



National Institute of  
Diabetes and Digestive  
and Kidney Diseases

# Overview of AI-Readiness and Preparing AI-Ready Datasets

Courtney Shelley, PhD, Booz Allen Hamilton

*NIDDK-CR Office Hours: December 1 through 15, 2023*



NIDDK Central Repository  
Supporting the NIDDK scientific and research community

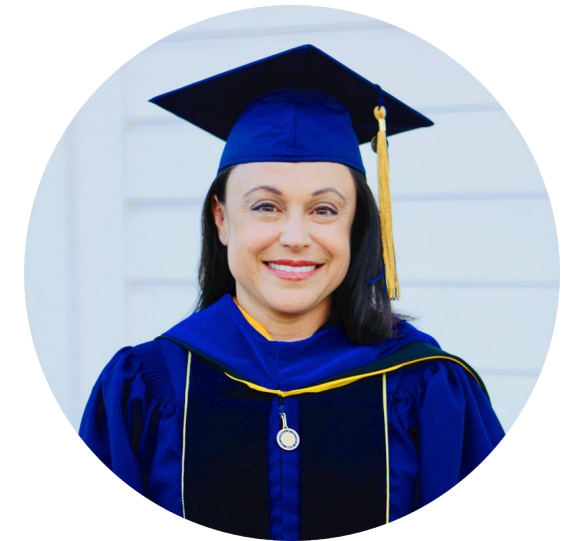
# Speaker Introduction

**Courtney D. Shelley, PhD**, is a Health Data Scientist at Booz Allen Hamilton, where she focuses on data science education and AI-readiness of health-related data.

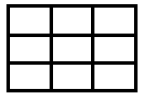
She has supported the NIH Office of Data Science Strategy to develop online data science learning resources for pre-college and collegiate audiences, and to assess data science education across US universities to promote collaborative research between biomedical researchers and AI professionals.

Prior to working at Booz Allen Hamilton, Dr. Shelley worked at Los Alamos National Laboratory, where she received the Postdoctoral Distinguished Performance Award for COVID-19 response efforts at local, state, and federal levels, as well as conducted research in suicide prevention with the support of the Department of Veterans Affairs and Million Veteran Program.

She completed her PhD in Epidemiology with a focus on causal inference at University of California, Davis.



# Data Centric Challenge – Submission Requirements



1. **A single “raw” dataset** A single “Raw” dataset resulting from data aggregation and harmonization of all study data files (from TEDDY or TrialNet – as per level of participation), but has not otherwise been altered. This dataset must be represented as a single rectangular file (i.e., tabular, spreadsheet, or matrix) in .csv file format



2. **An “AI-ready” version** of the raw dataset that has been enhanced for AI-readiness. This dataset must be represented as a single rectangular file (i.e., tabular, spreadsheet, or matrix) in .csv file format



3. **The code script**, in Python or R, used to generate the raw and AI-ready files submitted to your private GitHub repository



4. **A human-readable data dictionary/codebook** documenting the AI-ready dataset (Excel format preferred, with the following information included at a minimum: variable name, variable label/description, variable type, measurement unit as applicable (e.g., pounds, kilograms), and corresponding code lists as needed (e.g., 0 = No, 1 = Yes).



5. **Challenge Solution Submission Form**, describing the 1) AI-ready dataset, 2) methods for preparing the AI-ready dataset, and 3) potential use cases for the prepared dataset as it relates to T1D, or other disease areas of interest to NIDDK.

Submission Instructions are posted to the Challenge.gov under the [How to Enter](#) tab for Phase 2: Data Enhancement

# What is an AI-ready dataset?

**AI-readiness** refers to data that are *machine-readable, reliable, accurate, explainable, predictive, and accessible for future AI applications*

- An AI-ready dataset consists of:
  - Data that is reflective of the population from which it was drawn
  - Data that is well documented and FAIR (findable, accessible, interoperable, and reusable)
  - Data that is model-agnostic
- AI-readiness will include:
  - ✓ **pre-processing steps** such as addressing errant values,
  - ✓ **handling of missing values,**
  - ✓ **relabeling and recoding** of data elements (aka columns, variables, features, or attributes) and values during harmonization to ensure consistency and standardized formatting
  - ✓ **documentation** of all data handling steps, all variables, and the dataset itself
- When possible,
  - attempt to **retain as much information as possible** by creating new data elements that are transforms of existing elements without deleting or overwriting existing elements.



