ANOVA continued, and data cleaning

Overview

Review and continuation of ANOVAs

- Unbalanced data
- Repeated measures/block designs and random effects models

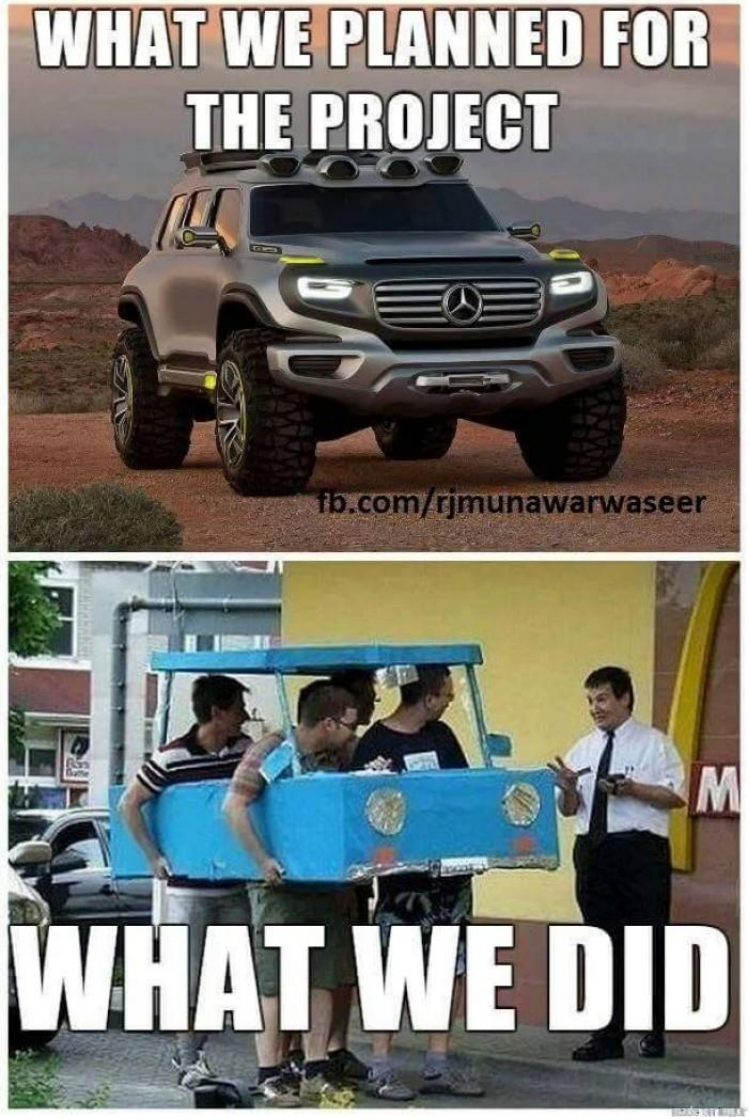
Text manipulation with stringr

Pivoting data with tidyr

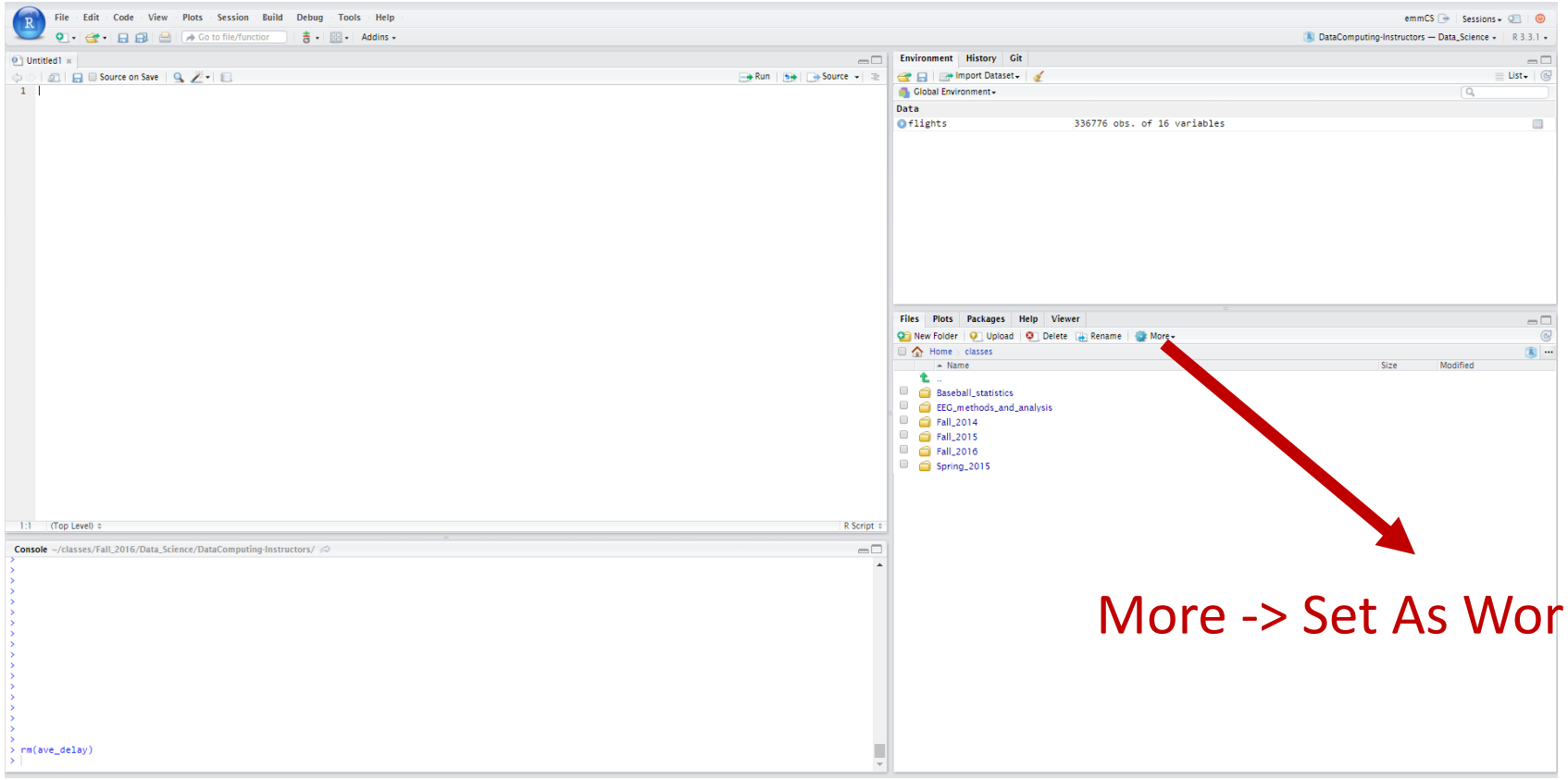
If there is time: joining data

How are final projects going?

Any questions about the final project?



Setting your working directory



More -> Set As Working Directory

1. In the files tab, navigate to the directory that contains your data and final project .Rmd file
2. Click More -> Set As Working Directory
3. You should note be able to use read.csv() to load data that is in you working directory

ANOVA review

An Analysis of Variance (ANOVA) can be viewed as:

- A hypothesis test comparing multiple means
- A model for predicting means from categorical variables

In a **factorial ANOVA**, we model the response variable y as a function of **more than one** categorical predictor

For a two-way ANOVA we have:

$$y_{ijk} = \mu + \alpha_j + \beta_k + \gamma_{jk} + \varepsilon_{ijk}$$

Random error for the ijk^{th} data point

i^{th} response when:
- factor A has level j
- Factor B has level k

Overall mean

Main effect for factor A at level j

Main effect for factor B at level k

Specific interaction for j^{th} level of A and k^{th} level of B

Two-way ANOVA hypotheses

<u>Main effect for A</u>

H_0: $\alpha_1 = \alpha_2 = \ldots = \alpha_J = 0$

H_A: $\alpha_j \neq 0$ for some j

<u>Main effect for B</u>

H_0: $\beta_1 = \beta_2 = \ldots = \beta_K = 0$

H_A: $\beta_k \neq 0$ for some k

<u>Interaction effect:</u>

H_0: All $\gamma_{jk} = 0$

H_A: $\gamma_{jk} \neq 0$ for some j, k

Where:

α_j: is the "effect" for factor A at level j

β_k: is the "effect" for factor B at level k

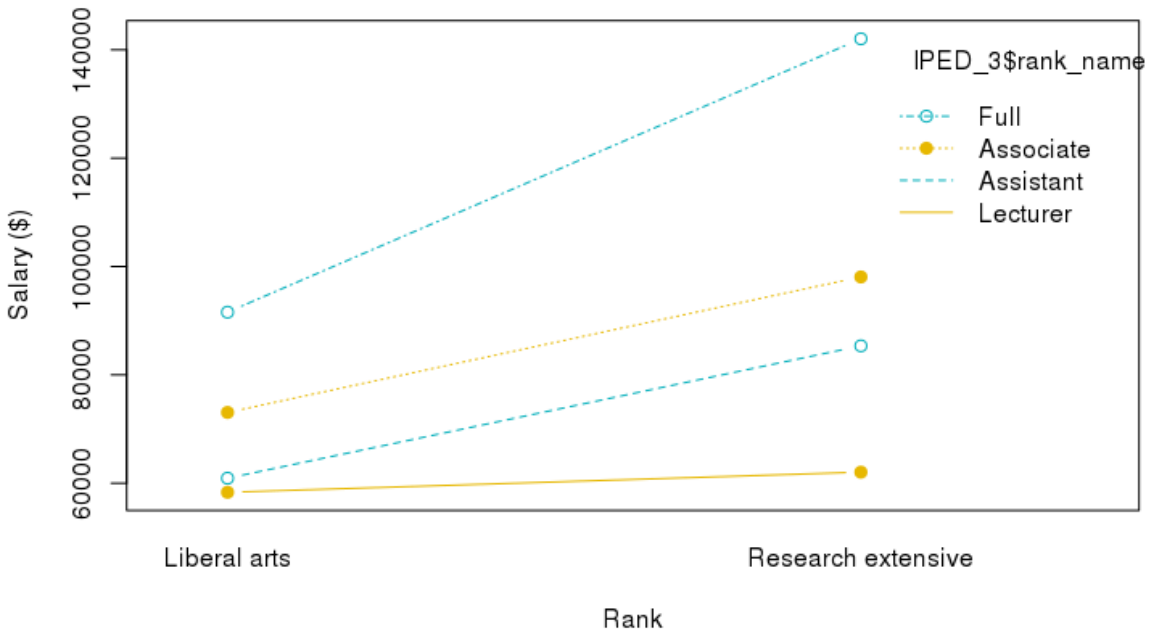
γ_{jk} : is the interaction between level j of factor A, and level k of factor B.

