

# Logistic regression

# Overview

Information on the final project

Brief mention: Visualizing linear models using ggplot

Logistic regression

If there is time: Joining data frames

# Final projects!

The final project is a **5-8 page** R Markdown report where you analyze your own data to address a question that you find interesting

- It's a chance to practice everything you've learned in class!

**The goal of the project is to present a clear and compelling analyses of data showing a few interesting results!**

A few sources for data sets are listed on Canvas

- You can use data you collect as well. If you use data for another class your work must be unique for each class.

# Final projects!

An R Markdown template describing sections in the project is on the class GitHub site.

- An R function you can run to download this template is listed on Canvas.

A key challenge is going to be to fit your analyses into 5-8 pages:

- You can include an appendix with additional code that does not count against your 5-8 pages
  - E.g., you can include functions in your appendix and then just call them in the body of your report

Project is due at 11pm on Sunday December 11<sup>th</sup>

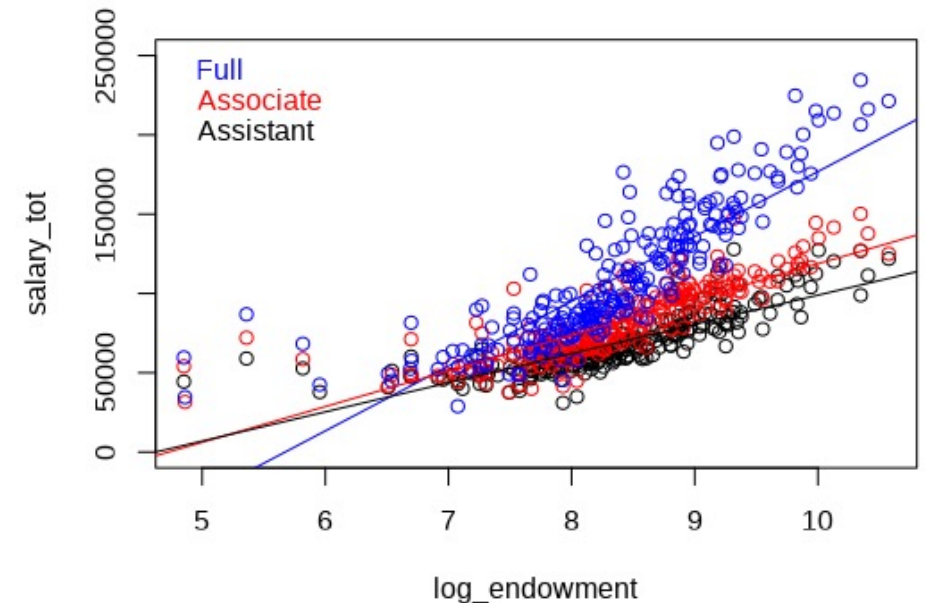
- i.e., the day before the start of reading period

# Plotting multiple regression models with ggplot

So far we have plotted our multiple regression models using base R graphics

This was useful for seeing the relationship between how R fits linear models, and what these models represent

However, if you want an easier/prettier way to visualize linear models, we can use ggplot!



Let's try it in R!

# Logistic regression

# Logistic regression

In **logistic regression** we try to predict whether a case belongs to one of two categories

- Does a case belong to category  $a$  or category  $b$ ?
- Example: can we predict if a faculty member is an Assistant or Full professor based on the salary level?

Making predictions for a categorical variable is called **classification**

- The field of Machine Learning has developed many classification methods

In logistic regression we build a conditional probability model:

- $\Pr(\text{Class} = a \mid x)$
- $\Pr(\text{Assistant Professor} \mid \text{salary} = \$60,000)$

# Logistic regression

**Question:** could we use linear regression to make these predictions?

$$P(Y = a \mid x_1) = \beta_0 + \beta_1 x_1$$

**Problem:** we could get negative probabilities and probabilities greater than 1!



# Logistic regression

**Question:** what if we transformed the probability to odds?

$$\frac{P(Y = a \mid x_1)}{P(Y = b \mid x_1)}$$

**Question:** what are the range of values odds can take on?

**A:** 0 to  $\infty$

# Logistic regression

Instead, we model the log odds as a linear function of our predictors

$$\log\left(\frac{P(Y=a|x)}{1-P(Y=a|x)}\right)$$

log-odds or logit



This scales values in the range of  $[0, 1]$  to values in the range of  $(-\infty, \infty)$

