3.4: Combining Data (Joins and Binds)

The Context

After the series of incidents where a number of the collars made by Budget Collars LLC seem to be failing, our team decided to try and replace as many of them as possible. Besides, their battery life is far inferior.

We're placing as many collars on as many seals as we can, and we are starting to run short on collars.

One of our intrepid data science team members found an old box of collars and some data on all of the collars (it was a real challenge getting this data off of an old floppy drive, but she managed).

Our Tasks

We're going to spend the next lessons tackling two tasks:

1) First, we'll work to join data sets together.

Our main goal is to create a single data set that includes the data we have previously looked at for collars, the data on the new collars, as well as the additional data on the old collars.

This is a little tricky, as the collar IDs need to be matched up with their counterparts across datasets, and new collars need be added. Unfortunately, the new collars have IDs and some additional data, but we don't know the maker of the new collars.

2) Second, we will try to identify the maker of mystery collars.

In order to do this, we will skim the surface of machine learning. We can use a common classification algorithm, K-Nearnest Neighbors (KNN), to predict which maker made which of our unidentified collars. Don't worry, we won't go into *too* much detail here, but I want to you to at least have a general idea of how it works.

Combining Data (Joins and Binds)

Often times, we have a lot of data for one project that are related but storing all of the data in one file would add unnecessary redundancy (e.g., data in certain rows would need to be repeated too often). Other times, data has been collected separately and needs to be combined before analysis.

Being able to join together data from related tables is a key skill in data science, and for working with larger data structures (databases with their own languages, like SQL).

The Data

Let's load in the `tidyverse` and the data we're working.

```
# Load the tidyverse
library(tidyverse)
```

```
Warning: package 'tidyverse' was built under R version 4.2.3

Warning: package 'ggplot2' was built under R version 4.2.3

Warning: package 'tibble' was built under R version 4.2.3

Warning: package 'tidyr' was built under R version 4.2.2

Warning: package 'readr' was built under R version 4.2.3

Warning: package 'purrr' was built under R version 4.2.3

Warning: package 'dplyr' was built under R version 4.2.3

Warning: package 'stringr' was built under R version 4.2.2

Warning: package 'forcats' was built under R version 4.2.2

Warning: package 'lubridate' was built under R version 4.2.3
```

```r
# Loading in the data
collars <- read_csv("data/collar_data.csv")
new_collars <- read_csv("data/new_collars.csv")
old_collars_new_data <- read_csv("data/old_collars_new_data.csv")

# Tell me about these data, how are we going to join them?
```

First, let's explore our data. We want to focus on 2 things here:

(1) The columns: Which ones match columns in other datasets?
(2) Collar identity: Which datasets have matching collars or new collars?

```r
# View first few rows of each dataset
head(collars)
```

```
# A tibble: 6 x 5
  collar_id maker          battery_life signal_distance  fail
      <dbl> <chr>                 <dbl>           <dbl> <dbl>
1         1 Collarium Inc.         141.            4171.     0
2         2 Collarium Inc.         121.            4134.     0
3         3 Collarium Inc.         126.            4277.     1
4         4 Collarium Inc.         127.            4198.     0
5         5 Collarium Inc.         141.            4173.     1
6         6 Collarium Inc.         105.            4175.     0
```

```r
head(new_collars)
```

```
# A tibble: 6 x 6
  collar_id maker battery_life signal_distance antenna_length weight
      <dbl> <lgl>        <dbl>           <dbl>          <dbl>  <dbl>
1       129 NA           104.            4322.           5.86   21.8
2       138 NA            91.2           4297.           6.78   20.1
3       134 NA            88.0           4279.           6.77   23.0
4       140 NA            79.9           4253.           7.91   22.9
5       104 NA           127.            4179.           4.82   23.3
6       118 NA           127.            4208.           5.67   23.6
```

```r
head(old_collars_new_data)
```

```
# A tibble: 6 x 4
  collar_id maker              antenna_length weight
      <dbl> <chr>                       <dbl>  <dbl>
1        30 Collarium Inc.               6.74   20.2
2        51 Budget Collars LLC           5.99   25.1
3        60 Budget Collars LLC           5.56   18.4
4        19 Collarium Inc.               4.94   23.1
5        53 Budget Collars LLC           4.95   18.6
6        25 Collarium Inc.               5.17   24.4
```

Diagramming

In small groups, talk through the process of combining these three datasets. Think about the following:

- Which columns match and which ones don't?
- Which rows match and which ones don't?
- Does the order in which we combine datasets matter?

Draw out a diagram that represents how this process might go.

