Research Methods and Data Analysis (IAWEL)

Jill R D MacKay

2024-05-08

Table of contents

1 Research Methods and Data Analysis (IAWEL)

Preface

This book accompanies the Research Methods and Data Analysis course on the International Animal Welfare Ethics and Law MSc at the Royal (Dick) School of Veterinary Studies.

It is a companion document to the course, and not core to the materials.

Throughout the RMDA Lectures, you will see a number of statistical tests, data visualisations, data manipulation, text mining, and simple calculations. Almost inevitably, each one of these steps will have been performed in R.

Your R textbook is R@R(D)SVS, and that textbook will explain how to download and install R, how to run simple commands in R, and more. This RMDA textbook is like an accompanying document to your lecture materials, and is a place to help you move your R and statistical knowledge along.

Packages in this textbook

There are a range of packages used in this book, including Tidyverse (Wickham et al. 2019), effsize (Torchiano 2020), ggstatsplot (Patil 2021), vcd (Zeileis, Meyer, and Hornik 2007), wordcloud (Fellows 2018), easystats (Lüdecke et al. 2022), rstan (Stan Development Team 2023), rstanarm (Brilleman et al. 2018)

You may need to download and install a package or load a package for some of these commands to work.

Licensing

for the benefit of the public at large and to the detriment of our heirs and successors. We intend this dedication to be an overt act of relinquishment in perpetuity of all present and future rights to this software under copyright law.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

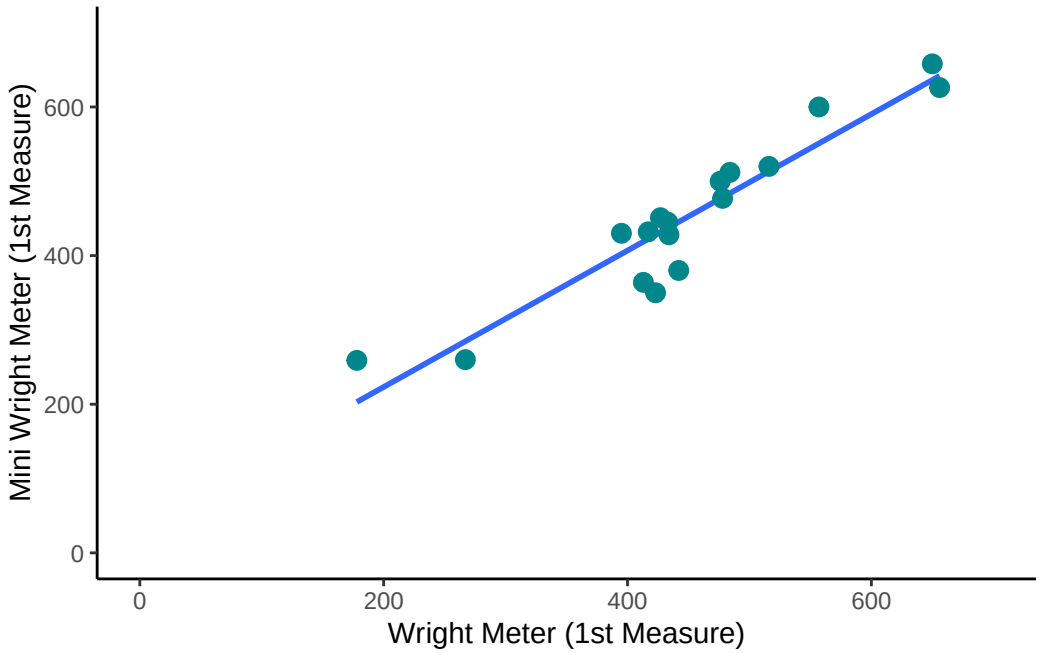
For more information, please refer to https://unlicense.org