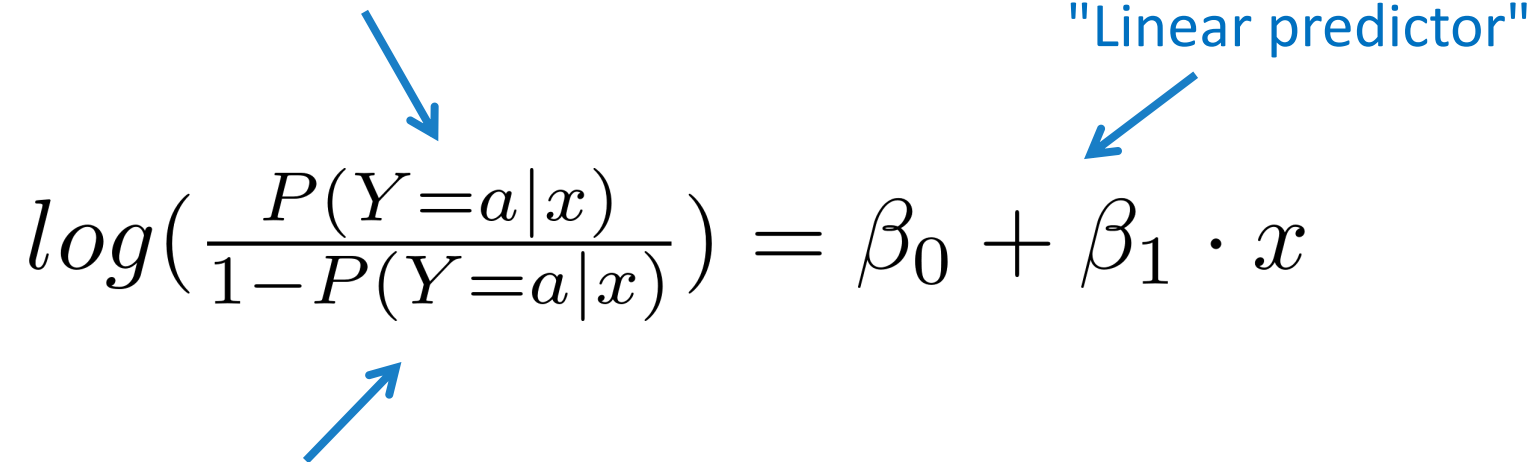
# Generalized linear models

**Generalized linear models** use a linear combinations of predictors to predict ***a function of the mean***

If  $Y$  is a binary response variable ( $Y = 0$  or  $1$ )

$P(Y = 1|x)$  is the mean of  $Y$

"Linear predictor"


$$\log\left(\frac{P(Y=a|x)}{1-P(Y=a|x)}\right) = \beta_0 + \beta_1 \cdot x$$

The logit function (log-odds) is a "link function" that links the mean to the linear predictor

