DA4: Reshaping data

First let's load the requisite packages and some data (surveys).

In [ ]:
```
# A Python data analysis and manipulation tool
import pandas as pd

# Python equivalent of `ggplot2`
from plotnine import *

# If using seaborn for plotting
import seaborn as sns
import matplotlib.pyplot as plt

surveys_join = pd.read_csv("data/surveys_join.csv")
```

*Now we use the **.pivot()** method to pivot wider*

In [ ]:
```
surveys_wide = surveys_join.pivot(
    index = "record_id",
    columns = "year",
    values = "weight"
)


surveys_wide.head()
```

Let's tidy up those headers with a yr| prefix (unfortunately this will break record_id so there's an extra step to fix it)._

In [ ]:
```
# Add 'yr_' prefix to year columns
surveys_wide.columns = [f"yr_{col}" for col in surveys_wide.columns]

# Reset index to make 'record_id' a column again
surveys_wide = surveys_wide.reset_index()

surveys_wide.head()
```

Example usage: Explore any possible relationship between 2021 and 2022 using plotnine aka why did we do all of this processing?

In [ ]:
```
# Colour the points tomato red - because why not? Demonstrates setting parameters in levels.

p = (ggplot(surveys_wide, aes(x = "yr_2021", y = "yr_2022")) +
    geom_point(colour="tomato"))

p.show()
```

*For reasons unknown, the opposite to **.pivot()** is **.melt()**. In the interests of moving forward, let's just use it.*

In [ ]:
```
surveys_long = surveys_wide.melt(
    id_vars = "record_id",           # columns to keep fixed
    var_name = "year",               # name of the new 'year' column
    value_name = "weight"            # name of the new 'weight' column
)

surveys_long.head()
```

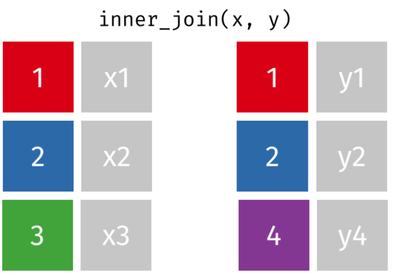
Let's fix those year column values back to numbers by substituting the string and then casting to integer.

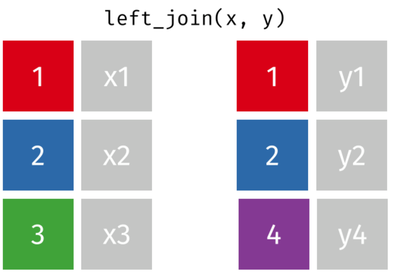
In [ ]:
```
surveys_long["year"] = surveys_long["year"].str.replace("yr_", "").astype(int)

surveys_long.head()
```

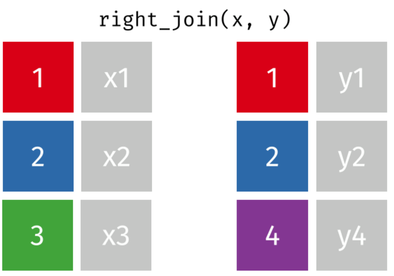
Much better !

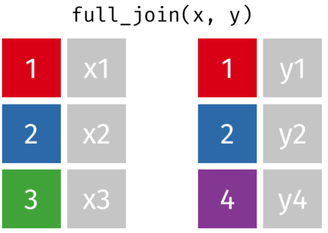
DA4: Combining data

Different types of joins









Let's do a simple Left join to illustrate how to do this in Python

Then when you work through the chapter you can try other joins.

In [ ]:
```
# A Python data analysis and manipulation tool
import pandas as pd

# Python equivalent of `ggplot2`
from plotnine import *

surveys = pd.read_csv("data/surveys.csv")

# Read in a file that can be used to translate plotIDs into plot-types
plot_types = pd.read_csv("data/plots.csv")

plot_types # list the plot types
```

We use the **.merge()** method with data coming from **surveys** and **plot_types**, the how being a **left-join** and the join being on the common field of **plot_id**.

In [ ]:
```
surveys_left = pd.merge(surveys, plot_types, how = "left", on = "plot_id")

surveys_left
```

Let's use our newly derived data structure to create a box-plot of plot_type vs. hindfoot_length that names the type of enclosure rather than using an id number.

In [ ]:
```
p = (ggplot(surveys_left, aes(x = "plot_type", y = "hindfoot_length")) +
  geom_boxplot())

p.show()
```

Now work through the material of chapter 13 and try the exercises at the end.

In [ ]:
```
```