National Institute of  
Diabetes and Digestive  
and Kidney Diseases

# Performing Pre-Model Processing and Data Quality Checks using R



NIDDK Central Repository

Supporting the NIDDK scientific and research community

# Importing Data

- Installing packages:
  - Only have to do this ONCE

```
install.packages("package-name") # Note the double quotations!
```

- To use a package:
  - Will need to do this once every session

```
library(package-name) # No quotations here
```

- Putting it all together -

```
install.packages("openxlsx")  
library(openxlsx)  
df <- read.csv("data.xlsx", header = TRUE)
```

# Understanding the Data

- Check you read it in correctly
- Get a feel for its size and complexity

```
> head(df)      # first five rows of dataset. Does this look as expected?
> dim(df)       # dimensions of dataset in rows, columns
> names(df)     # column names
```

# Understanding the Data

- Assess Variable Types:
  - Each value is an **element**
  - Data types in R include: character, numeric, integer, complex, and logical
    - Integer (denoted with an L, rarely used in health science)
    - Complex (<REAL> + <IMAGINARY>I, also rarely used in health science)
    - Numeric are any numbers, including negative and decimal values
    - Logical is TRUE/FALSE
    - Character is “Latino”, “always”, “California”

```
> class(6)
[1] "numeric"
> class(TRUE)
[1] "logical"
> class("friend")
[1] "character"
```



# Understanding the Data

- R can handle more complexity also, including vectors, matrices, data frames, and lists. These **objects** are composed of elements:

- o A vector is made with the concatenate function, `c()`:

```
> c("red", "yellow", "green")  
> c(1, 2, 87)
```

- A matrix is made of numeric elements:

```
> matrix(1:25, nrow = 5)
```

- o A data frame is made of many vectors of the same length:

```
> data.frame(color = c("red", "yellow", "green"),  
+           age = c(1, 2, 87))
```

# Understanding the Data

1. All elements of a vector must be the same type.
2. R uses a process called **coercion** to attempt to make sense of input.



THEREFORE: If you create a vector like `c("red", 1, TRUE)`, R will coerce the elements to all be the same type. In this case, it will force 1 and TRUE to also be characters.

```
> c("red", 1, TRUE)
[1] "red"  "1"    "TRUE"
```

- **Why do you care?** Because a typo will coerce a vector to become a different type than you expect.
  - Ex: A typo such as 4. when you meant 4.0 will be read as a character, so that the entire vector will then be coerced to character. Now it won't behave as a numeric vector when you try to analyze it - you won't be able to find min/max or do math.
- **Always ask yourself - what class do I expect from this variable? If it doesn't look how you expected it to, check for errors!**