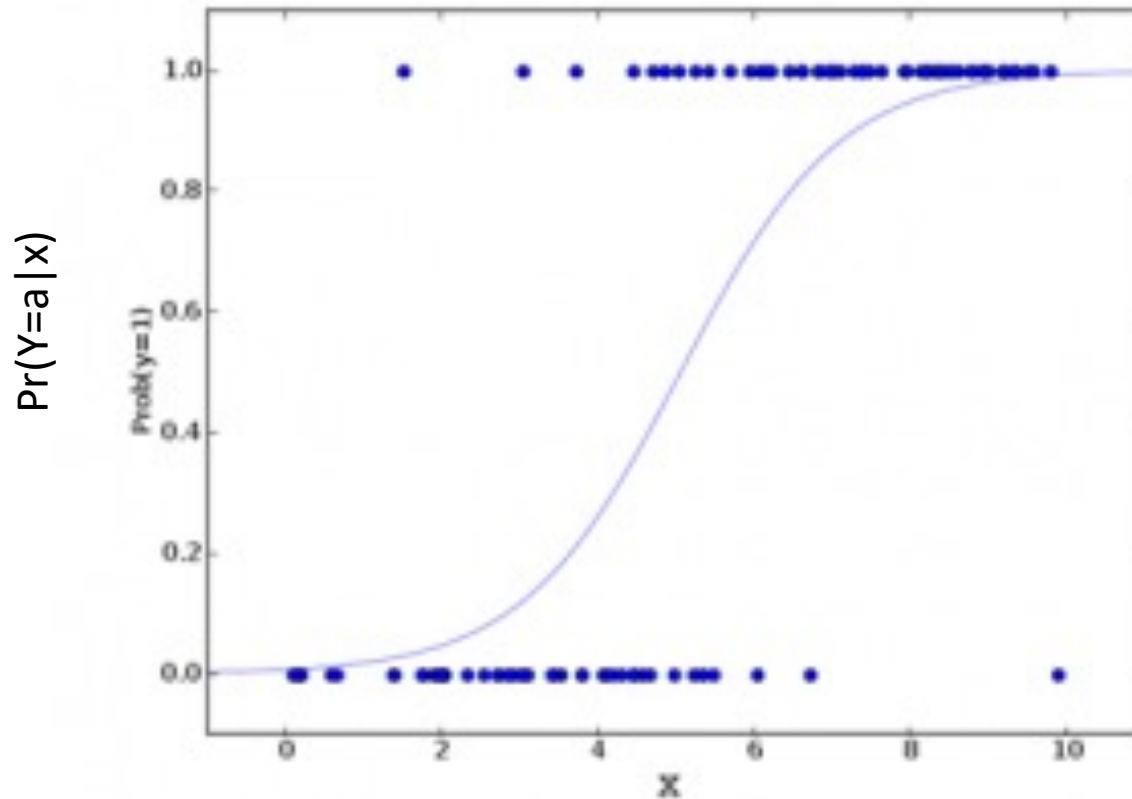
# Logistic function

$$\log\left(\frac{P(Y=a|x)}{1-P(Y=a|x)}\right) = \beta_0 + \beta_1 \cdot x$$

Solving for  $P(Y = a | x)$  we get the "inverse link" function, which in the case of logistic regression is called a ***logistic function***

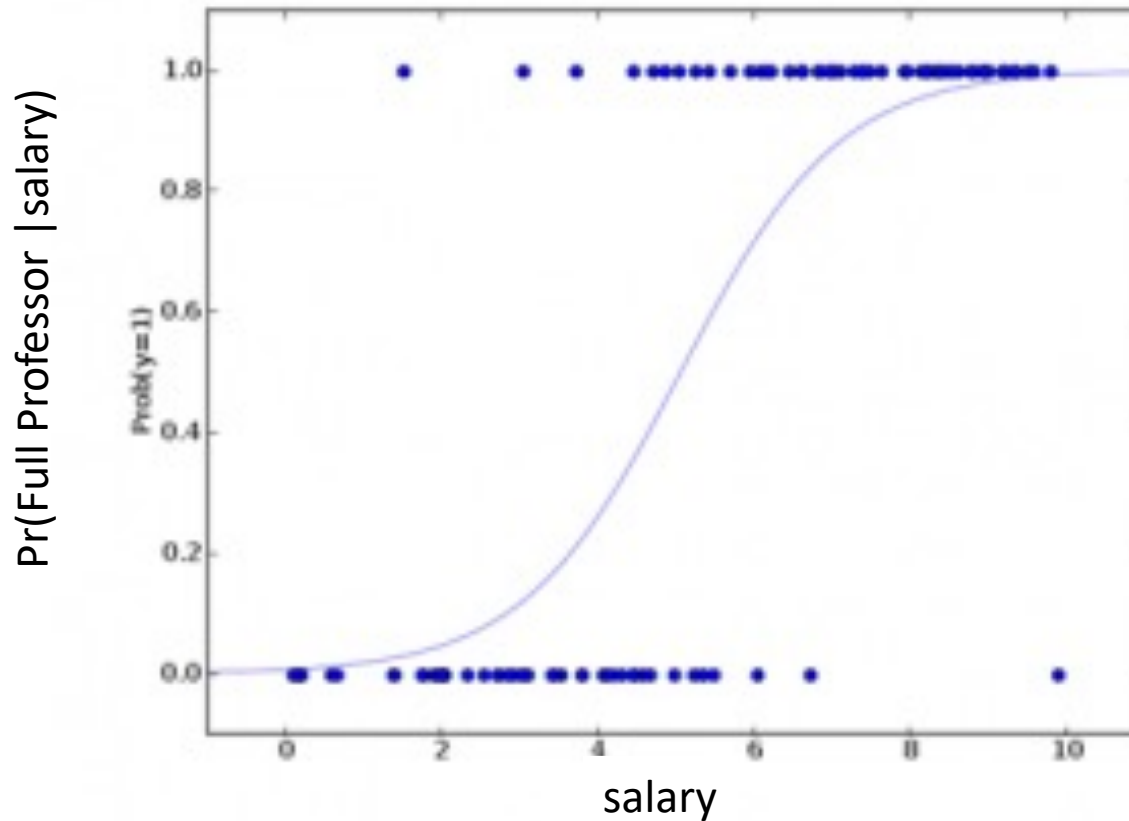
$$P(Y = a|x) = \frac{\exp(\beta_0 + \beta_1 \cdot x_1)}{1 + \exp(\beta_0 + \beta_1 \cdot x_1)} = \frac{e^{\beta_0 + \beta_1 \cdot x_1}}{1 + e^{\beta_0 + \beta_1 \cdot x_1}}$$

# Plotting the logistic function



$$P(Y = a|x_1) = \frac{e^{\beta_0 + \beta_1 \cdot x_1}}{1 + e^{\beta_0 + \beta_1 \cdot x_1}}$$