$$y_{ijk} = \mu + \alpha_j + \beta_k + \gamma_{jk} + \varepsilon_{ijk}$$

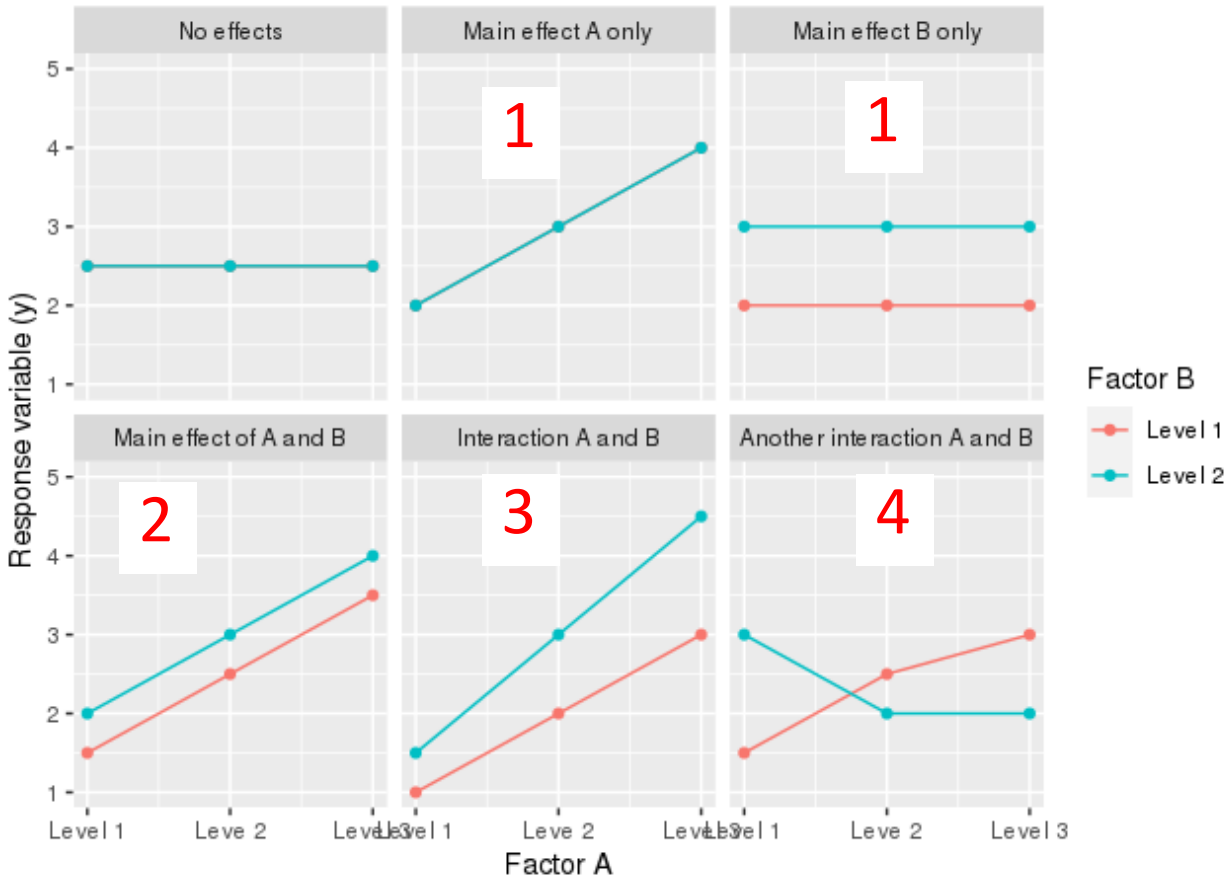
Interaction plots

Interaction plots can help us visualize main effects and interactions

- Plot the levels of one of the factors on the x-axis
- Plot the levels of the other factor as separate lines

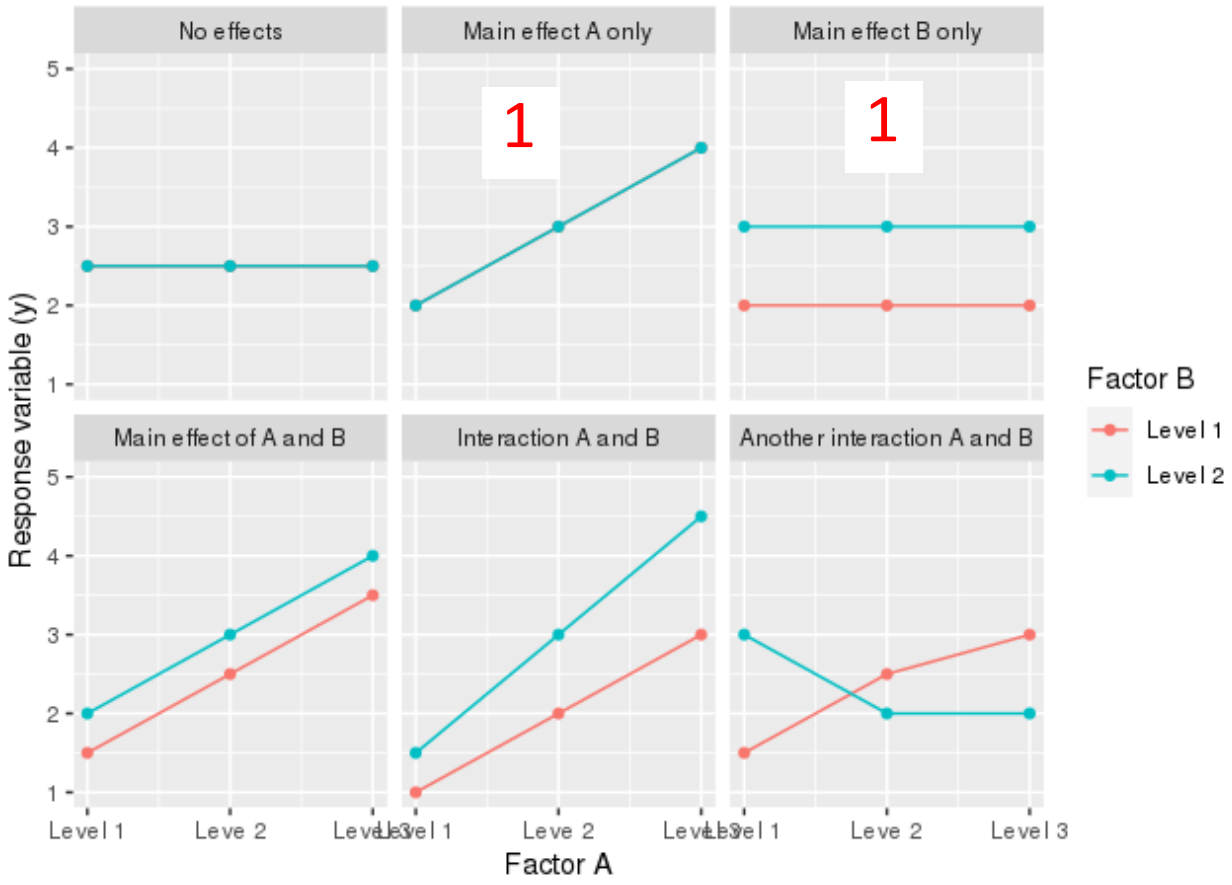


Interpreting interaction through interaction plots



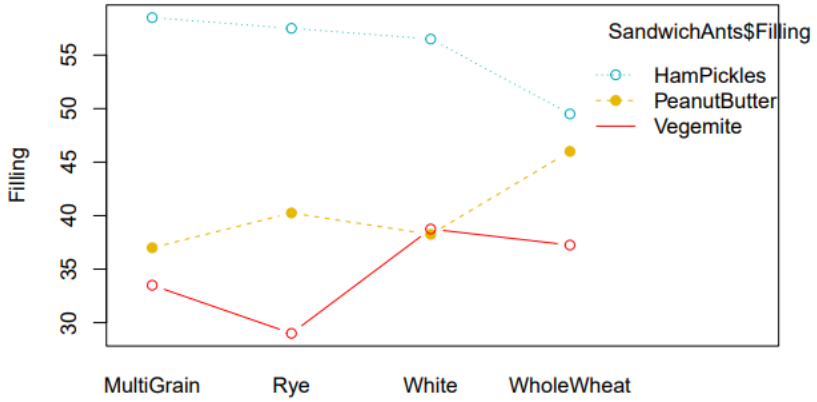
What are examples we have seen in class of the interactions in plots 1, 2, 3 and 4?

Interpreting interaction through interaction plots



What are examples we have seen in class of 1 (main effect only in one factor)?

