

Linear statistical models in R

Melbourne Statistical Consulting Platform

University of Melbourne

April 2024

Linear regression

Estimate intercept and slopes of line of best fit.

Test hypothesis that slope is zero.

Residuals assumed to independent, normally distributed with equal variance.

ANOVA

Estimate group means.

Test hypothesis that all group means are equal.

Residuals assumed to independent, normally distributed with equal variance.

Linear regression

Estimate intercept and slopes of line of best fit.

Test hypothesis that slope is zero.

Residuals assumed to independent, normally distributed with equal variance.

ANOVA

Estimate mean for reference category, and difference between reference category and each other category.

Test hypothesis that all between-group differences are equal to zero.

Residuals assumed to independent, normally distributed with equal variance.

Same statistical model, different research questions.

Some theory (with examples and pictures): <https://lindeloev.github.io/tests-as-linear/>

$$\text{Weight}_{(\text{tons})} = 2.4 + \underline{0.3}(\text{height}) + \dots$$



@allison_horst

if all other variables constant, we expect a 1 foot taller dragon to weigh 0.3 tons more, on average.

$$\text{Weight}_{(\text{tons})} = 2.4 + \underline{0.6} * (\text{spotted}) + \dots$$

pattern reference level: striped

if other variables constant, spotted dragons will weigh 0.6 tons more than striped dragons, on average.



@allison_horst

Fitting a linear model

The file `car_interest_rates.csv` contains data on car interest rates in six American cities.

```
car_interest <- read_csv("car_interest_rates.csv")
```

```
car_interest %>%  
  group_by(City) %>%  
  summarise(n = n(),  
            Interest_rate_mean = mean(Interest_rate_perc),  
            Interest_rate_sd = sd(Interest_rate_perc))
```

A tibble: 6 × 4

	City <chr>	n <int>	Interest_rate_mean <dbl>	Interest_rate_sd <dbl>
1	Atlanta	9	13.2	0.447
2	Chicago	9	12.6	0.710
3	Houston	9	13.3	0.556
4	Memphis	9	13.2	0.967
5	New York	9	13.5	0.719
6	Philadelphia	9	12.2	0.504

We use the `lm()` function to fit a linear model to this data. If the explanatory variable (`City`) is a character or factor, `lm()` will treat it as a categorical variable (like ANOVA). If it is a numeric variable, `lm()` will treat it as continuous (like linear regression). Beware categories encoded as numbers!

```
m <- lm(Interest_rate_perc ~ City,  
       data = car_interest)
```

Unlike fitting models in most statistical software, `lm()` doesn't produce any output itself. Instead we save the linear model to an object, e.g. `m` in this case, and then examine the model using other functions.

Simple model summaries

The `summary()` function produces a lot of information about a model. Includes coefficient estimates, standard errors, and p-values; R-squared and adjusted R-squared. Doesn't include 95% confidence intervals for coefficients or overall tests for factors.

```
summary(m)
```

Call:

```
lm(formula = Interest_rate_perc ~ City, data = car_interest)
```

Residuals:

Min	1Q	Median	3Q	Max
-1.3544	-0.3050	-0.1111	0.3781	1.7556

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	13.1944	0.2244	58.793	< 2e-16 ***
CityChicago	-0.5833	0.3174	-1.838	0.07226 .
CityHouston	0.1122	0.3174	0.354	0.72520
CityMemphis	0.0500	0.3174	0.158	0.87548
CityNew York	0.2889	0.3174	0.910	0.36725
CityPhiladelphia	-0.9944	0.3174	-3.133	0.00295 **

Signif. codes:

0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6733 on 48 degrees of freedom

Multiple R-squared: 0.3347, Adjusted R-squared: 0.2654

F-statistic: 4.829 on 5 and 48 DF, p-value: 0.001175

Assessing model fit: R-squared and others

Is the model a good fit? The `model_performance()` function in the `performance` package provides a number of model summary measures in a data frame.

```
library(performance)
model_performance(m)
```

```
# Indices of model performance
```