# Plotting the logistic function



$$P(\text{Full Professor} \mid \text{salary}) = \frac{e^{\hat{\beta}_0 + \hat{\beta}_1 \cdot \text{salary}}}{1 + e^{\hat{\beta}_0 + \hat{\beta}_1 \cdot \text{salary}}}$$

# Multivariate logistic regression

We can easily extend our logistic regression model to include multiple explanatory variables

$$\log\left(\frac{P(Y=a|x)}{1-P(Y=a|x)}\right) = \hat{\beta}_0 + \hat{\beta}_1 \cdot x_1 + \hat{\beta}_2 \cdot x_2 + \dots + \hat{\beta}_k \cdot x_k$$

We can also use categorical predictors via dummy variable encoding as we did for regular multiple linear regression

# Interpreting categorical predictors

When using a categorical predictor,  $x_2$ , in a logistic regression model, the exponential of the regression coefficient  $e^{\hat{\beta}_2}$  is the **odds ratio**

- Tells us how many times greater the odds are when  $x_2 = 1$  vs. when  $x_2 = 0$

$$\log\left(\frac{P(Y=a|x_1, x_2)}{1-P(Y=a|x_1, x_2)}\right) = \hat{\beta}_0 + \hat{\beta}_1 \cdot x_1 + \hat{\beta}_2 \cdot x_2$$

Dummy variable

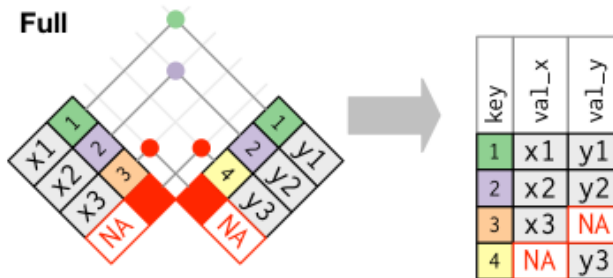
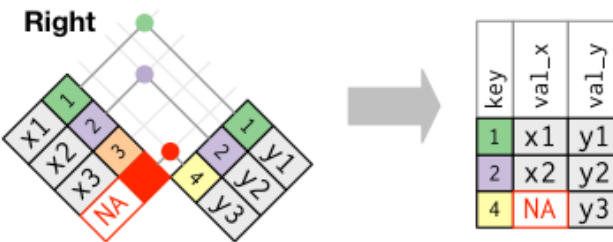
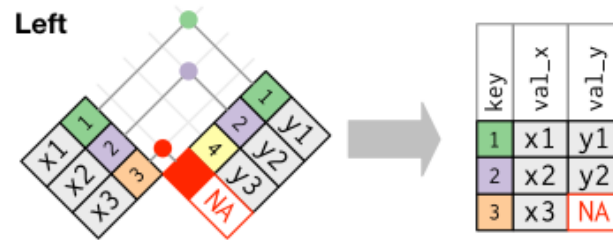


$$\begin{array}{l} \text{If } x_2 = 1 \\ \frac{P(Y|x_1, x_2=1)}{1-P(Y|x_1, x_2=1)} = \frac{e^{\hat{\beta}_0} e^{\hat{\beta}_1 \cdot x_1} e^{\hat{\beta}_2}}{e^{\hat{\beta}_0} e^{\hat{\beta}_1 \cdot x_1}} = e^{\hat{\beta}_2} \end{array}$$
$$\begin{array}{l} \text{If } x_2 = 0 \\ \frac{P(Y|x_1, x_2=0)}{1-P(Y|x_1, x_2=0)} = e^{\hat{\beta}_0} e^{\hat{\beta}_1 \cdot x_1} \end{array}$$



Let's look at this in R...

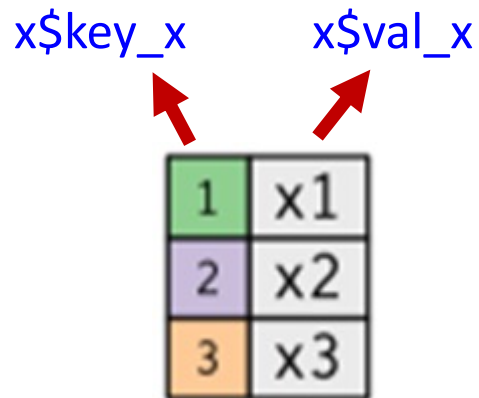
# Joining data frames



# Left and right tables

Suppose we have two data frames called *x* and *y*

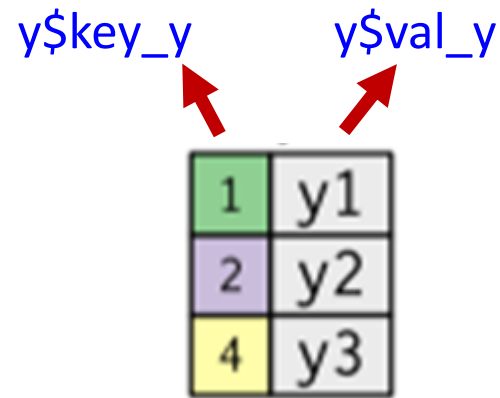
- *x* has two variables called *key\_x*, and *val\_x*
- *y* has two variables called *key\_y* and *val\_y*