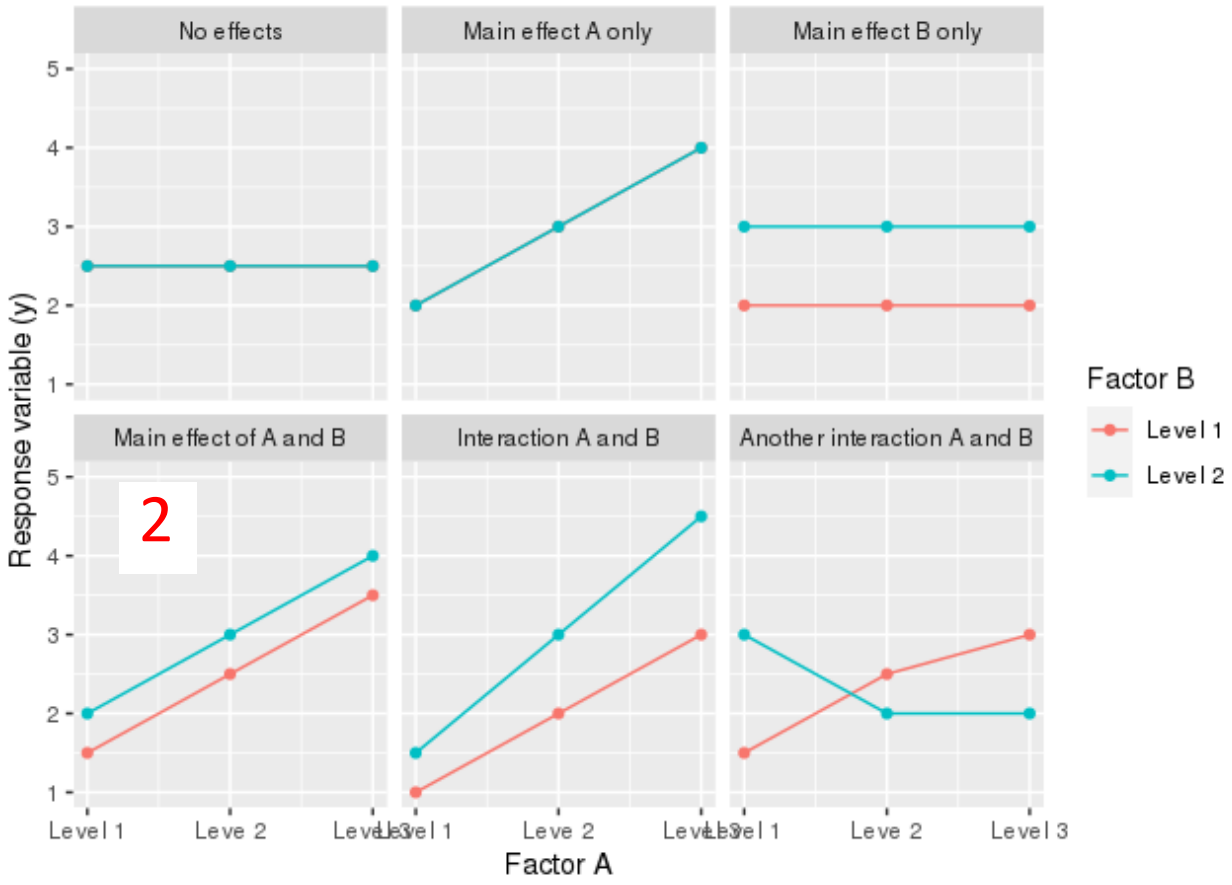
1: Ants sandwich preferences




```
##                Df Sum Sq Mean Sq F value    Pr(>F)
## Bread           3   40.5   13.50  0.0754 0.9728619
## Filling         2 3720.5 1860.25 10.3860 0.0002748 ***
## Bread:Filling   6  577.0   96.17  0.5369 0.7765447
## Residuals      36 6448.0  179.11
```

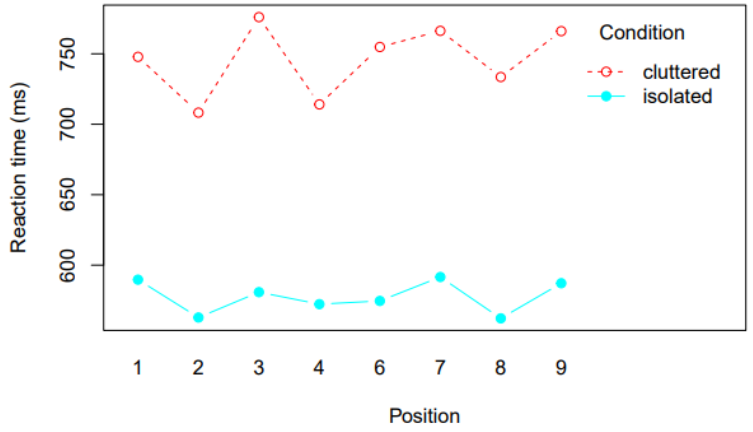
$$y_{ijk} = \mu + \beta_k + \varepsilon_{ijk}$$

Interpreting interaction through interaction plots



What are examples we have seen in class of 2 (main effect in both factors, no interaction)?

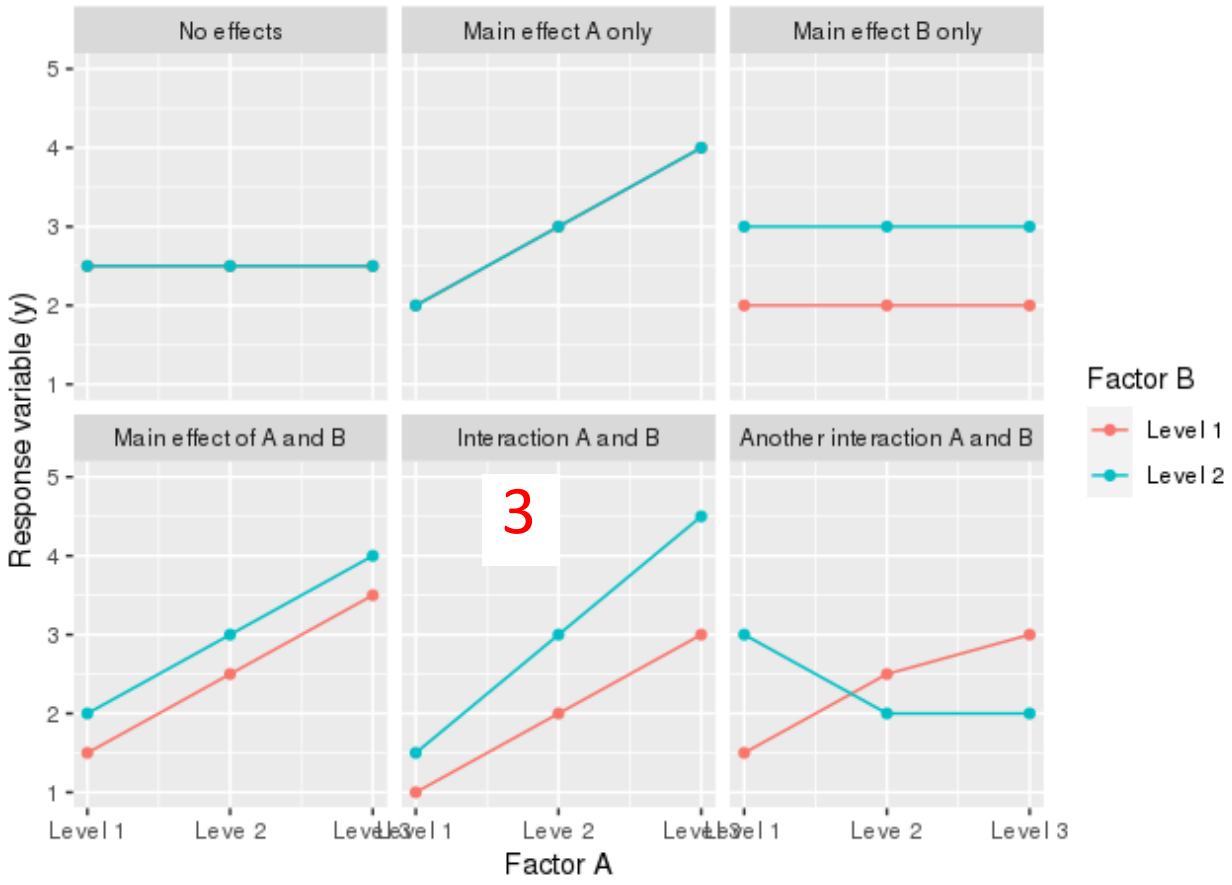
2: Pop-out attention




```
##                     Df Sum Sq Mean Sq  F value              Pr(>F)
## condition            1  77.63   77.63 1457.276 < 0.0000000000000002 ***
## position             7   2.83    0.40    7.579        0.00000000406 ***
## condition:position   7   0.56    0.08    1.504                0.161
## Residuals         5054 269.25    0.05
```

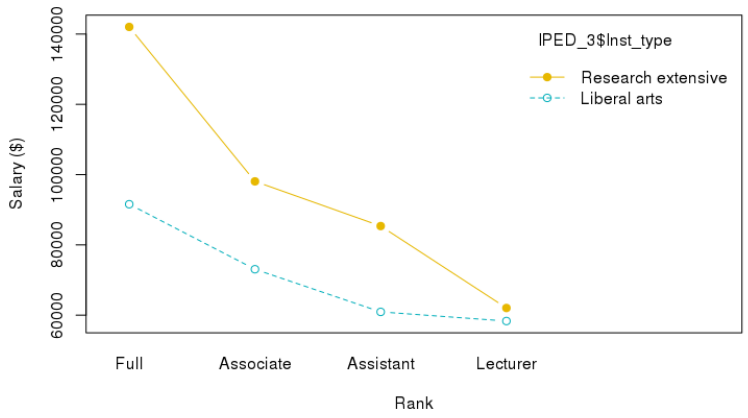
$$y_{ijk} = \mu + \alpha_j + \beta_k + \varepsilon_{ijk}$$

Interpreting interaction through interaction plots



What are examples we have seen in class of 3 (main effects and interaction)?