# Quiz

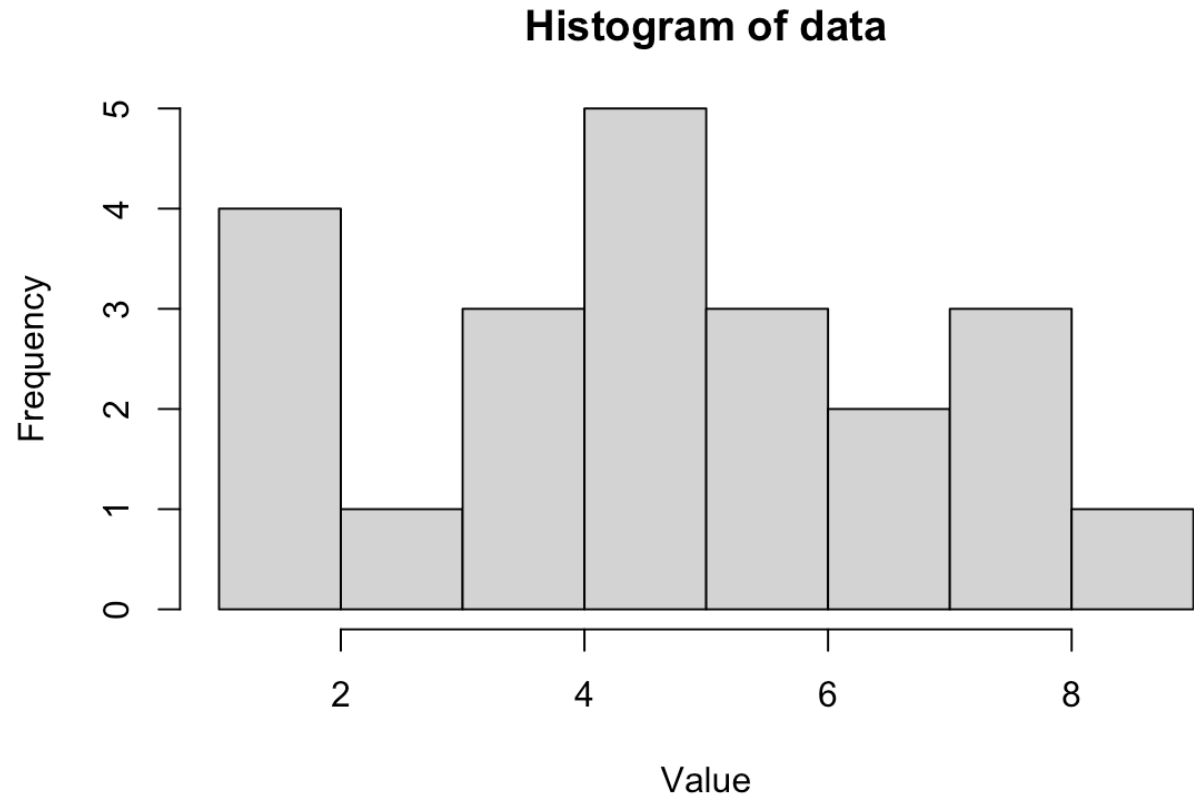
Why do you want to use a programming language like R or Python or SAS, rather than spreadsheet software like Excel for data exploration and analysis?

# Visualizing Data

## HISTOGRAMS

- Suitable for numeric data with (at least theoretically) continuous values.
- Creates a specified number of bars representing a value range with height equal to the number of observations within that range.

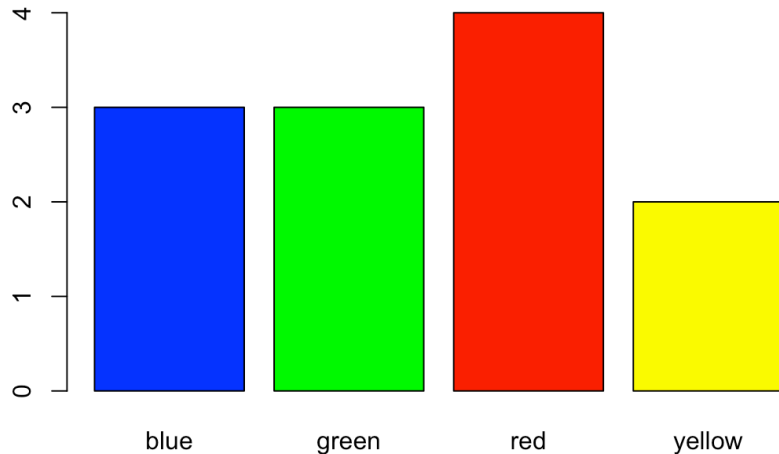
```
data = c(1,1,4,5,8,3,5,7,9,1,7,  
         1,4,5,6,8,6,5,4,5,6,8)  
  
hist(data)
```



# Visualizing the Data

## BARPLOTS

- Suitable for categorical (i.e., count) data.
- Generates a bar for each category with height representing the number of observations within each category.



```
data = c("red", "yellow", "green", "red",  
         "red", "blue", "blue",  
         "yellow", "red", "green",  
         "green", "blue")  
table(data)
```

```
data  
  blue green red yellow  
    3    3  4    2
```