A diagram of Data frame x. It consists of a 3x2 grid of cells. The first column contains the values 1, 2, and 3, each in a colored box (green, purple, and orange respectively). The second column contains the values x1, x2, and x3, each in a grey box. Above the first column, the label x\$key\_x has a red arrow pointing to the first cell. Above the second column, the label x\$val\_x has a red arrow pointing to the first cell.

|   |    |
|---|----|
| 1 | x1 |
| 2 | x2 |
| 3 | x3 |

**Data frame x**



A diagram of Data frame y. It consists of a 3x2 grid of cells. The first column contains the values 1, 2, and 4, each in a colored box (green, purple, and yellow respectively). The second column contains the values y1, y2, and y3, each in a grey box. Above the first column, the label y\$key\_y has a red arrow pointing to the first cell. Above the second column, the label y\$val\_y has a red arrow pointing to the first cell.

|   |    |
|---|----|
| 1 | y1 |
| 2 | y2 |
| 4 | y3 |

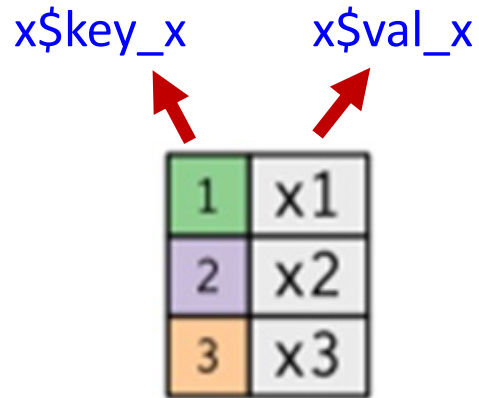
**Data frame y**

`SDS230:download_data('x_y_join.rda')`

# Left and right tables

Suppose we have two data frames called x and y

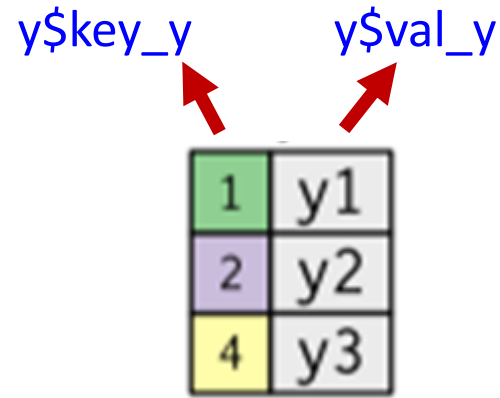
- x have two variables called `key_x`, and `val_x`
- y has two variables called `key_y` and `val_y`



A diagram of Data frame x. It consists of a 3x2 grid of cells. The first column contains the values 1, 2, and 3, each in a colored box (green, purple, and orange respectively). The second column contains the values x1, x2, and x3, each in a grey box. Above the first column, the text 'x\$key\_x' has a red arrow pointing to the first cell. Above the second column, the text 'x\$val\_x' has a red arrow pointing to the first cell.

|   |    |
|---|----|
| 1 | x1 |
| 2 | x2 |
| 3 | x3 |

**Data frame x**



A diagram of Data frame y. It consists of a 3x2 grid of cells. The first column contains the values 1, 2, and 4, each in a colored box (green, purple, and yellow respectively). The second column contains the values y1, y2, and y3, each in a grey box. Above the first column, the text 'y\$key\_y' has a red arrow pointing to the first cell. Above the second column, the text 'y\$val\_y' has a red arrow pointing to the first cell.