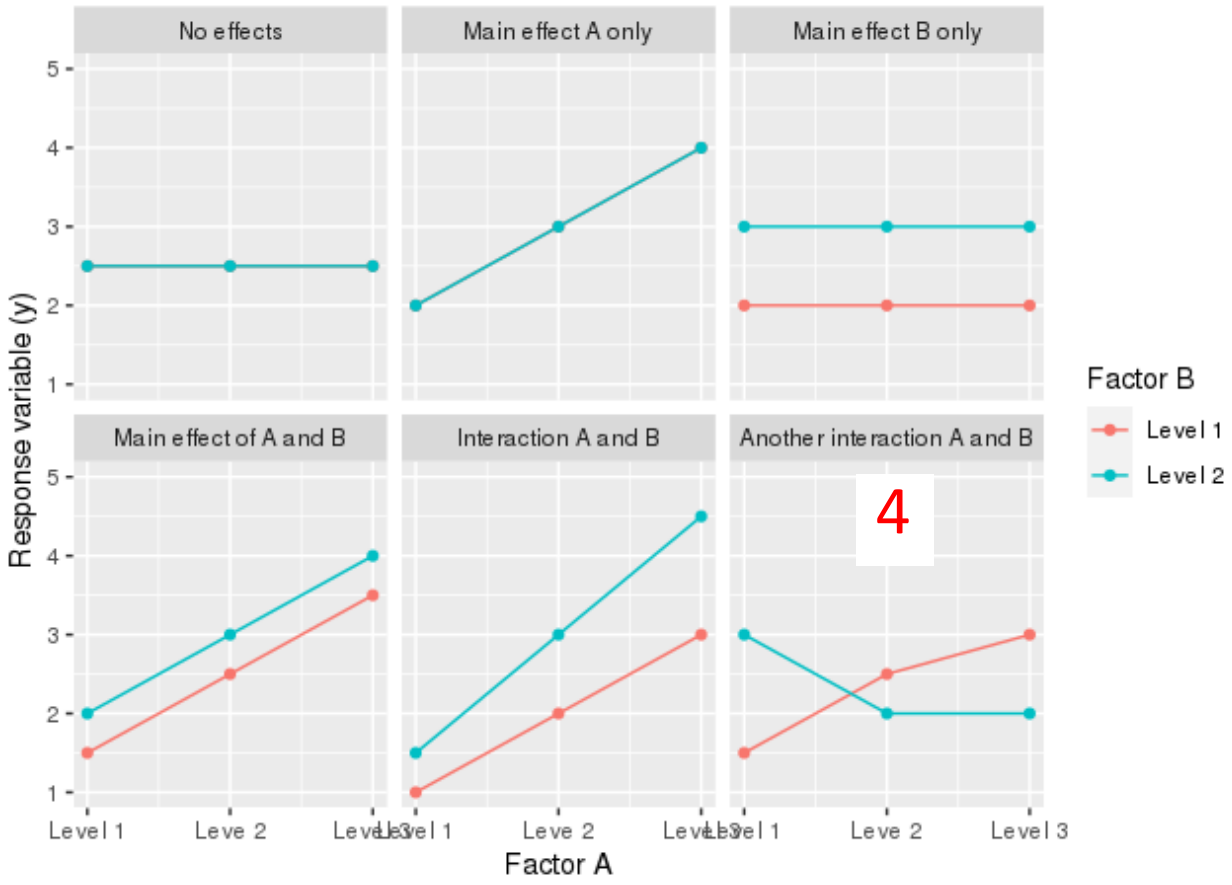
3: Faculty salaries



	Sum Sq	Df	F value	Pr(>F)	
(Intercept)	1752515009149	1	4422.319	< 0.00000000000000022	***
rank_name	116444984138	3	97.946	< 0.00000000000000022	***
Inst_type	223284242856	1	563.438	< 0.00000000000000022	***
rank_name:Inst_type	71691991643	3	60.303	< 0.00000000000000022	***
Residuals	496153389805	1252			

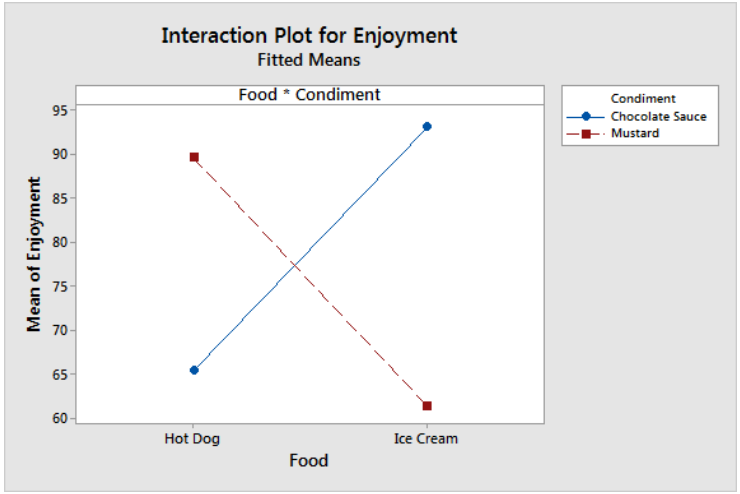
$$y_{ijk} = \mu + \alpha_j + \beta_k + \gamma_{jk} + \varepsilon_{ijk}$$

Interpreting interaction through interaction plots



What are examples we have seen in class of 4 (reverse ordering of effects with interaction)?

4: Foods and condiments




```
> summary(aov(Enjoyment ~ Food*Condiment, data = condiments_food ))
               Df Sum Sq Mean Sq F value               Pr(>F)
Food            1      2       2   0.064              0.80136
Condiment       1    278     278  11.071              0.00135 **
Food:Condiment  1  15696   15696 626.153 < 0.0000000000000002 ***
Residuals      76   1905      25
```

$$y_{ijk} = \mu + \alpha_j + \beta_k + \gamma_{jk} + \varepsilon_{ijk}$$

Complete and balanced designs

Complete factorial design: at least one measurement for each possible combination of factor levels

- E.g., in a two-way ANOVA for factors A and B, if there are K levels for factor A, and J levels for factor B, then there needs to be at least one measurement for each of the KJ levels

Balanced design: the sample size is the same for all combination of factor levels

- E.g., there are the same number of samples in each of the KJ level combinations.
- The computations and interpretations for non-balanced designs are a bit harder.

Unbalanced designs

For unbalanced designs, there are different ways to compute the sum of squares, and hence one can get different p-values

- The problem is analogous to multicollinearity. If two explanatory variables are correlated either can account for the variability in the response data.

Type I sum of squares, (also called sequential sum of squares) the order that terms are entered in the model matters.

- anova(lm(y ~ A*B)) gives different results than using anova(lm(y ~ B*A))
- SS(A) is taken into account before SS(B) is considered etc.

Type III sum of squares, the order that that terms are entered into the model does not matter.

- Car::Anova(lm(y ~ A*B) , type = "III") is the same as car::Anova(lm(y ~ B*A) , type = "III")
- For each factor, SS(A), SS(B), SS(AB) is taken into account after all other factors are added

Repeated measures ANOVA

In a **repeated measures ANOVA**, the same case/observational units are measured at each factor level

Example: Do people prefer chocolate, butterscotch or caramel sauce?

Between subjects experiment: different people rate chocolate, butterscotch or caramel sauce.

- Run a between subjects ANOVA (as we have done before)

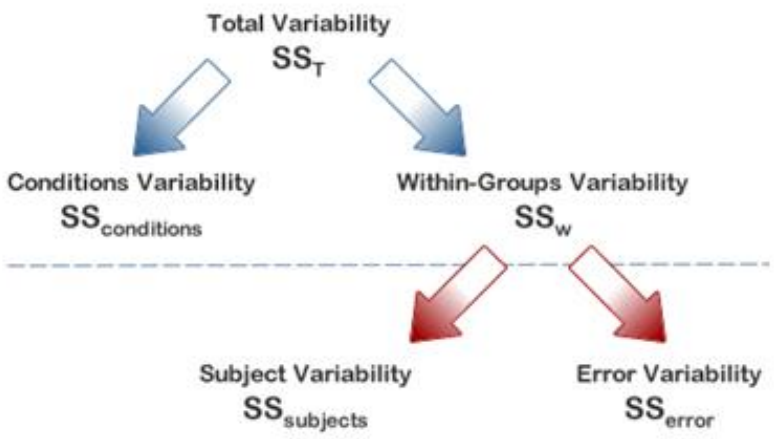
Within subjects experiment: each person in the experiment gives ratings for all three toppings.