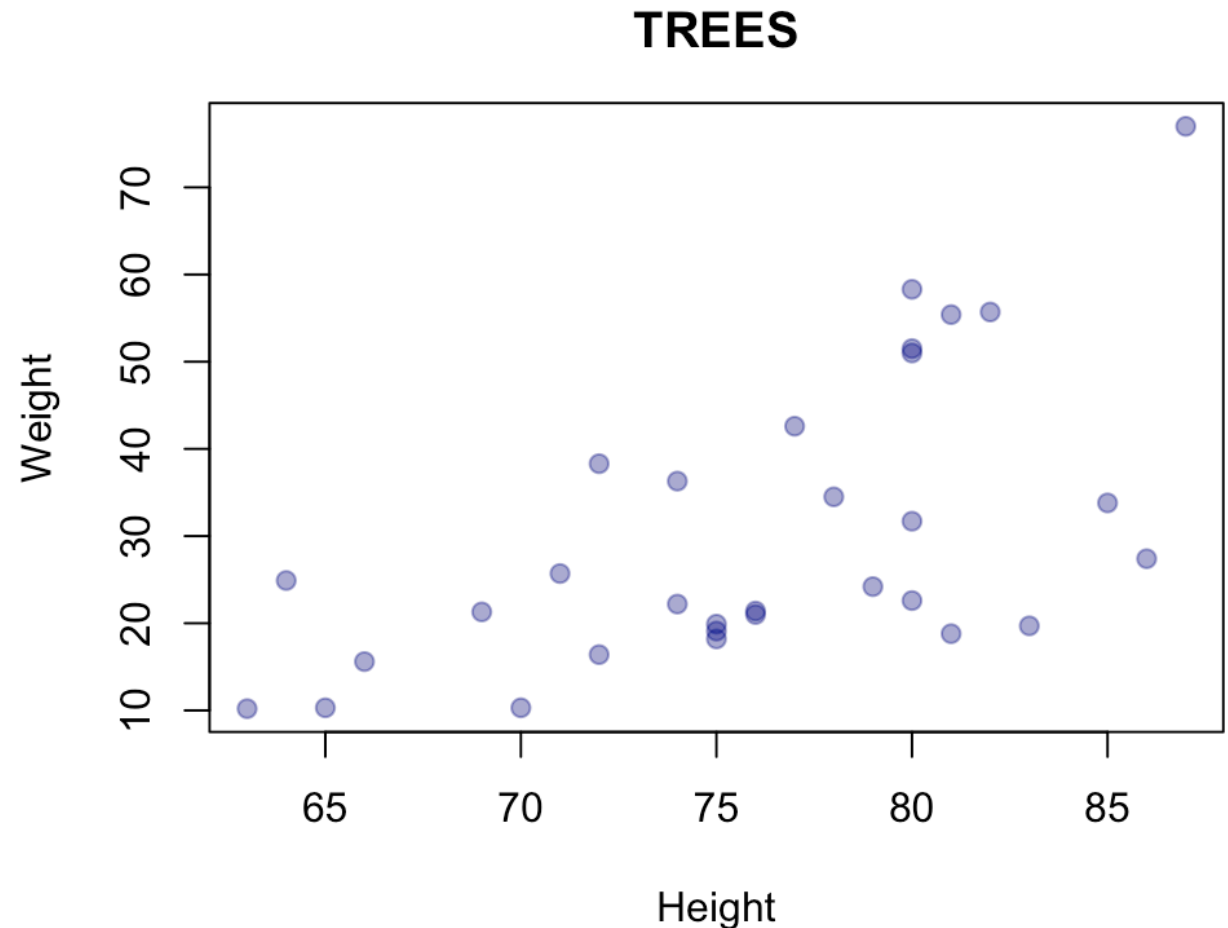
```
barplot(table(data), col = c("blue",  
                             "green", "red", "yellow"))
```

# Visualizing the Data

## SCATTERPLOTS

- Suitable for viewing the relationship between two continuous variables
- Most often plotted with the independent variable on the x-axis and the dependent (response) variable on the y-axis

```
plot(Height, Volume, data = TREES)
```



# Quiz

I have 20 observations on patients' reported gender, sex assigned at birth, height and weight. The data is in a .xlsx format, so I can open it in Excel.

1. What do you expect the class of gender, sex, height, weight to be?
  - a) "category", "category", "number", "number"
  - b) "factor", "factor", "integer", "integer"
  - c) "character", "character", "numeric", "numeric"
2. Which two variables can I visualize together using a scatterplot?
  - a) gender and sex
  - b) sex and weight
  - c) height and weight

# Quiz

3. In R, I ran `hist(weight)` and received the following error:

```
Error in hist.default(data$weight) : 'x' must be numeric
```

What is wrong and what could I do to investigate?