|   |    |
|---|----|
| 1 | y1 |
| 2 | y2 |
| 4 | y3 |

**Data frame y**

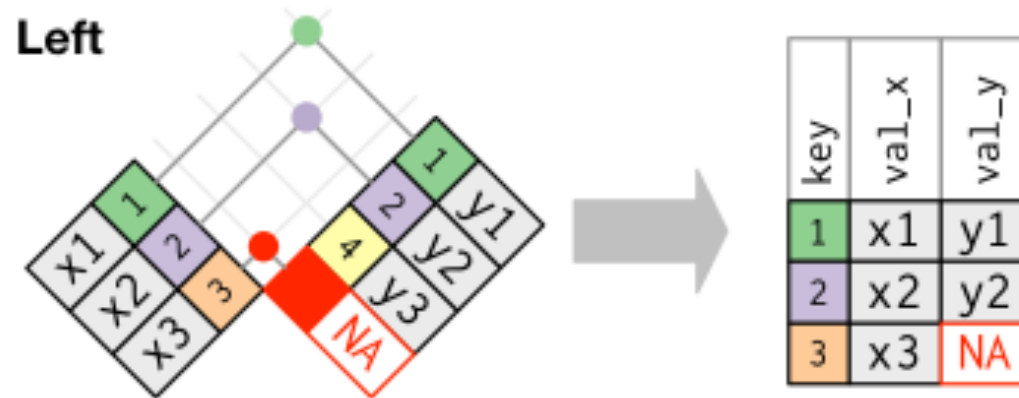
Joins have the general form:

```
join(x, y, by = c("key_x" = "key_y"))
```

# Left joins

**Left joins** keep all rows in the left table.

Data from right table is added when there is a matching key, otherwise NA is added.

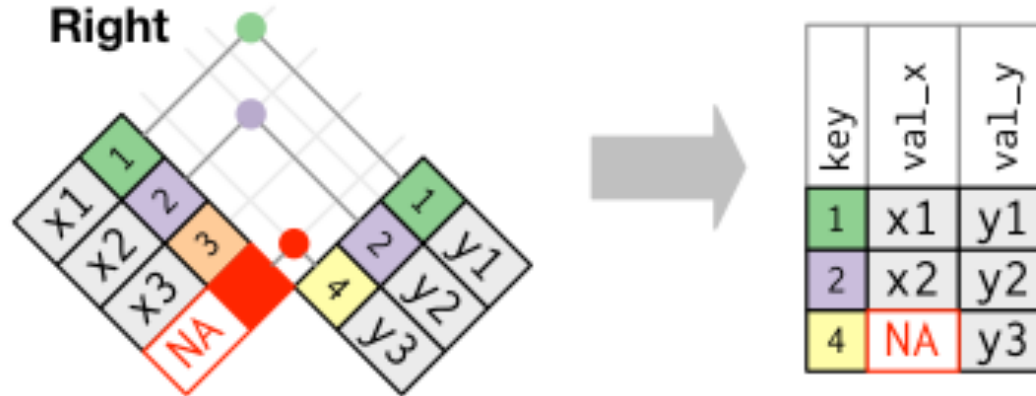


```
> left_join(x, y, by = c("key_x" = "key_y"))
```

# Right joins

**Right joins** keep all rows in the right table.

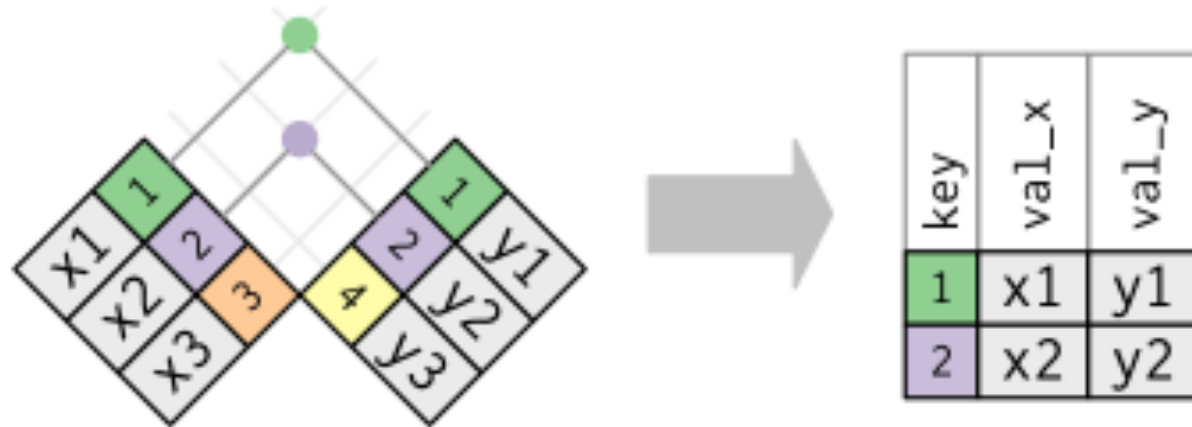
Data from left table added when there is a matching key, otherwise NA as added.



```
> right_join(x, y, by = c("key_x" = "key_y"))
```