2 Week 1: The Philosophy of Science

Lecture 3: The Replication Crisis

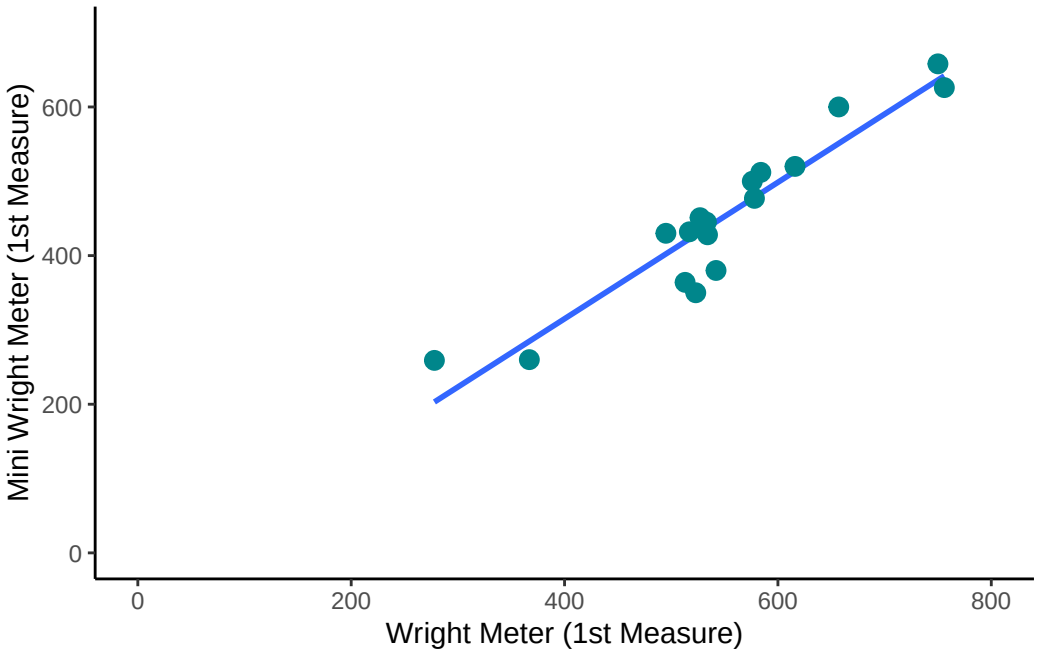
Bland-Altmann Plots are generated with the following code.

```
library(tidyverse)

bland <-  tibble(
  subject = c(1:17),
  Wright1 = c(484,395,516,434,476,557,413,442,650,433,417,656,267,478,178,423,427),
  Wright2 = c(490,397,512,401,470,611,415,431,638,429,420,633,275,492,165,372,421),
  Mini1 = c(512,430,520,428,500,600,364,380,658,445,432,626,260,477,259,350,451),
  Mini2 = c(525,415,508,444,500,625,460,390,642,432,420,605,227,467,268,370,443)
)

bland |>
  ggplot(aes(x = Wright1, y = Mini1)) +
  stat_smooth(method="lm", se=FALSE) +
  geom_point(colour = "turquoise4", size = 3) +
  scale_x_continuous(limits = c(0,700)) +
  scale_y_continuous(limits = c(0,700)) +
  theme_classic() +
  labs(x = "Wright Meter (1st Measure)", y = "Mini Wright Meter (1st Measure)")
```



And then if we add 100 to each measure, we see a very similar plot:

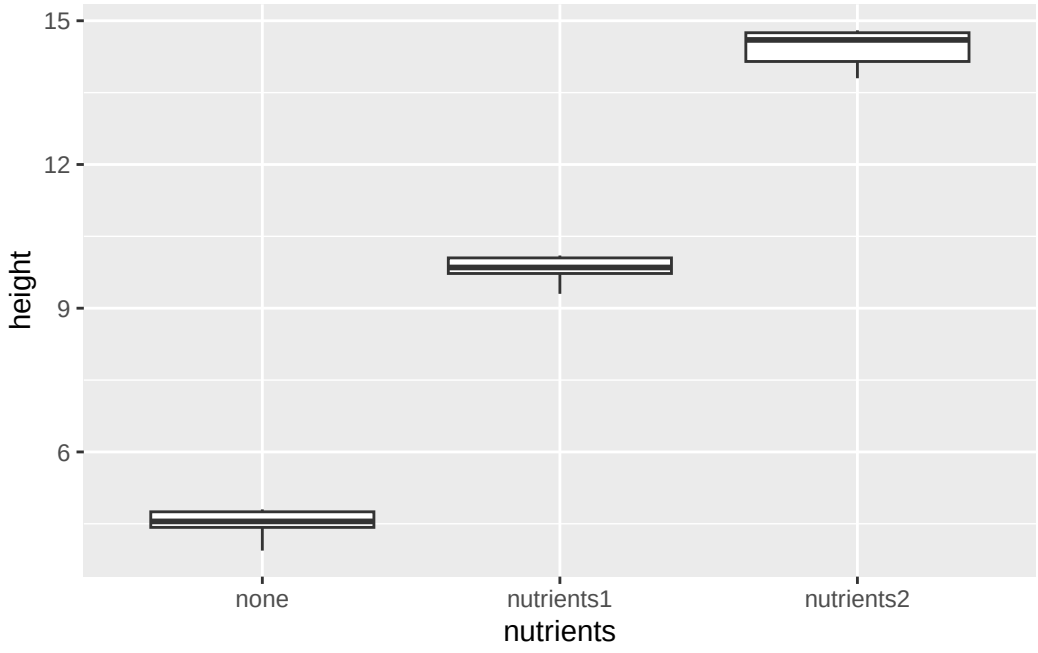
```
bland |>
  mutate (Wright1 = (Wright1+100)) %>%
  ggplot(aes(x = Wright1, y = Mini1)) +
  stat_smooth(method="lm", se=FALSE) +
  geom_point(colour = "turquoise4", size = 3) +
  scale_x_continuous(limits = c(0,800)) +
  scale_y_continuous(limits = c(0,700)) +
  theme_classic() +
  labs(x = "Wright Meter (1st Measure)", y = "Mini Wright Meter (1st Measure)")
```