# Data Cleaning and Pre-Processing

## TYPOS

- A common place we see typos is in self-identified categories.
- Sometimes datasets will use codes rather than the actual labels so that these variables look and behave like numeric count data instead of categorical data:
  - When Race is input as 1 = White, 2 = Black, 3 = Hispanic/Latino, 4 = Other, we can calculate `mean(Race)`, but what does that mean?
  - R can handle **factors** so that categorical names will be treated as labels.
  - **BUT...** if these same values are input by name, we tend to see “white”, “White”, “WHITE”, “hisp”, “hispanic”, “Latina”, which creates all kinds of categories with the same meaning!
- The function `table()` is very helpful because it will work on both numeric data and character data.
- Recoding these isn't too hard – just pick a standard and stick to it! (*we recommend referring to a standard ontology like SNOMED*).

# Data Cleaning and Pre-Processing

## TYPOS

- Typos may also look like ***clinically implausible values*** or extreme outliers. Assessing `min()` and `max()` for numeric values will often find these.
  - Example: Age of 250 instead of 25 or a height of 5.4 cm that is probably 5.4 ft.
- When handling medical data, you'll likely need to speak with a clinician to understand the clinically plausible values of laboratory measurements, as well as understanding what the laboratory cut-off values may be (such as all readings over 1,000 recorded as 1,000, which will look like a normal curve with its tail cut off).

# Data Cleaning and Pre-Processing

## MISSINGNESS

- May occur in different ways:
  - **Missing Completely At Random (MCAR):** the fact that data is missing is independent of the data value itself, such that there is no systematic difference between those with missingness and those without, such as a batch of laboratory samples processed improperly that result in missing laboratory values.
  - **Missing At Random (MAR):** Missingness is systematically related to observed data, such as male patients being less likely to answer survey questions about depression. Here the probability of completing the survey is due to being male, not the severity of the depression. This can be seen in medical studies if patients cannot get time off work to attend follow-up visits.
  - **Missing Not At Random (MNAR):** Similar to MAR, but missingness is systematically related to unobserved data, such as not answering the survey about depression based on severity of depression. This can be seen in medical studies if patients are too ill to attend follow-up visits

*Handling missingness depends on the type of missingness observed!*

# Data Cleaning and Pre-Processing

- Regrouping categories (such as combining two variables of RACE and ETHNICITY into a single RACE\_ETHNICITY variable) can help reduce missingness, though will also reduce information in the data as data handling decisions are made:

RACE	ETHNICITY
White	Hispanic/Latino
Black	Not Hispanic/Latino
Asian	
Other	



RACE_ETHNICITY
NonHispanic_White
Hispanic_White
NonHispanic_Black
Hispanic_Black
NonHispanic_Asian
Hispanic_Asian
NonHispanic_Other
Hispanic_Other

- It's up to you to assess that this regrouping is "better". Statistical tests can assess whether those who did not answer one question were more likely to also not answer the other question, or that those who did not answer these questions did not also systematically differ in other measured ways.
- *But how would you know if they differed in unmeasured ways?*


# Data Cleaning and Pre-Processing

## MISSINGNESS

- A common approach to handling missingness is to delete those rows or columns that contain missingness. But this will also necessarily reduce sample size or potential explanatory features.
  - When deleting rows (i.e., observations or patients), ensure those with missingness are not systematically different from those with complete information.
  - If feasible, a new variable can be created that indicates missing data such that models can be fitted with this variable that omit the observation from calculations but can show a signal for missingness itself. This is how censoring works in time-to-event analysis.
  - When deleting columns (i.e., variables or features), use a systematic rule such as the column must contain at least 5% filled cells.

# Data Cleaning and Pre-Processing

- Combining related columns may help – sort of the reverse of one-hot encoding or indicator (dummy) variables that we’ll discuss later.