# Inner joins

**Inner joins** only keep rows in which there are matches between the keys in both tables.

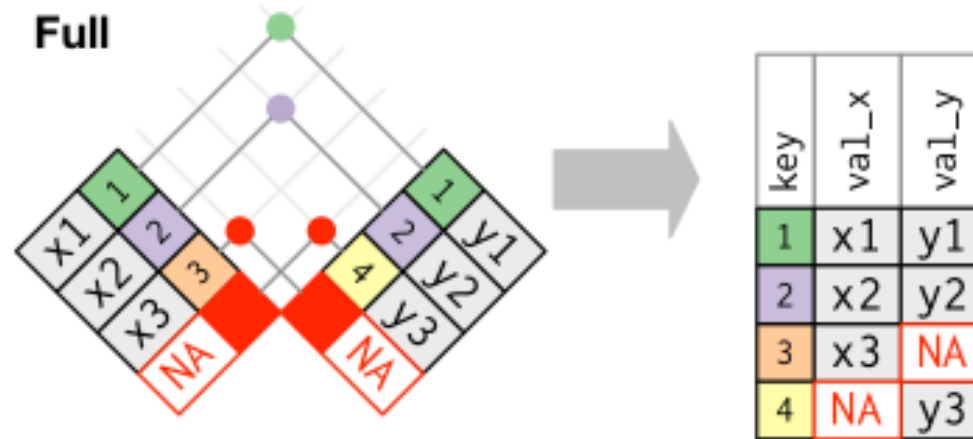


```
> inner_join(x, y, by = c("key_x" = "key_y"))
```

# Full joins

**Full joins** keep all rows in both table.

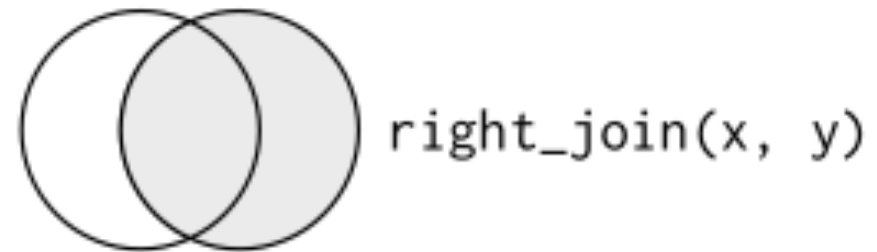
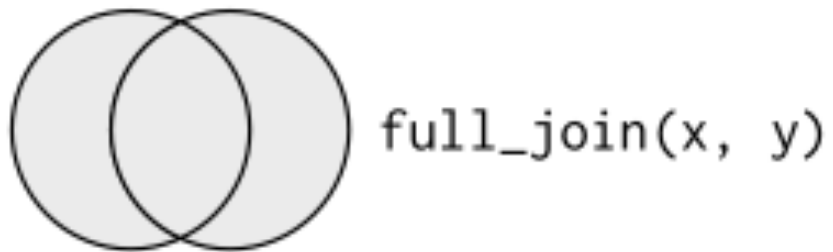
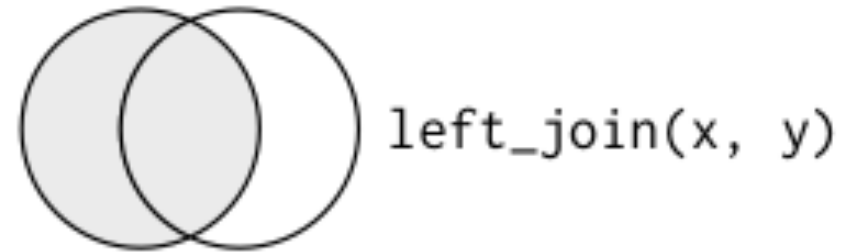
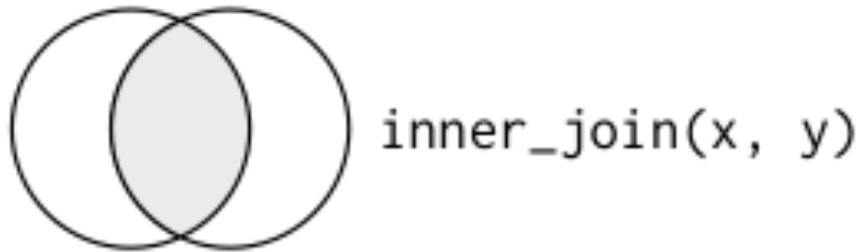
NAs are added where there are no matches.