- Run a repeated measures ANOVA

Repeated measures ANOVA

The advantages of a repeated measures ANOVA is that we can potentially reduce a lot of the variability between the cases

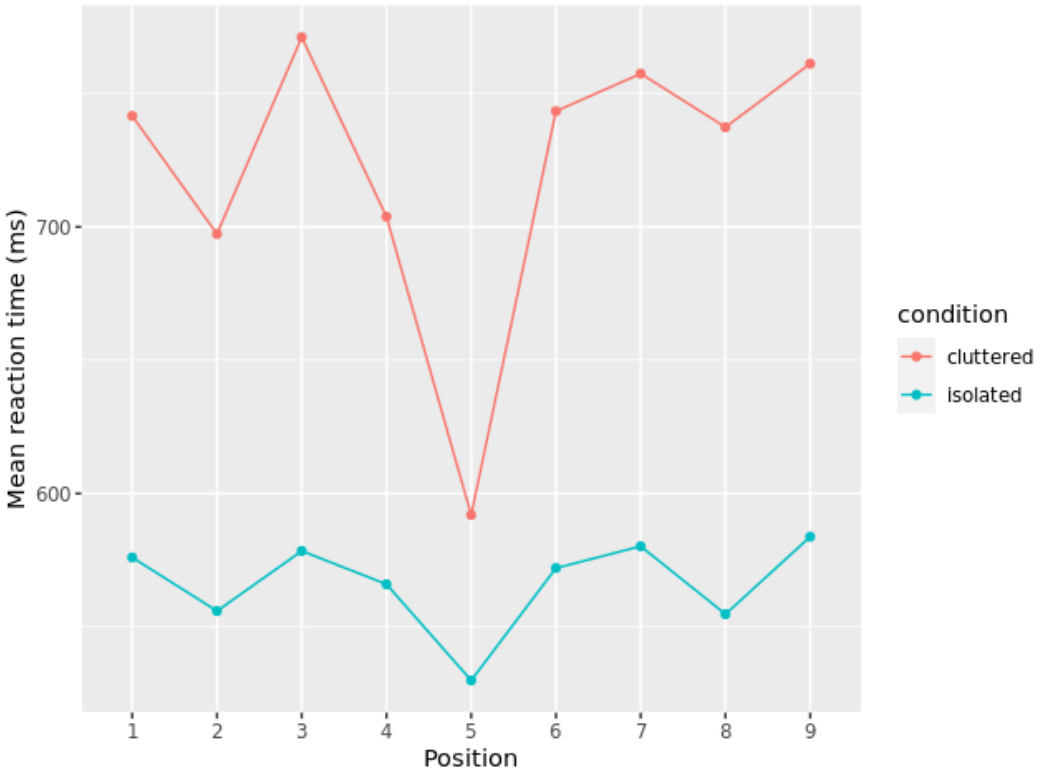
- This is a generalization of a paired t-test to more than two population means

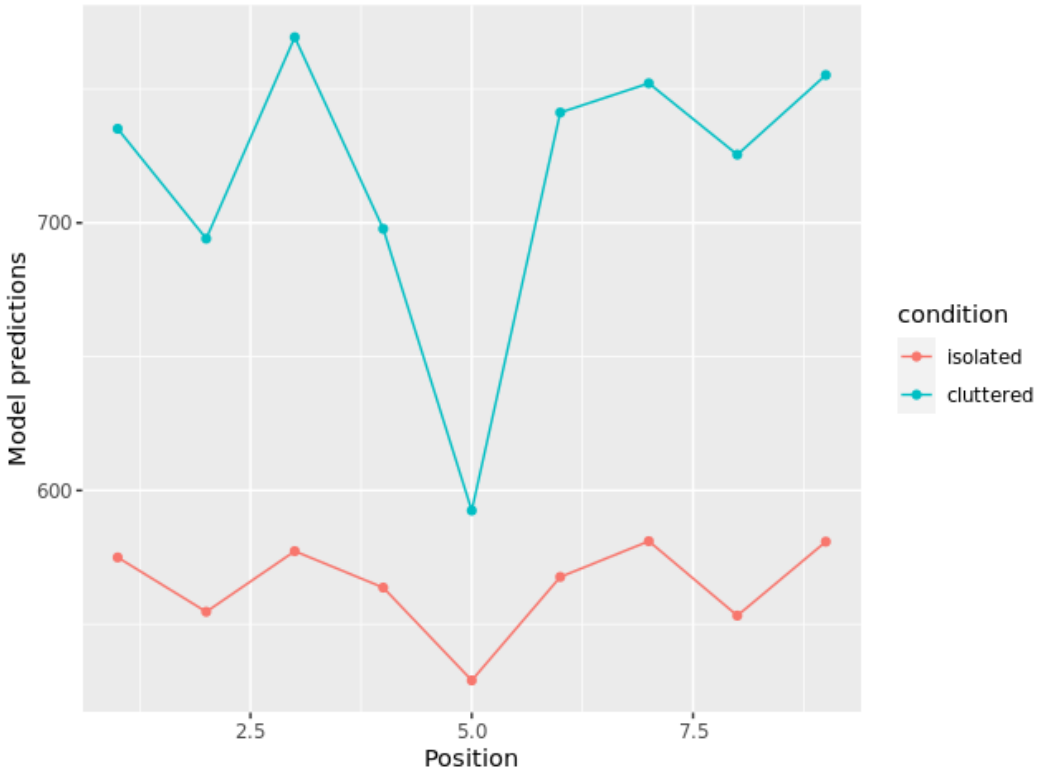


To run a repeated measures ANOVA, we use a factor called ID (or participant, etc.) that has a unique value for each observational unit

```
aov(reaction_time ~ condition * position +  participant,
    data = popout_log_data)
```