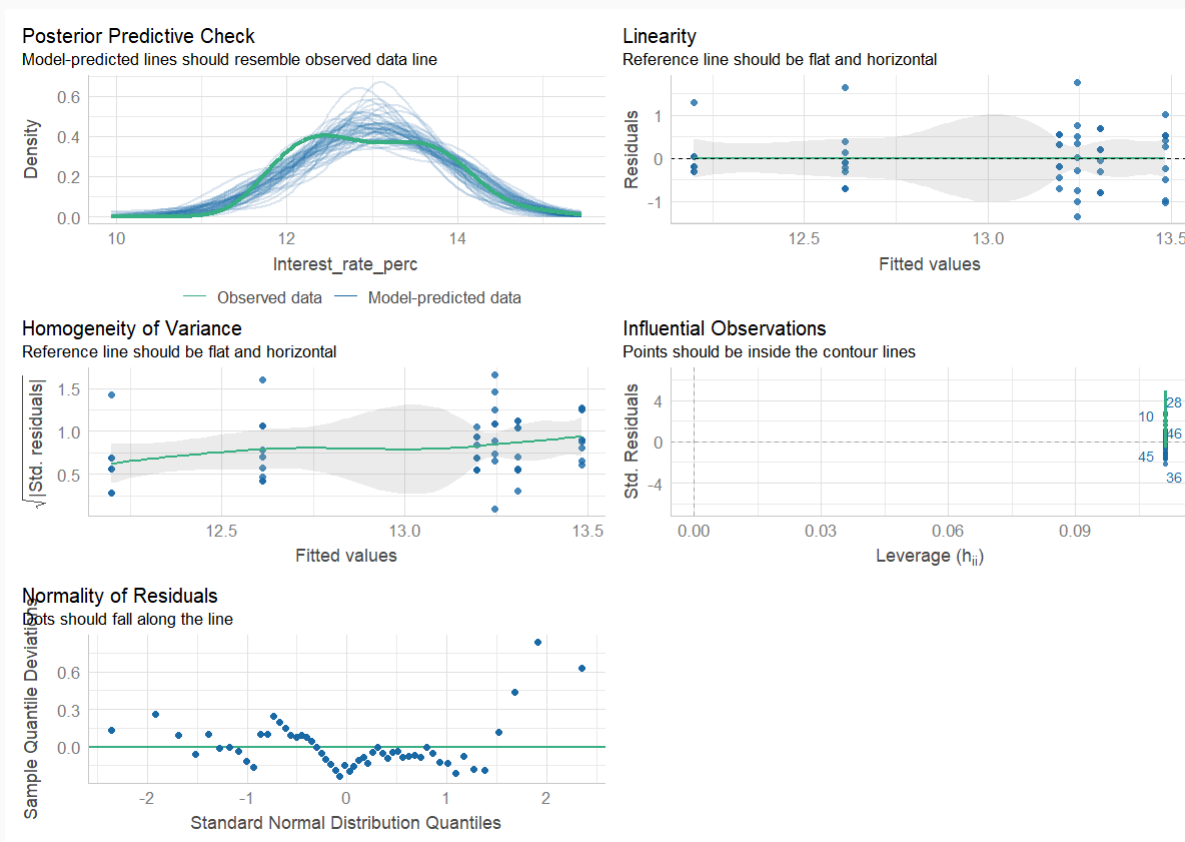
AIC	AICc	BIC	R2	R2 (adj.)	RMSE	Sigma
118.160	120.594	132.082	0.335	0.265	0.635	0.673

The `R2` of `0.335` indicates that the model explains 33.5% of the variation in the data. Interpreting this further requires context.

Assessing model fit: residual plots

The `check_model()` function in the **performance** package provides a number of useful diagnostic plots.

```
library(performance)
check_model(m)
```



Overall/joint tests (ANOVA)

The `joint_tests()` function in the `emmeans` package provides 'overall' or 'joint' tests for each variable in a statistical model.

(Pedantic note: the `joint_tests()` function computes Type III ANOVA, which means it tests for evidence of difference between full model and model with each variable removed.)

```
library(emmeans)
```

```
joint_tests(m)
```

model	term	df1	df2	F.ratio	p.value
City		5	48	4.829	0.0012

Parameter estimates (coefficients) in a table

We saw the parameter estimates shown in the output from `summary()` but we can also present them nicely in a table using `tbl_regression()` (from the `gtsummary` package). This is typically more interesting for a regression than ANOVA.

```
library(gtsummary)
tbl_regression(m)
```