> `full_join(x, y, by = c("key_x" = "key_y"))`



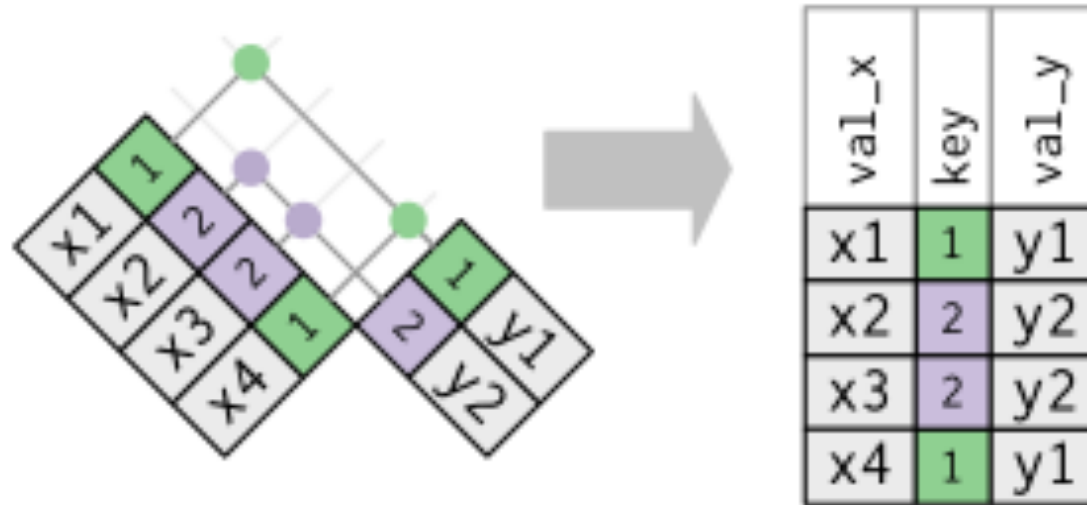
# Summary



# Duplicate keys

Duplicate keys are useful if there is a many-to-one relationship

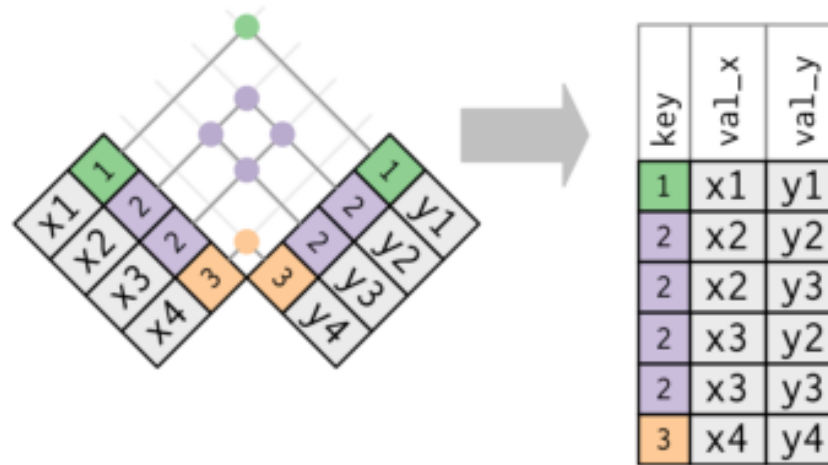
- e.g., duplicates are useful in the left table when doing a left join



# Duplicate keys

If both tables have duplicate keys you get all possible combinations of joined values (Cartesian product).

- **This is usually an error!**



Always check the output size after you join a table because even if there is not a syntax error you might not get the table you are expecting!

- You can check how many rows a data frame has using the `nrow()` function

# Duplicate keys

To deal with duplicate keys in both tables, we can join the tables using multiple keys in order to make sure that each row is uniquely specified.

We can do this using the syntax:

```
join(x2, y2, by = c("key1_x" = "key1_y", "key2_x" = "key2_y"))
```

# Duplicate keys

```
> x2 <- data.frame(key1_x = c(1, 2, 2),  
  key2_x = c("a", "a", "b"),  
  val_x = c("y1", "y2", "y3"))
```

```
> y2 <- data.frame(key1_y = c(1, 2, 2, 3, 3),  
  key2_y = c("a", "a", "b", "a", "b"),  
  val_y = c("y1", "y2", "y3", "y4", "y5"))
```

```
> left_join(x2, y2, c("key1_x" = "key1_y"))
```

```
> left_join(x2, y2, c("key1_x" = "key1_y", "key2_x" = "key2_y"))
```

# Structured Query Language

Having multiple tables that can be joined together is common in Relational Database Systems (RDBS).

- A common language used by RDBS is Structured Query Language (SQL)

| dplyr                                   | SQL   |
|---|---|
| <code>inner_join(x, y, by = "z")</code> | <code>SELECT * FROM x INNER JOIN y USING (z)</code>       |
| <code>left_join(x, y, by = "z")</code>  | <code>SELECT * FROM x LEFT OUTER JOIN y USING (z)</code>  |
| <code>right_join(x, y, by = "z")</code> | <code>SELECT * FROM x RIGHT OUTER JOIN y USING (z)</code> |
| <code>full_join(x, y, by = "z")</code>  | <code>SELECT * FROM x FULL OUTER JOIN y USING (z)</code>  |

Let's try it in R...