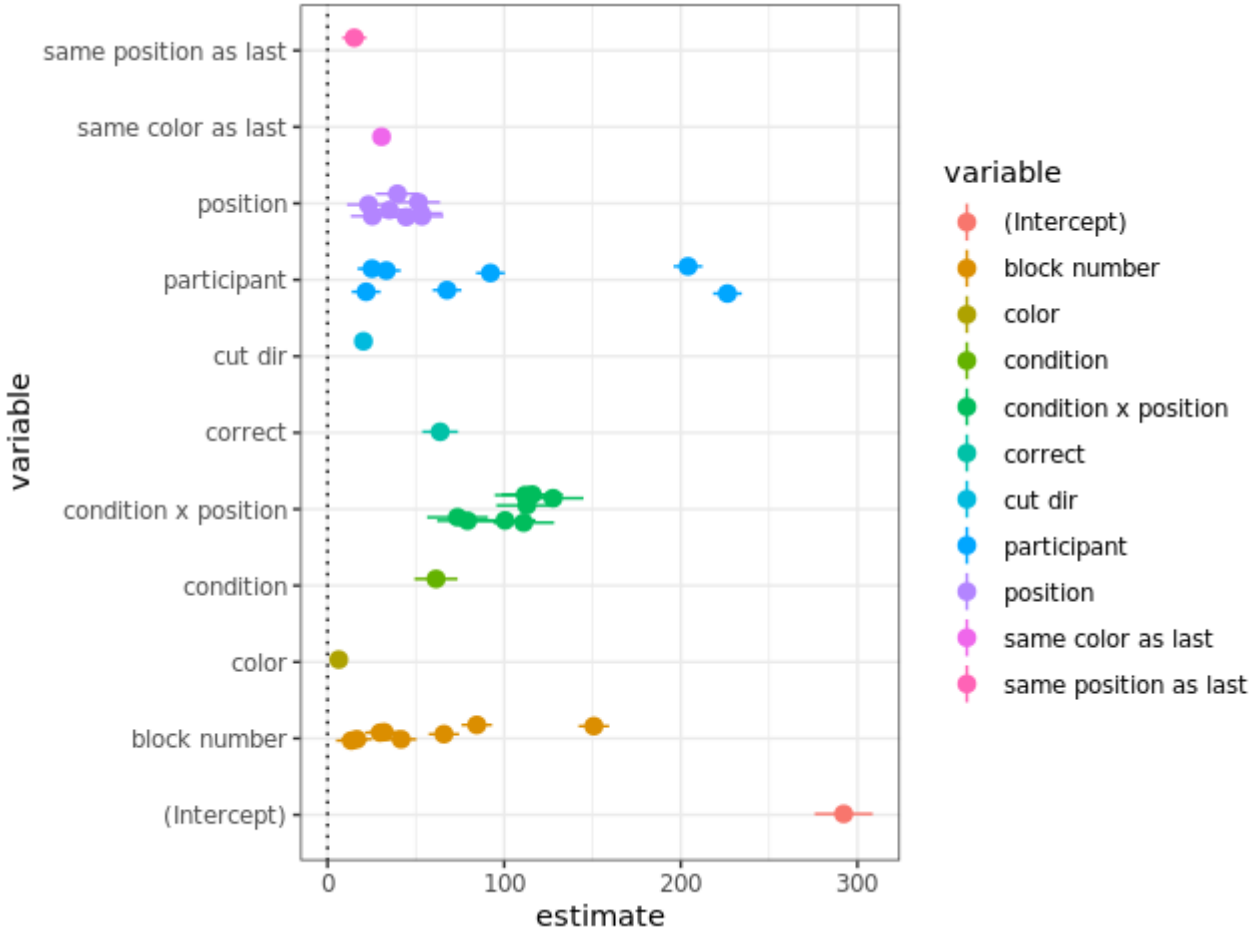
The ANOVA model – popout data

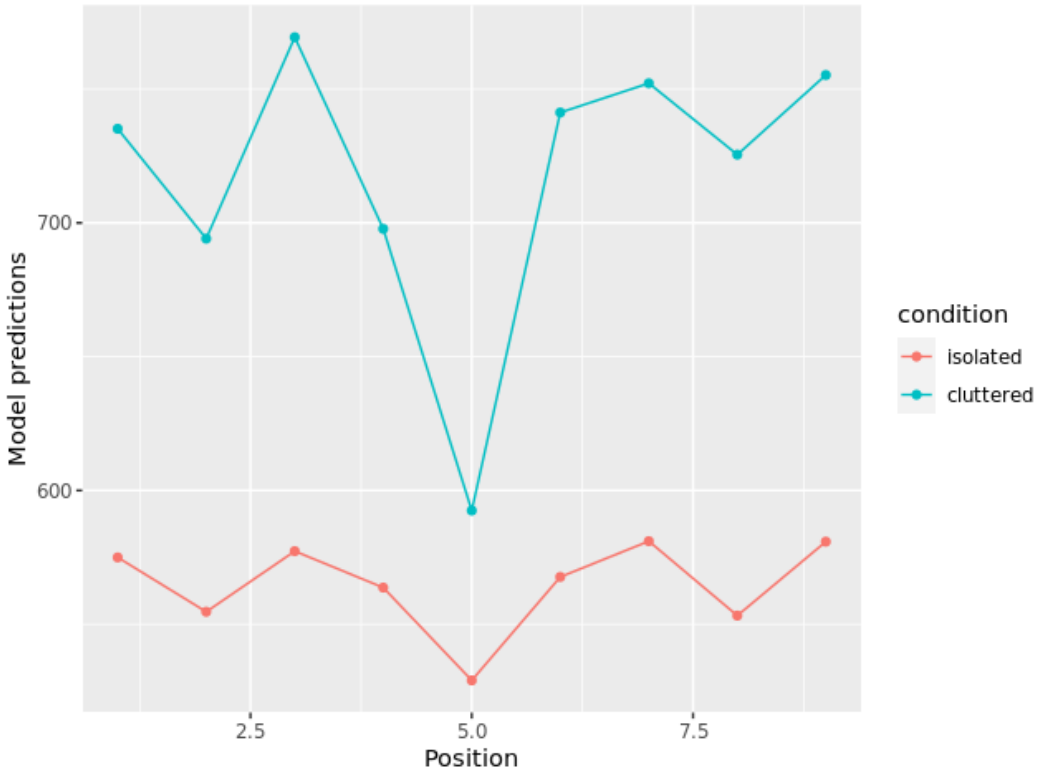




aov(reaction_time ~ condition + position + color + cut_dir + correct + block_number + participant + same_position_as_last + same_color_as_last + position * condition, data = popout_data)

The ANOVA model – popout data

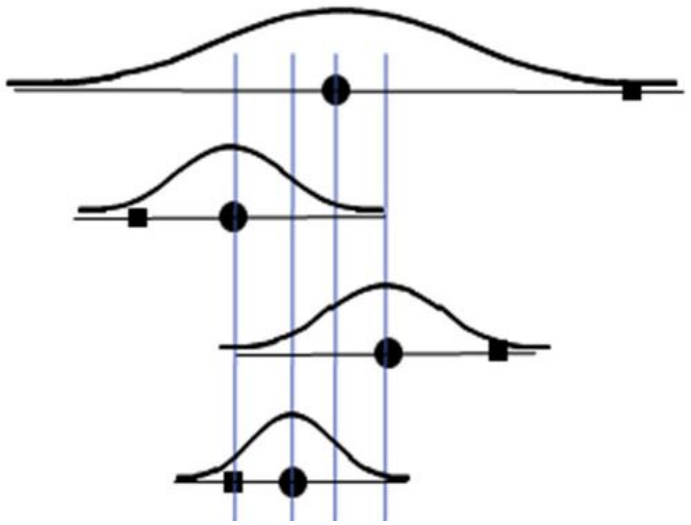




aov(reaction_time ~ condition + position + color + cut_dir + correct + block_number + participant + same_position_as_last + same_color_as_last + position * condition, data = popout_data)

Brief mention: random effects models

In a random effects ANOVA, factor levels are viewed as being randomly generated from an underlying distribution, rather than having a fixed number of levels.



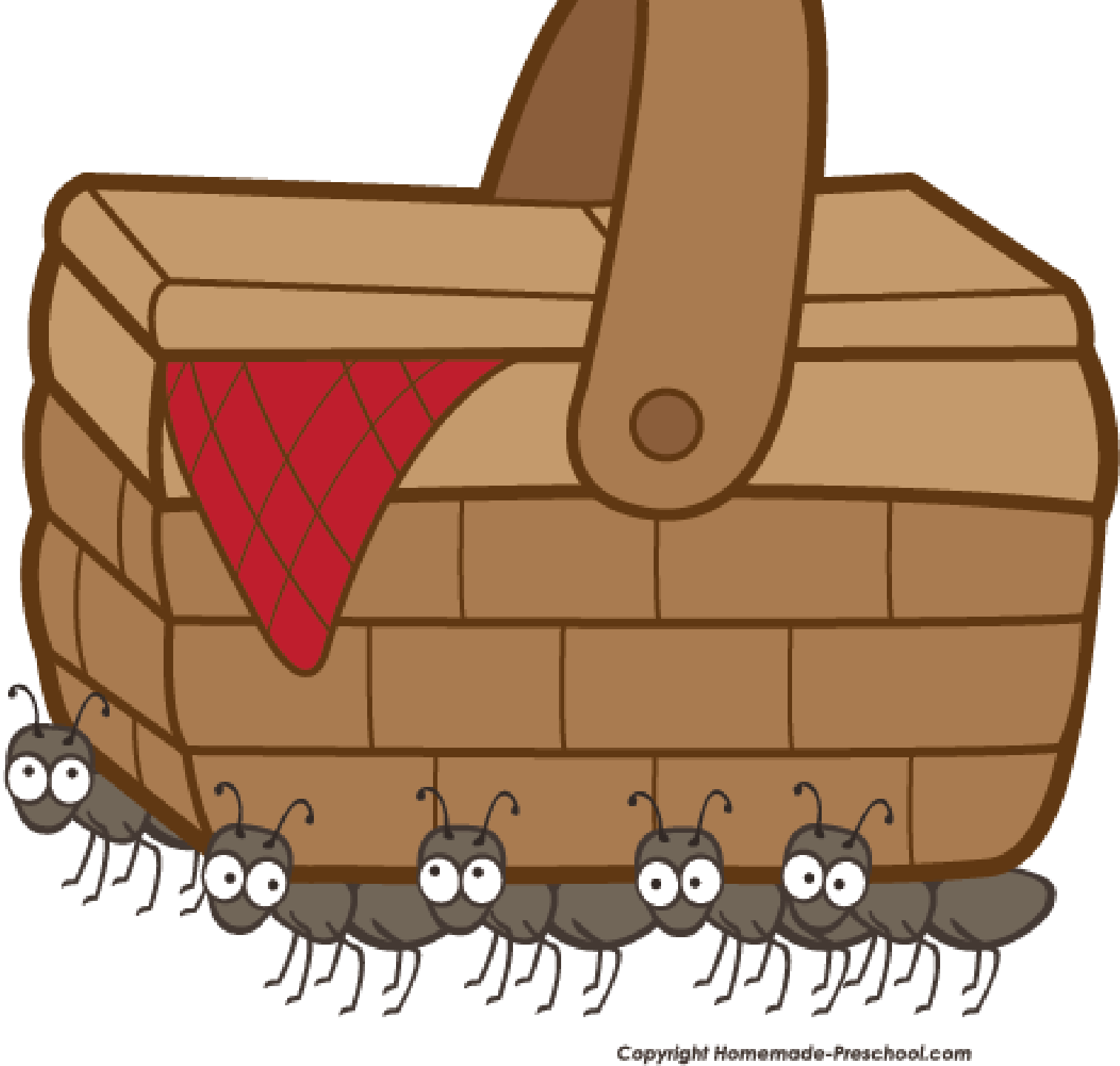
For example, we could view participants in an experiment as being a random sample from participants in a population.

- We then just estimate a mean and standard deviation for the underlying population, rather a separately ID for each participant.
  - This leads to few parameters and hence more degrees of freedom.

You can run mixed effects models in R using the lme4 package

- This is beyond what we will do in this class :/

Let’s explore these topics in R…

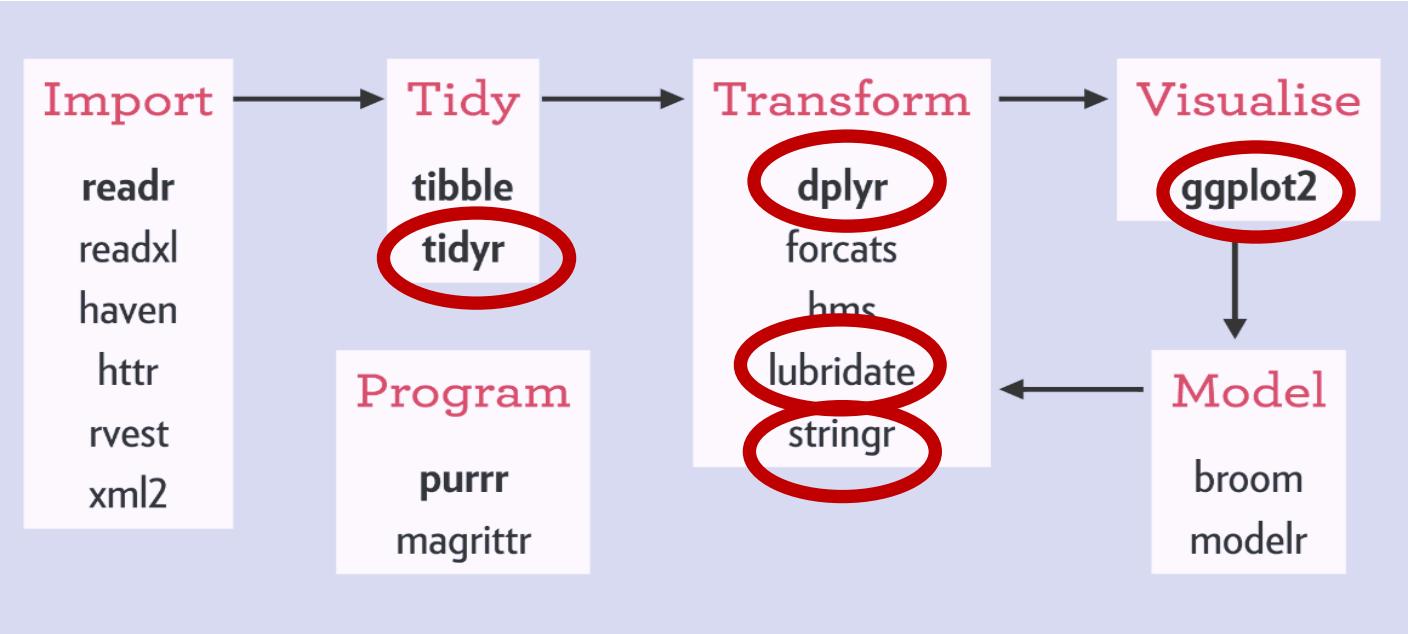


Continuation of tidyverse packages useful for your projects

Tidyverse packages useful for your projects

The packages share a common design philosophy

- Most written by Hadley Wickham



The posit cheat sheets can be very useful

stringr

stringr is a package for manipulate character strings

- library(stringr)