Do you feel anxious about your diagnosis?	Do you feel sad about your diagnosis?	Do you feel accepting of your diagnosis?	
Yes	<b>NA</b>	No	 <i>How do you feel about your diagnosis?</i> 1 = Anxious 2 = Sad 2 = Sad 1 = Anxious 2 = Sad 4 = Other/NA 5 = Accepting
No	Yes	<b>NA</b>	
No	Yes	<b>NA</b>	
Yes	No	<b>NA</b>	
<b>NA</b>	Yes	<b>NA</b>	
<b>NA</b>	No	<b>NA</b>	
<b>NA</b>	<b>NA</b>	Yes	

- Another solution is to create a variable that systematically accounts for missingness such as “Did Patient answer any of Questions 9-12?” Now those who did not answer can be “No” instead of “NA” for four responses.

# Data Cleaning and Pre-Processing

- Imputation is a method for predicting values based on surrounding data. If data is MCAR, missing values can be inferred from the complete dataset. If data is MAR, a predictive model fit using only similar participants can predict missing values (in the depression and gender example, predict missing male values from males who did answer). If data is MNAR, imputation is more complex but still possible.

Pedersen AB, Mikkelsen EM, Cronin-Fenton D, Kristensen NR, Pham TM, Pedersen L, Petersen I. Missing data and multiple imputation in clinical epidemiological research. Clin Epidemiol. 2017 Mar 15;9:157-166. [doi: 10.2147/CLEP.S129785](https://doi.org/10.2147/CLEP.S129785). PMID: 28352203; PMCID: PMC5358992.

# Quiz

- In our last quiz example, we encountered an issue with recorded weights. The error caused `hist(weight)` to not run, with an error message that our data was not numeric class. This was probably a typo that caused one value to be character rather than numeric, which coerced the entire vector to character class.
- The data is read into R and stored as an object called `study`. I viewed `study$weight` and observed the following:

```
[1] "118.4" "164"    "191.9" "149.2" "156.9" "189.5" "146.2" "135.3" "165.3" "121"    "179.4" "151.2"  
[13] "136."  "136.7" "162.1" "164.6" "121.1" "137.8" "149.9" "120.8"
```

What can be done to fix this typo?



# Data Cleaning and Pre-Processing

## DATES

- Dates are also an element class in R.
- Dates may be recorded in separate columns of DAY, MONTH, and YEAR or may be recorded in single columns but with differing formats, such as “19 AUG 2023”, “2023-11-07”, or “2023/11/07”.
- In R, all date types can be handled using `as.Date()`

```
as.Date('1/15/2001', format='%m/%d/%Y')
[1] "2001-01-15"
> as.Date('April 26, 2001', format='%B %d, %Y')
[1] "2001-04-26"
> as.Date('22JUN01', format='%d%b%y')    # %y is system-
specific; use with caution
[1] "2001-06-22"
```

# Data Cleaning and Pre-Processing

- If in a three-column form, you will first need to collapse columns of DAY, MONTH, YEAR into a single column

```
dates <- paste(DAY, MONTH, YEAR, sep = "/")  
as.Date(dates, "%d/%b/%y")
```

You'll probably have to play around for a while to ensure your code does what you want it to do.

Be sure to check simple cases first and create a new variable rather than altering your data!

# Data Cleaning and Pre-Processing

## CREATING NEW VARIABLES

- Categorical variables will most likely be treated separately in any statistical analysis you do. If you have a categorical variable of car color, one category (or level or factor) will be treated as the referent category and all other categories will compare to this one. This is achieved using one-hot encoding or indicator variables (i.e., dummy variables). For a categorical variable with  $n$  categories,  $n - 1$  new variables will be created and filled with 0/1 to indicate whether or not each observation is in each category.

COLOR		COLOR_BLUE	COLOR_RED	COLOR_BLACK
White	→	0	0	0
Blue		1	0	0
Red		0	1	0
Black		0	0	1

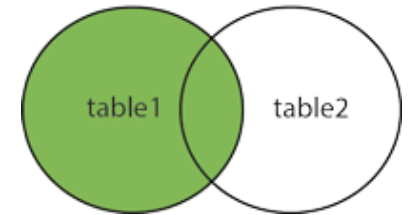
- Here white is the reference color such that white is inferred by not being blue, red, or black. A model fit with these values will estimate differences from white car color.