Joins vs. Binds

Now that we've decided on a process for how to combine our data, let's figure out which functions we are going to use to accomplish this task.

We have 2 main methods of combining data sets, and they work in different ways.

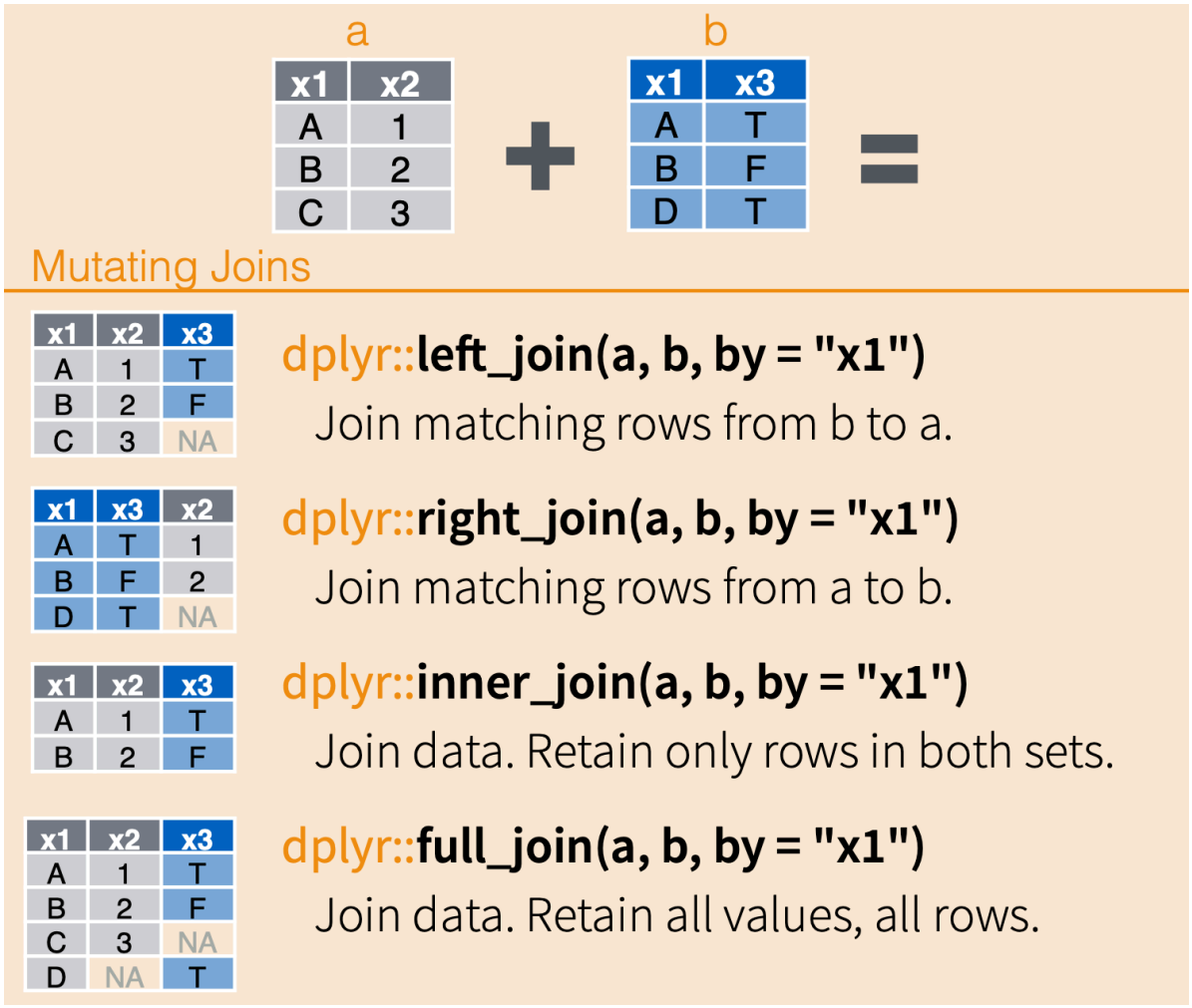
Joins

Joins are arguably the more complicated of the two types of ways to combine data, but they are, therefore, the more flexible and useful.

The magic of comes comes because they match up columns of data based on unique identifiers in each row of data.

In the following diagram, the two example data frames have the column `x1` in common, and each of the values in `x1` are unique (no repeats in the same data frame). When combining the data sets, all of the columns are added, and their rows are matched up to their respective values in the `x1` column.