Mini Wright Meter (1st Measure)
600
400
200
0
0
200
400
600
800
Wright Meter (1st Measure)

Lecture 5: Introduction to Research Methods

2.1 Create data and plot

```
library(tidyverse)

plants <- tibble(none = c(4.8, 4.8, 3.94, 4.4,4.5,4.6),
                 nutrients1  = c( 10.1, 9.7, 9.8, 9.9, 9.3, 10.1),
                 nutrients2 = c(14.8, 14.6, 14.8, 14, 13.8, 14.6))

plants |>
  pivot_longer(cols = c(none, nutrients1,nutrients2),
               names_to = "nutrients",
               values_to = "height") |>
  ggplot(aes(x = nutrients, y = height)) +
  geom_boxplot()
```



2.2 Run an ANOVA on Plant data

```
longplants <- plants |>
  pivot_longer(cols = c(none, nutrients1,nutrients2),
                              names_to = "nutrients",
                              values_to = "height")

plant_model <- aov(height ~ nutrients, data = longplants)

summary(plant_model)
```

```
            Df Sum Sq Mean Sq F value Pr(>F)
nutrients    2 296.10  148.05    1184 <2e-16 ***
Residuals   15   1.88    0.13
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

2.3 Read and Run Crude Chicken Correlations

```
crudechicks <- tibble(year = c("2000", "2001", "2002", "2003",
                               "2004", "2005", "2006", "2007",
                               "2008", "2009"),
                      chicken = c(54.2, 54, 56.8, 57.5, 59.3, 60.5, 60.9,
                                  59.9, 58.7, 56),
                      crude = c(3311, 3405, 3336, 3521, 3674, 3670, 3685,
                                3656, 3571, 3307))