# Data Handling and Pre-Processing

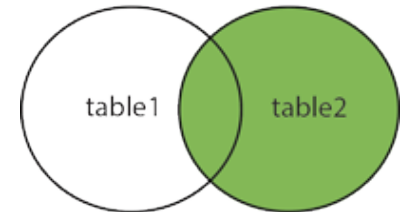
## MERGING FILES

- When working with files from a database, you will likely need to merge separate files on the same patients to achieve a single tabular (i.e., spreadsheet) dataset. Merging can be achieved with several R packages and differing techniques.
- Two easy to use methods for two datasets, `df1` and `df2`:
  - **Base R's** `merge()` function: `merge(df1, df2)`. Arguments within this function can specify which columns to merge if the names differ, and whether you want to keep all rows or only those with a match in both datasets.
  - **dplyr's** `join()` family of functions. The `dplyr` package uses SQL database syntax.
    - A *left join* means: Include everything on the left (what was the `df1` in `merge()`) and all rows that match from the right (`df2`) data frame. If the join columns have the same name, all you need is `left_join(df1, df2)`. If they don't have the same name, you need a `by` argument, such as `left_join(x, y, by = c("df1ColName" = "df2ColName"))`.
    - There is also `right_join()`, `inner_join()`, and `full_join()`.

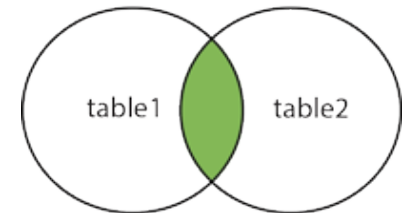
LEFT JOIN



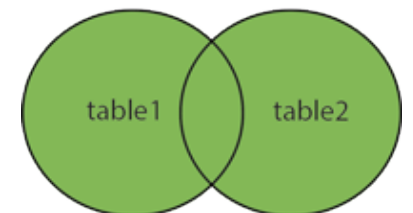
RIGHT JOIN



INNER JOIN



FULL OUTER JOIN



# Harmonization and Fusion

## HARMONIZATION

- Similar datasets collected over several years or in several locations under the same study design can be combined through ***data harmonization***.
  - This may require renaming columns so they match:
    - `date_diagnosed` and `date_of_diagnosis`, or creation of `date_diagnosed` by combining `day_diagnosed`, `month_diagnosed`, and `year_diagnosed`
  - When combining studies over several years or locations, create a new column of `YEAR` or `LOCATION` so that the original datasets can be examined separately.
  - ***NOTE: Harmonized datasets should be VERY SIMILAR.***

# Harmonization and Fusion

## FUSION

- Data fusion is the process of combining multiple datasets to test hypotheses and find patterns that would not be testable in a single available dataset.
- Combining multiple datasets multiplies your potential for introducing bias.
  - One such bias is the **ecological fallacy** – making inferences about individuals based on aggregate data for a group. AI is very good at committing this one by inferring race based on home address and neighborhood characteristics, inferring sexuality based on social media Likes
  - Another bias is reusing datasets that weren't collected for research purposes. Troublesome datasets will include data collected from cell phones or social media, which have known selection biases of age, urban/rural, and SES.
- Be careful about drawing causal conclusions from fused datasets. **Causal data fusion** is a branch of computational epidemiology with a growing body of theory that should be referenced.

# Data Documentation

Now that we've done all these steps, we should have a clean and AI-ready dataset that requires accompanying documentation so others can properly use the data.

## ELEMENTS OF A DATA DICTIONARY:

1. **Document the dataset itself** – the study or source of the data including details on inclusion/exclusion criteria, source population and target population, sampling schema used (if applicable). The goal is to not need to contact any original data creators for further details but to be able to successfully apply the data to a new application without introducing bias or non-portability
- 2. **Document the data elements** – again, the goal is for future users to not need to interact with the dataset creator (you!) so sufficient documentation *of each element* means:
  - a thorough description of what was collected, why, and how
  - variable type (e.g., numeric), unit of measurement (e.g., pounds, kilograms), and corresponding code lists (e.g., 0 = "No", 1 = "Yes")
  - summary statistics of each element ( $N$ ,  $N$  missing, min, max, median, mean, and interquartile range of continuous numeric values or  $N$ , categories and  $n$  for each).
  - missingness notation and codes used for categories if these were used.
  - Along with above, document for each element what pre-processing steps were taken.