Characteristic	Beta	¹ >95% CI ¹	p-value
City			
Atlanta	—	—	
Chicago	-0.58	-1.2, 0.05	0.072
Houston	0.11	-0.53, 0.75	0.7
Memphis	0.05	-0.59, 0.69	0.9
New York	0.29	-0.35, 0.93	0.4
Philadelphia	-0.99	-1.6, -0.36	0.003
¹ CI = Confidence Interval			

Parameter estimates (coefficients) in a data frame

We saw the parameter estimates shown in the output from `summary()` but we can also get them in a data frame using `tidy()` (from the `broom` package). This is typically more interesting for a regression than ANOVA. The parameter `conf.int = TRUE` shows confidence intervals.

```
library(broom)
tidy(m, conf.int = TRUE)
```

```
# A tibble: 6 × 7
```

	term	estimate	std.error	statistic	p.value	conf.low	conf.high
	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	(Intercept)	13.2	0.224	58.8	2.19e-46	12.7	13.6
2	CityChicago	-0.583	0.317	-1.84	7.23e- 2	-1.22	0.0548
3	CityHouston	0.112	0.317	0.354	7.25e- 1	-0.526	0.750
4	CityMemphis	0.0500	0.317	0.158	8.75e- 1	-0.588	0.688
5	CityNew York	0.289	0.317	0.910	3.67e- 1	-0.349	0.927
6	CityPhiladelphia	-0.994	0.317	-3.13	2.95e- 3	-1.63	-0.356

Estimated marginal means (a.k.a. least squares means)

The `emmeans` package addresses all of your needs for group means, confidence intervals and pairwise comparisons.

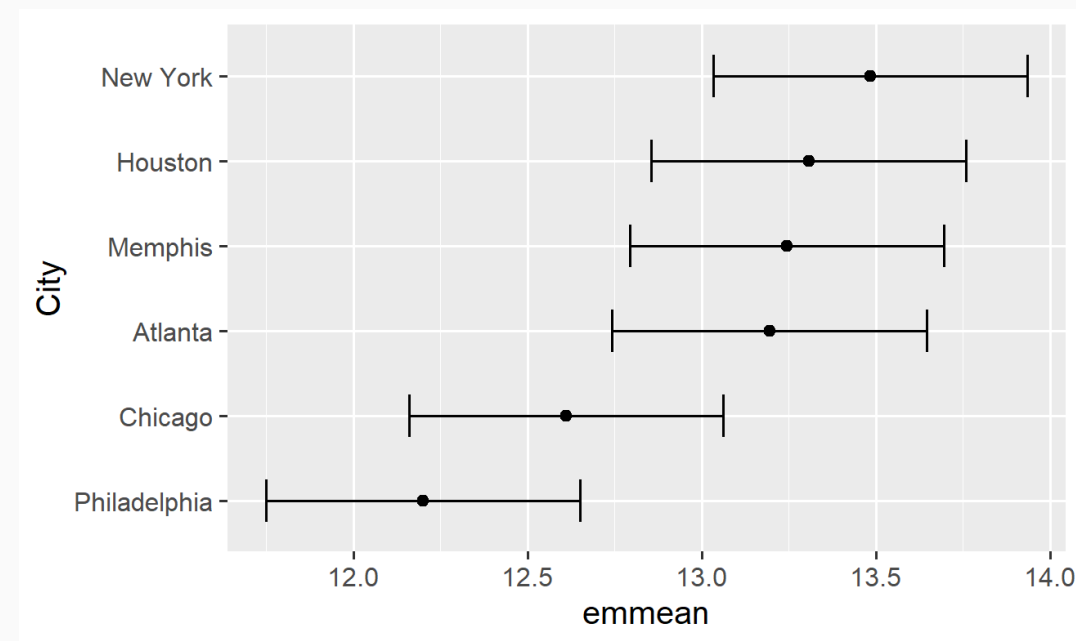
```
emmeans(m, "City")
```

City	emmean	SE	df	lower.CL	upper.CL
Atlanta	13.2	0.224	48	12.7	13.6
Chicago	12.6	0.224	48	12.2	13.1
Houston	13.3	0.224	48	12.9	13.8
Memphis	13.2	0.224	48	12.8	13.7
New York	13.5	0.224	48	13.0	13.9
Philadelphia	12.2	0.224	48	11.7	12.7

Confidence level used: 0.95

```
emmeans(m, "City") %>%  
  as_tibble() %>%  
  mutate(City = fct_reorder(City, emmean)) %>%  
  ggplot(aes(y = City, x = emmean,  
             xmin = lower.CL, xmax = upper.CL)) +  
  geom_point() +  
  geom_errorbar(width = 0.5)
```

The `as_tibble()` function converts the means and confidence intervals to a data frame. Good for custom tables or plotting.



