending on the join we choose.
- Let's do an **inner_join** between both data frames so we only keep sites where we have observations. Call the resulting data frame **df2**.

df1 (from Step 1)

site_id	park_code	park_district	park_name
AT-E	AT	NE	Altadena
AT-N	AT	NE	Altadena
AT-S	AT	NE	Altadena

surveys

survey_id	site_id	species_id	bird_count	direction
936	AT-E	EUST	1	N
370	AT-E	EUST	4	SE
87	AT-S	EUST	4	W
936	AT-E	MODO	1	S
370	AT-E	MODO	3	NE
87	AT-S	MODO	2	W

Result: df2 = inner_join(df1, surveys, by = "site_id")

survey_id	site_id	survey_date	observer	air_temp	park_code	park_district	park_name
87	AT-S	2000-12-28	B.R.	45	AT	NE	Altadena
370	AT-E	2001-03-09	J.L.	54	AT	NE	Altadena
936	AT-E	2001-09-16	B.R.	75	AT	NE	Altadena

Step 3:

- Join the **taxonomy** data frame to the **bird_records** data frame by **species_id**.
- This is similar to step 1: do a **left_join** of the bird_records (on the left) to the taxonomy data frame (on the right). We get a new **df3**.

birds_records

survey_id	site_id	species_id	bird_count	direction
936	AT-E	EUST	1	N
370	AT-E	EUST	4	SE
87	AT-S	EUST	4	W
936	AT-E	MODO	1	S
370	AT-E	MODO	3	NE
87	AT-S	MODO	2	W

taxonomy

species_id	common_name	asu_itis
EUST	European Starling	211002
MODO	Mourning Dove	162560

Result: df3 = left_join(birds_records, taxonomy, by = "species_id")

survey_id	site_id	species_id	bird_count	direction	common_name	asu_itis
936	AT-E	EUST	1	N	European Starling	211002
370	AT-E	EUST	4	SE	European Starling	211002
87	AT-S	EUST	4	W	European Starling	211002
936	AT-E	MODO	1	S	Mourning Dove	162560
370	AT-E	MODO	3	NE	Mourning Dove	162560
87	AT-S	MODO	2	W	Mourning Dove	162560

Step 4: At this point, we only have two datasets:

---

df2 = inner_join(df1, surveys, by = "site_id")

survey_id	site_id	survey_date	observer	air_temp	park_code	park_district	park_name
87	AT-S	2000-12-28	B.R.	45	AT	NE	Altadena
370	AT-E	2001-03-09	J.L.	54	AT	NE	Altadena
936	AT-E	2001-09-16	B.R.	75	AT	NE	Altadena

df3 = left_join(birds_records, taxonomy, by = "species_id")

survey_id	site_id	species_id	bird_count	direction	common_name	asu_itis
936	AT-E	EUST	1	N	European Starling	211002
370	AT-E	EUST	4	SE	European Starling	211002
87	AT-S	EUST	4	W	European Starling	211002
936	AT-E	MODO	1	S	Mourning Dove	162560
370	AT-E	MODO	3	NE	Mourning Dove	162560
87	AT-S	MODO	2	W	Mourning Dove	162560

- Notice that the simplest option for a primary key for **df2** would be **survey_id**. There's also the compound key **survey_id + site_id**.
- These two keys (**survey_id** and **survey_id + site_id**) appear as foreign keys in **df3**. So, we could do **left_join(df3, df2, by = ...)** by any of these keys.
- If we do the **left_join(df3, df2, by = ...)** by **survey_id** we would get a repeated **site_id** column (the **site_id** from **df2** and the **site_id** from **df3;** R would differentiate by naming them **site_id.x** and **site_id.y**).
    - Here this would be just an annoyance, since all the **site_id.x** and **site_id.y** would match, but this can be problematic if multiple **site_id** values matched with any **survey_id** values or vice versa.
- If we do the join by **survey_id + site_id** we would not get any repeated columns. So, to avoid any duplicate columns, let's do **left_join(df3, df2, by = c("survey_id", "site_id")**.

Final Data

survey_id	site_id	species_id	bird_count	direction	common_name	asu_itis	survey_date	observer	air_temp	park_code	park_district	park_name
936	AT-E	EUST	1	N	European Starling	211002	2001-09-16	B.R.	75	AT	NE	Altadena
370	AT-E	EUST	4	SE	European Starling	211002	2001-03-09	J.L.	54	AT	NE	Altadena
87	AT-S	EUST	4	W	European Starling	211002	2000-12-28	B.R.	45	AT	NE	Altadena
936	AT-E	MODO	1	S	Mourning Dove	162560	2001-09-16	B.R.	75	AT	NE	Altadena
370	AT-E	MODO	3	NE	Mourning Dove	162560	2001-03-09	J.L.	54	AT	NE	Altadena
87	AT-S	MODO	2	W	Mourning Dove	162560	2000-12-28	B.R.	45	AT	NE	Altadena