There are many useful functions in the stringr package including:

- str_to_lower()
- str_trim(), str_pad()
- str_detect()
- str_replace_all()

You can use **regular expressions** to make the string matching much more powerful

Let’s try it in R…

tidyr for pivoting data

Wide vs. Long data

Plotting data using ggplot requires that data is in the right format

- i.e., requires data transformations

Often this involves converting data from a **wide format** to **long format**

Wide data

Person	Age	Height
Bob	32	72
Alice	24	65
Steve	64	70

Long data

Person	name	value
Bob	Age	32
Bob	Height	72
Alice	Age	24
Alice	Height	65
Steve	Age	64
Steve	Height	70

library(tidyr)

tidyr::pivot_longer()

pivot_longer(df, cols) converts data from **wide** to **long**

- Takes multiple columns and converts them into two columns: name and value
  - Column names become categorical variable levels of a new variable called **name**
  - The data in rows become entries in a variable called **value**

Wide data

Person	Age	Height
Bob	32	72
Alice	24	65
Steve	64	70

Long data

Person	name	value
Bob	Age	32
Bob	Height	72
Alice	Age	24
Alice	Height	65
Steve	Age	64
Steve	Height	70

tidyr::pivot_wider()

pivot_wider(df, names_from, values_from) converts data from long to wide

- Turns the levels of categorical data into columns in a data frame

Narrow data

person	name	value
Bob	Age	32
Bob	Height	72
Alice	Age	24
Alice	Height	65
Steve	Age	64
Steve	Height	70

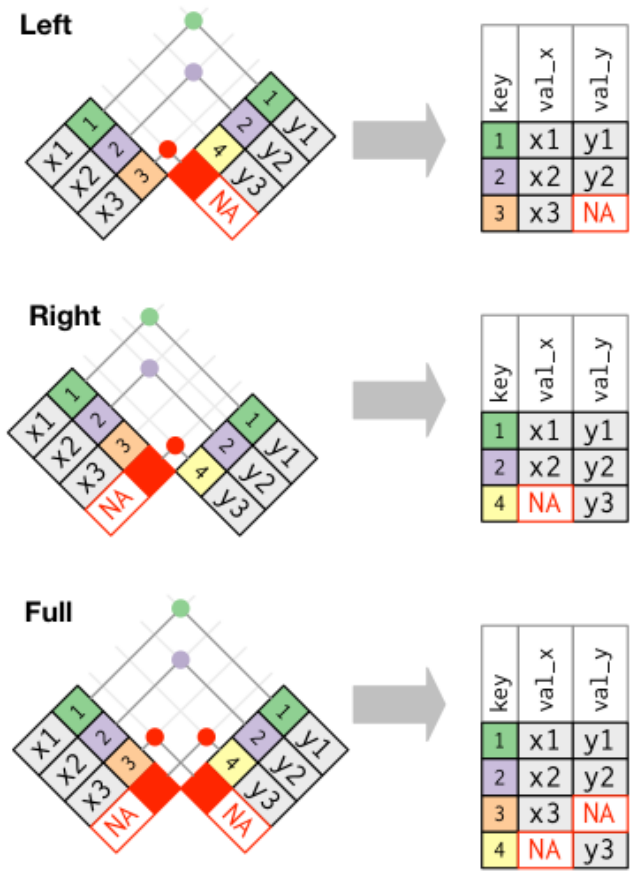
→

Wide data

Person	Age	Height
Bob	32	72
Alice	24	65
Steve	64	70

Let's try it in R…

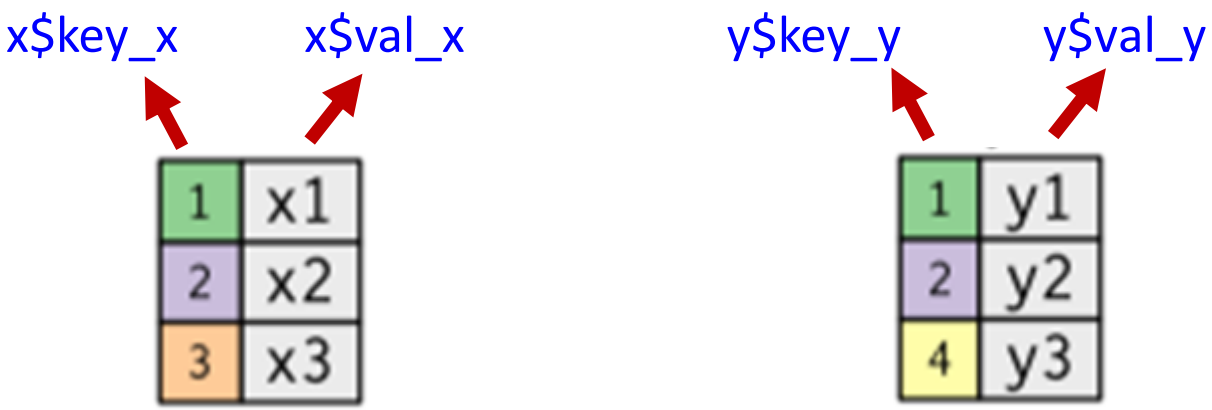
Joining data frames



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



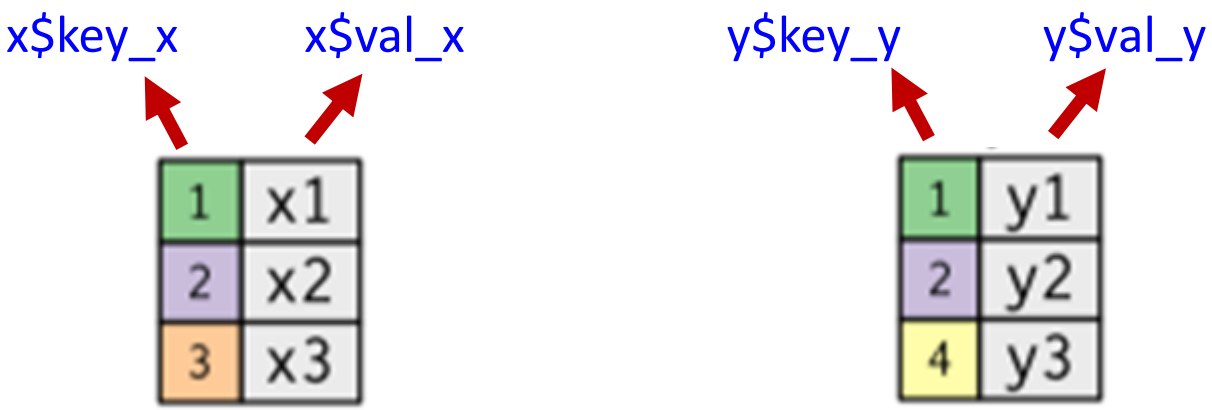
Data frame x **Data frame y**

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



Data frame x **Data frame y**

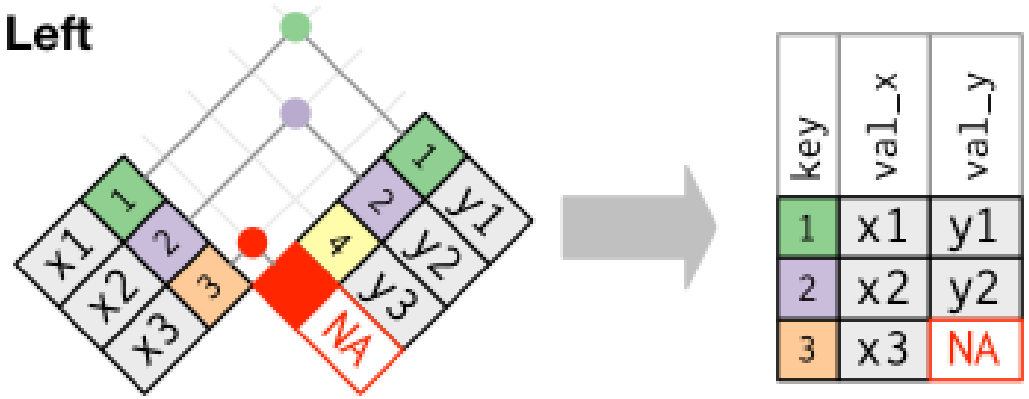
Joins have the general form:

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

Left joins

Left joins keep all rows in the <u>left</u> table.

Data from <u>right</u> table is added when there is a matching key, otherwise NA as added.