Pairwise comparisons

Here `adjust = "none"` computes Fisher's LSD comparisons, i.e. no adjustment for multiple comparisons. A common alternative is `adjust = "tukey"` for Tukey's HSD. `as_tibble()` can be used to get this output in a data frame, too.

```
emmeans(m, "City") %>%  
  pairs(adjust = "none", infer = TRUE)
```

contrast	estimate	SE	df	lower.CL	upper.CL	t.ratio	p.value
Atlanta - Chicago	0.5833	0.317	48	-0.0548	1.22148	1.838	0.0723
Atlanta - Houston	-0.1122	0.317	48	-0.7504	0.52592	-0.354	0.7252
Atlanta - Memphis	-0.0500	0.317	48	-0.6881	0.58814	-0.158	0.8755
Atlanta - New York	-0.2889	0.317	48	-0.9270	0.34925	-0.910	0.3673
Atlanta - Philadelphia	0.9944	0.317	48	0.3563	1.63259	3.133	0.0029
Chicago - Houston	-0.6956	0.317	48	-1.3337	-0.05741	-2.192	0.0333
Chicago - Memphis	-0.6333	0.317	48	-1.2715	0.00481	-1.995	0.0517
Chicago - New York	-0.8722	0.317	48	-1.5104	-0.23408	-2.748	0.0084
Chicago - Philadelphia	0.4111	0.317	48	-0.2270	1.04925	1.295	0.2014
Houston - Memphis	0.0622	0.317	48	-0.5759	0.70036	0.196	0.8454
Houston - New York	-0.1767	0.317	48	-0.8148	0.46148	-0.557	0.5804
Houston - Philadelphia	1.1067	0.317	48	0.4685	1.74481	3.487	0.0011
Memphis - New York	-0.2389	0.317	48	-0.8770	0.39925	-0.753	0.4553
Memphis - Philadelphia	1.0444	0.317	48	0.4063	1.68259	3.291	0.0019
New York - Philadelphia	1.2833	0.317	48	0.6452	1.92148	4.043	0.0002

Extension: interactions and interaction plots (1)

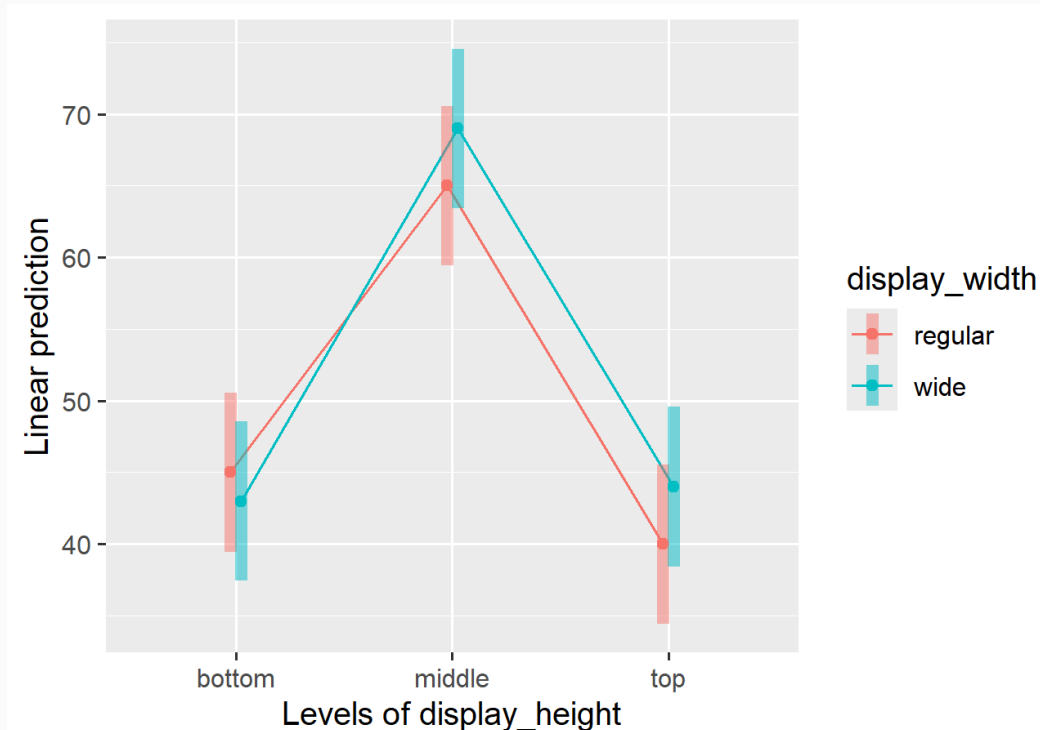
An adequate treatment of interactions or multiple regression is beyond this course.

```
bread_sales <- read_csv("bread_sales.csv")
m <- lm(sales_cases ~ display_height*display_width,
        data = bread_sales)
joint_tests(m)
```