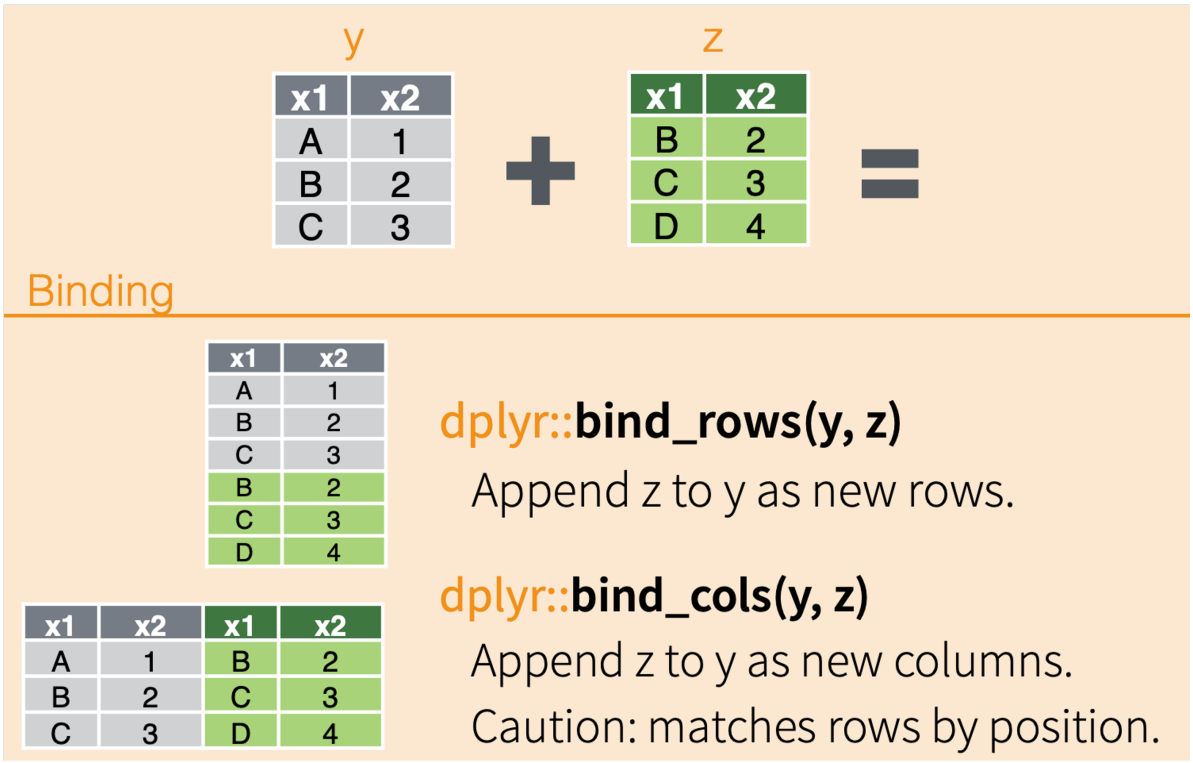
This can happen a couple ways, depending on which data frame is the reference and how much data you want to retain.



Binds

The other way we can combine data sets in through binds. Binds act similarly to gluing datasets together.

They don't match up data based on unique identifiers; instead they match up data by column name (`bind_rows`) or row position (`bind_cols`)



How should we go about combining our three data sets? Come up with a plan that you think will work.

Task 1: Combine Our Data

Step 1

Our first step in to merge the old collar data with the new data about those old collars.

```
# Use a left join

# Full join would accomplish the same thing in this case because we don't have any missing

old_collars_combined <- collars %>%
  left_join(old_collars_new_data, by = c("collar_id", "maker"))
```

If we take a look our new data frame, we should hopefully see that the values for `antenna_length` and `weight` have been matched up with their respective collar id.

Could we have used another tactic to combine these two datasets? What would the pros and cons be?

Step 2

Now we need to add the new collars to our dataset. What is our best method for combining?

```
# Add new collars
full <- bind_rows(old_collars_combined, new_collars)
```

Let's take a look at our new data frame! Have we correctly accomplished our first task?

Saving our New Data

Now that we've accomplished our first task, we are going to want to use this combined dataset to accomplish our next task, which is using a classification algorithm to help us predict which maker made the mystery collars.

To save our data as a .csv file that we can use in another analysis, we are going to use a function that exports the dataset (`write_csv`) instead of importing it (`read_csv`).

The `write_csv` function requires the name of the dataframe to export as the first argument and the name of the file we want to create as the second argument.

```
# Save our new data
write_csv(full, "data/all_collar_data.csv")
```

If we look over in our Files tab, you should see your new .csv file!