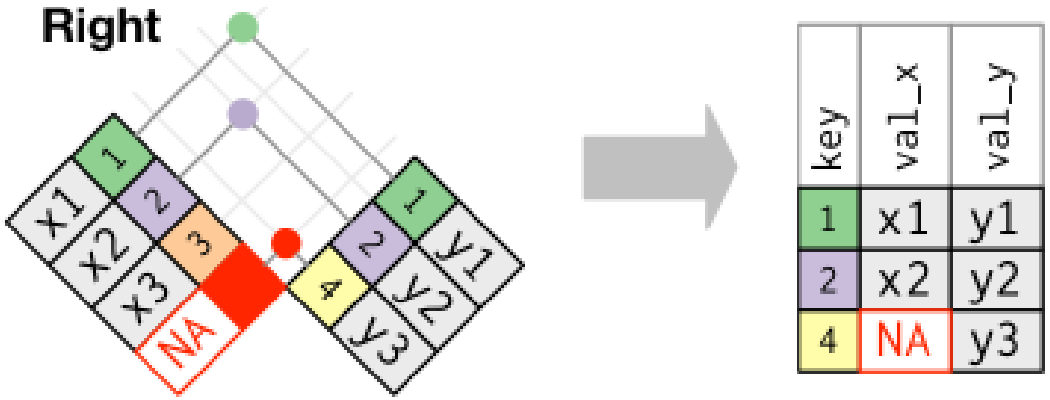



```
> left_join(x, y, by = c("key_x" = "key_y"))
```

Right joins

Right joins keep all rows in the right table.

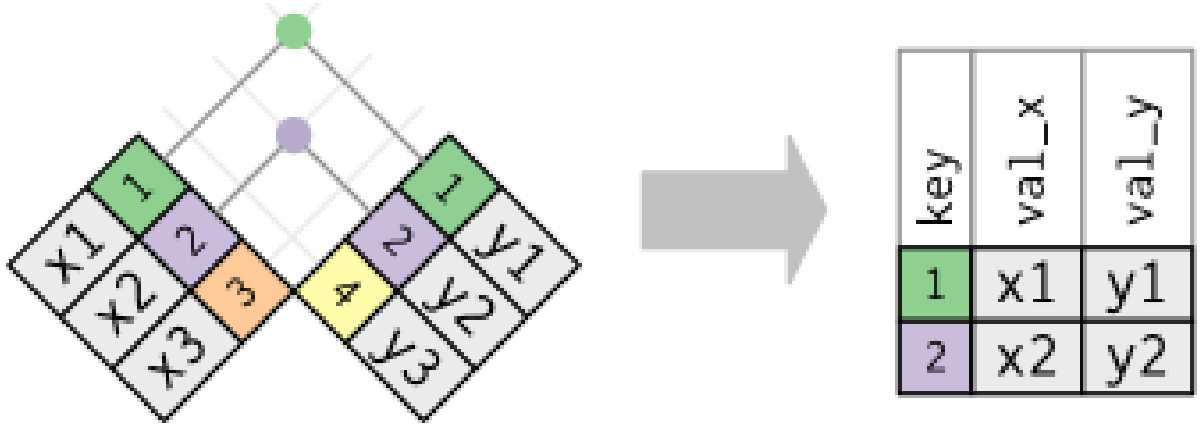
Data from left table added when there is a matching key, otherwise NA as added.




```
> 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.

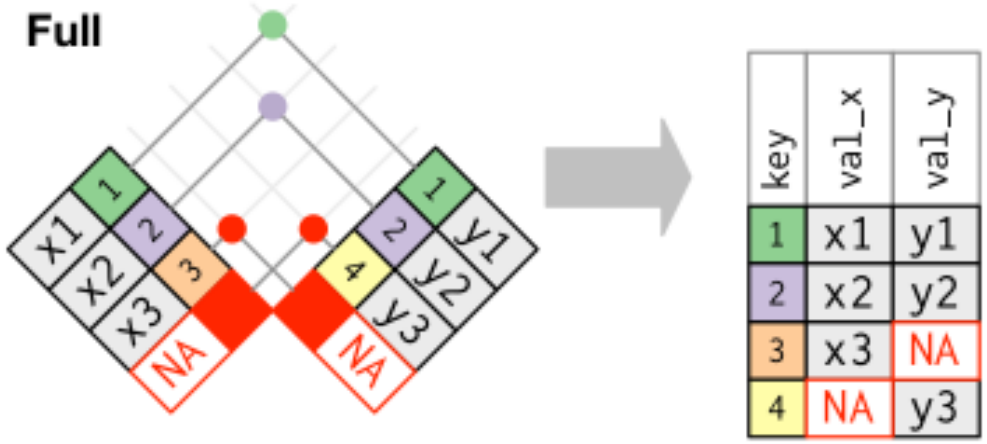



```
> inner_join(x, y, by = c("key_x" = "key_y"))
```

Full joins

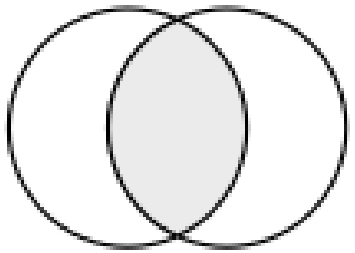
Full joins keep all rows in both table.

NAs are added where there are no matches.

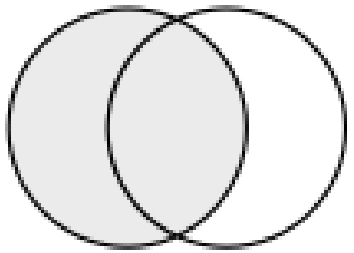



```
> full_join(x, y, by = c("key_x" = "key_y"))
```

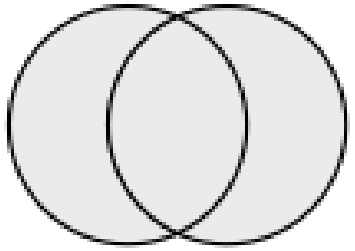
Summary



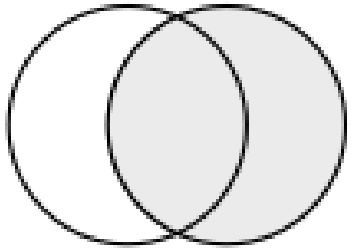
inner_join(x, y)



left_join(x, y)



full_join(x, y)

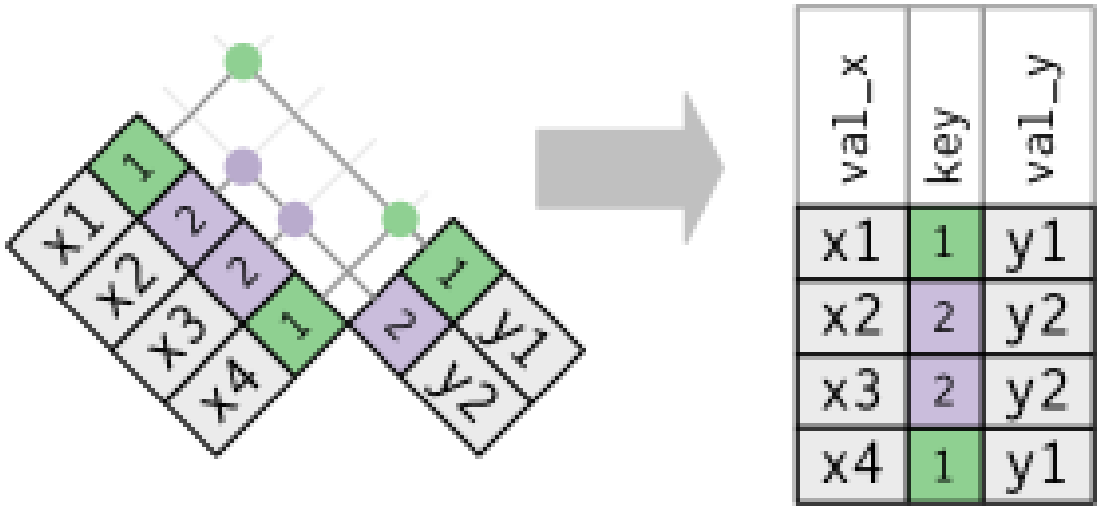


right_join(x, y)

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

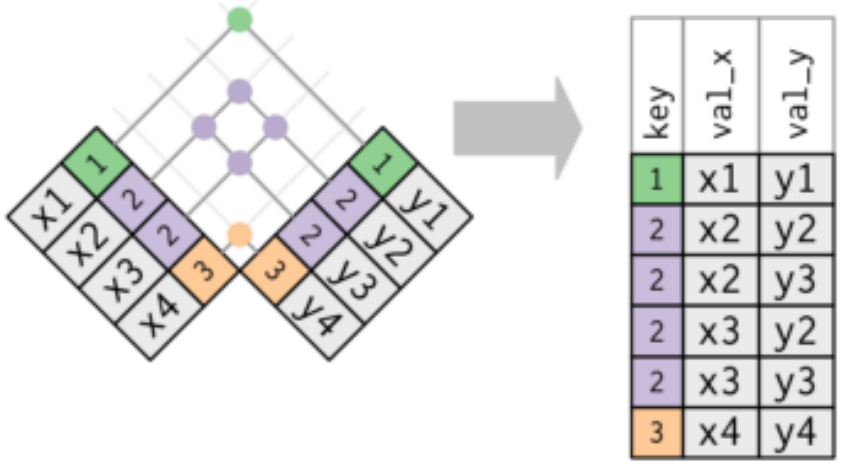


val_x	key	val_y
x1	1	y1
x2	2	y2
x3	2	y2
x4	1	y1

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 <u>multiple keys</u> 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 <- 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
`inner_join(x, y, by = "z")`	`SELECT * FROM x INNER JOIN y USING (z)`
`left_join(x, y, by = "z")`	`SELECT * FROM x LEFT OUTER JOIN y USING (z)`
`right_join(x, y, by = "z")`	`SELECT * FROM x RIGHT OUTER JOIN y USING (z)`
`full_join(x, y, by = "z")`	`SELECT * FROM x FULL OUTER JOIN y USING (z)`

Let’s try it in R…