model term	df1	df2	F.ratio	p.value
display_height	2	6	74.710	0.0001
display_width	1	6	1.161	0.3226
display_height:display_width	2	6	1.161	0.3747

`emmip()` is a fantastic way to explore interactions. It can even produce faceted plots for multi-way interactions.

```
emmip(m, display_width ~ display_height, CIs = TRUE)
```



Extension: interactions and interaction plots (2)

Means for each combination of variables:

```
emmeans(m, c("display_height", "display_width"))
```

display_height	display_width	emmean	SE	df	lower.CL	upper.CL
bottom	regular	45	2.27	6	39.4	50.6
middle	regular	65	2.27	6	59.4	70.6
top	regular	40	2.27	6	34.4	45.6
bottom	wide	43	2.27	6	37.4	48.6
middle	wide	69	2.27	6	63.4	74.6
top	wide	44	2.27	6	38.4	49.6

Confidence level used: 0.95

Comparisons for a main effect:

```
emmeans(m, "display_height") %>%  
  pairs(adjust = "none", infer = TRUE)
```

contrast	estimate	SE	df	lower.CL	upper.CL	t.ratio	p.value
bottom - middle	-23	2.27	6	-28.56	-17.44	-10.119	0.0001
bottom - top	2	2.27	6	-3.56	7.56	0.880	0.4128
middle - top	25	2.27	6	19.44	30.56	10.999	<.0001

Results are averaged over the levels of: display_width

Confidence level used: 0.95

Comparing levels of one factor within another factor:

```
emmeans(m, "display_width", by = "display_height") %>%  
  pairs(adjust = "none", infer = TRUE)
```

display_height = bottom:

contrast	estimate	SE	df	lower.CL	upper.CL	t.ratio	p.value
regular - wide	2	3.21	6	-5.87	9.87	0.622	0.5567

display_height = middle:

contrast	estimate	SE	df	lower.CL	upper.CL	t.ratio	p.value
regular - wide	-4	3.21	6	-11.87	3.87	-1.244	0.2598

display_height = top:

contrast	estimate	SE	df	lower.CL	upper.CL	t.ratio	p.value
regular - wide	-4	3.21	6	-11.87	3.87	-1.244	0.2598

Confidence level used: 0.95

Extension: fitting multiple models at once (1)

Often there are multiple outcome or explanatory variables of interest, and we want to fit a model with a similar structure to each. We can use `pivot_longer()`, `group_by()` and `summarise()` to help us here.

This is the piglet data we've seen before, aggregated over time so each row represents an independent observation of a pig pen.

```
pig_behaviour_means <- read_csv("pig_behaviour_means.csv")
glimpse(pig_behaviour_means)
```

```
Rows: 35
Columns: 14
$ Pen      <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 1...
$ Housing  <chr> "FC", "FC", "FC", "FC", "FC", "FC", "PS", "PS", ...
$ Treatment <chr> "HC", "HC", "HC", "HC", "HC", "HC", "HC", "HC", ...
$ HousTreat <chr> "FC, HC", "FC, HC", "FC, HC", "FC, HC", "FC, HC", ...
$ Sex      <chr> "M", "M", "M", "M", "M", "M", "M", "M", "M", "M", ...
$ Total_pigs <dbl> 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, ...
$ Time_points <dbl> 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 7, 7, 7, 7, ...
$ Upright  <dbl> 3.3750, 2.6250, 6.1250, 5.6250, 7.1250, 7.6250, ...
$ Aggression <dbl> 0.0000, 0.0000, 0.2500, 0.2500, 0.0000, 0.3750, ...
$ Chewing_pig <dbl> 1.0000, 0.1250, 1.8750, 1.6250, 2.0000, 2.7500, ...
$ Playing  <dbl> 0.0000, 0.1250, 0.0000, 0.1250, 0.3750, 0.6250, ...
```