cor.test(crudechicks$chicken, crudechicks$crude, method = "spearman")
```

```
    Spearman's rank correlation rho

data:  crudechicks$chicken and crudechicks$crude
S = 20, p-value = 0.001977
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.8787879
```

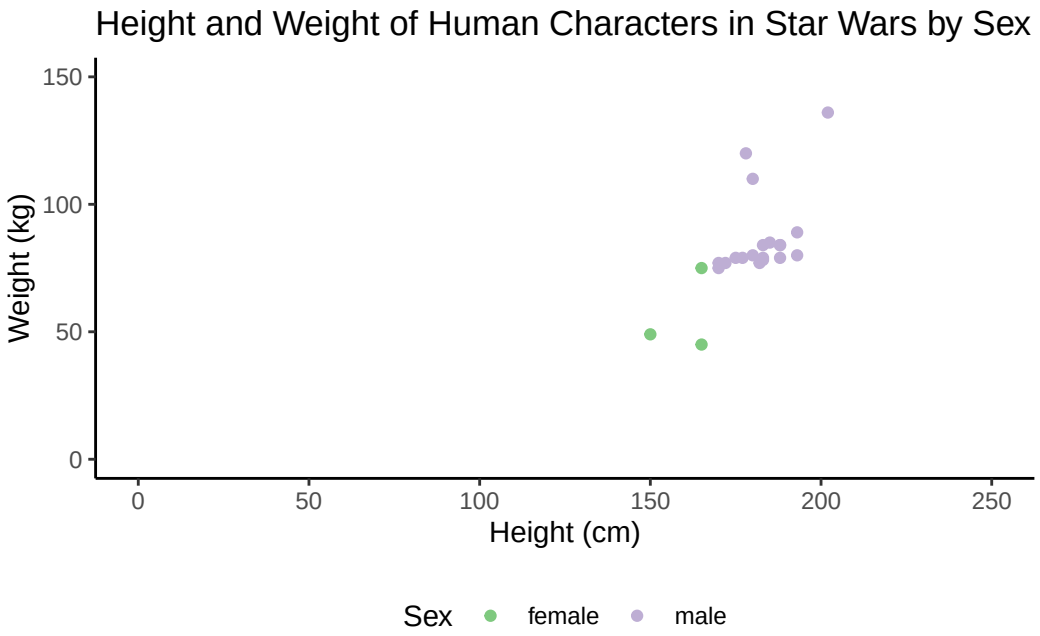
3 Week 2: The Use and Abuse of Data

Lecture 2: Data Visualisation

This code will help you replicate the charts in Lecture 2

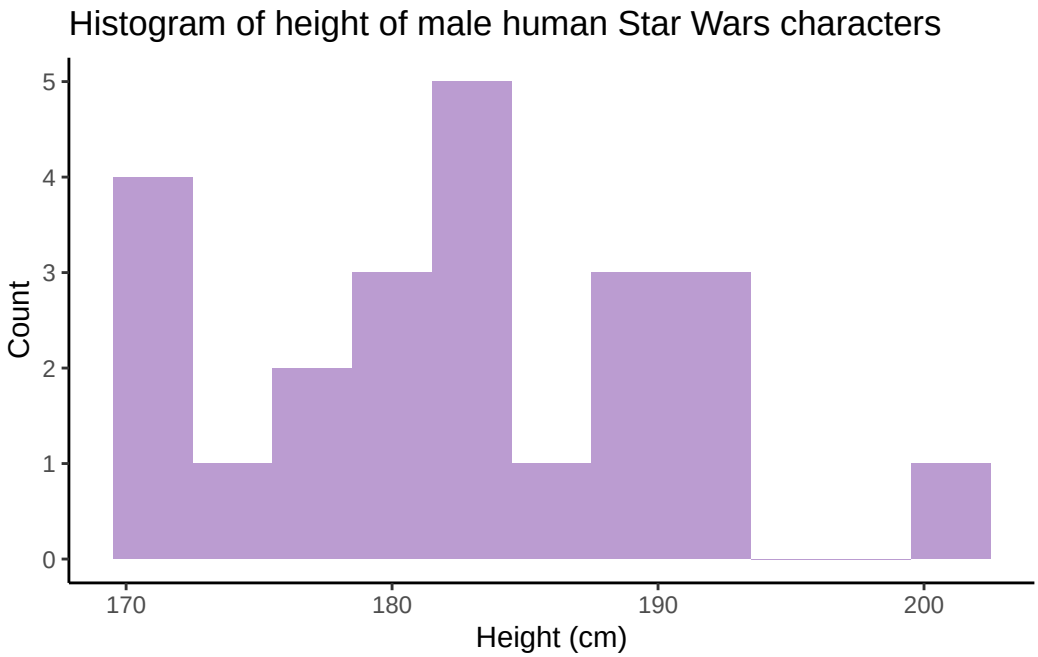
Height vs Weight by Sex