We can turn this data into long form with `pivot_longer()`, so the different types of pig behaviour are levels of a factor rather than different variables.

```
pig_long <- pivot_longer(pig_behaviour_means,
  Upright:Nosing_pen,
  names_to = "Behaviour",
  values_to = "Num_pigs")
glimpse(pig_long)
```

```
Rows: 245
Columns: 9
$ Pen      <dbl> 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3,...
$ Housing  <chr> "FC", "FC", "FC", "FC", "FC", "FC", "FC", "FC", "FC", "F...
$ Treatment <chr> "HC", "HC", "HC", "HC", "HC", "HC", "HC", "HC", "H...
$ HousTreat <chr> "FC, HC", "FC, HC", "FC, HC", "FC, HC", "FC, HC", "FC, HC", "FC, HC", ...
$ Sex      <chr> "M", "M", "M", "M", "M", "M", "M", "M", "M", "M", ...
$ Total_pigs <dbl> 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, ...
$ Time_points <dbl> 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8, ...
$ Behaviour <chr> "Upright", "Aggression", "Chewing_pig", "Playing",...
$ Num_pigs  <dbl> 3.375, 0.000, 1.000, 0.000, 0.250, 2.625, 0.125, 2...
```


Extension: fitting multiple models at once (2)

Now group by `Behaviour` and use `summarise()` to fit a linear model to each group. The `cur_data()` function returns the data frame for the current group inside `mutate()` or `summarise()`.

```
pig_models <- pig_long %>%
  group_by(Behaviour) %>%
  summarise(model = list(lm(Num_pigs ~ Housing*Treatment,
                           data = cur_data())))
```

```
pig_models
```

```
# A tibble: 7 × 2
  Behaviour      model
  <chr>         <list>
1 Aggression    <lm>
2 Chewing_pig   <lm>
3 Exploring_pen <lm>
4 Nosing_pen    <lm>
5 Playing       <lm>
6 Upright       <lm>
7 Vocalising   <lm>
```

We now have a column `model` containing a linear model for each behaviour.

We can use `rowwise()`, `summarise()` and `emmeans()` to extract information from these models, which we could then e.g. plot.

```
pig_means <- pig_models %>%
  rowwise(Behaviour) %>%
  summarise(as_tibble(emmeans(model,
                              c("Housing", "Treatment"))))
```

Warning: Returning more (or less) than 1 row per ``summarise()`` group was deprecated in dplyr 1.1.0.

• Please use ``reframe()`` instead.

• When switching from ``summarise()`` to ``reframe()``, remember that ``reframe()`` always returns an ungrouped data frame and adjust accordingly.

Call ``lifecycle::last_lifecycle_warnings()`` to see where this warning was generated.

```
pig_means
```

```
# A tibble: 28 × 8
```

```
# Groups:   Behaviour [7]
```

	Behaviour	Housing	Treatment	emmean	SE	df	lower.CL	upper.CL
	<chr>	<fct>	<fct>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	Aggression	FC	C	0.241	0.0822	31	0.0735	0.409
2	Aggression	PS	C	0.145	0.0775	31	-0.0131	0.303
3	Aggression	FC	HC	0.154	0.0701	31	0.0113	0.297
4	Aggression	PS	HC	0.227	0.0878	31	0.158	0.516

Extension: beyond the linear model

The linear model (`lm()`) provides the template for using almost all other statistical models in R, such as...

Generalised linear models, including logistic regression and Poisson regression: `glm()` (built into R)

Mixed effects models, using the `lme4` package: `lmer()` and `glmer()`

Cox proportional hazards models for survival analysis, using the `survival` package: `coxph()`

Generalised additive models, using the `mgcv` package: `gam()`

The functions we've seen here to interpret a model (e.g. `emmeans()`, `check_model()`, `tbl_regression()`, etc) works with all of the model types on the left and more.

Exercise 4.3.