```
starwars |>
  filter(species == "Human") |>
  ggplot(aes(x = height, y = mass, colour = sex)) +
  geom_point() +
  theme_classic() +
  scale_x_continuous(limits = c(0,250)) +
  scale_y_continuous(limits = c(0,150)) +
  scale_colour_brewer(palette = "Accent", name = "Sex") +
  theme(legend.position = "bottom") +
  labs(x = "Height (cm)",
       y = "Weight (kg)",
       title = "Height and Weight of Human Characters in Star Wars by Sex")
```



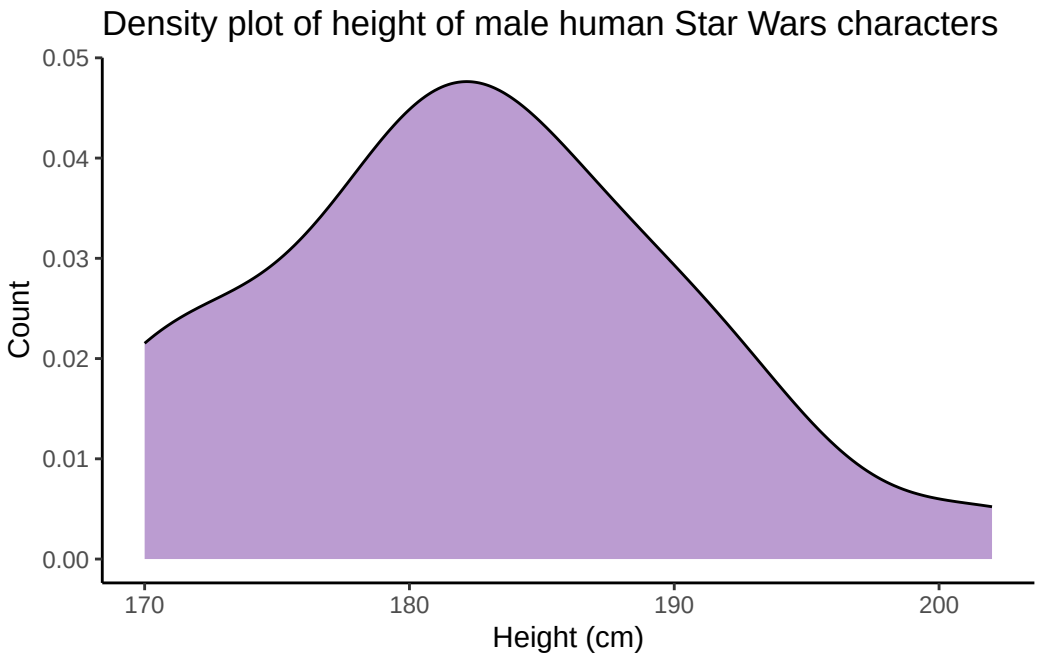
Histogram of male height

```
starwars |>
  filter(species == "Human",
         sex == "male") |>
  ggplot(aes(x = height)) +
  geom_histogram(binwidth = 3, fill = "#bb9cd1") +
  theme_classic() +
  labs(x = "Height (cm)",
       y = "Count",
       title = "Histogram of height of male human Star Wars characters")
```



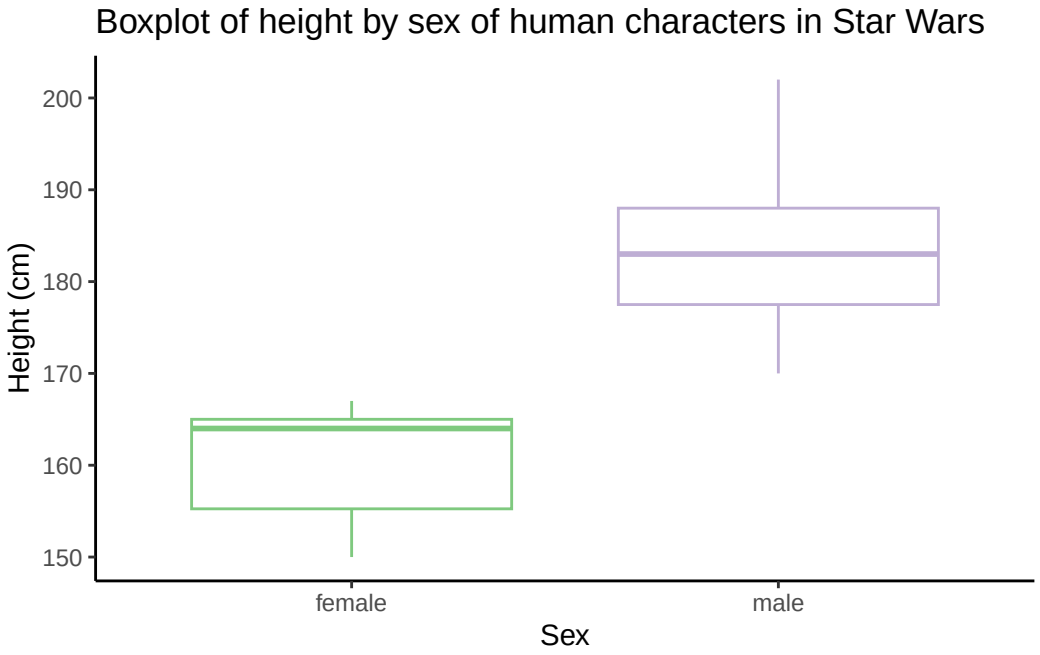
Density plot of male height

```
starwars |>
  filter(species == "Human",
         sex == "male") |>
  ggplot(aes(x = height)) +
  geom_density(fill = "#bb9cd1") +
  theme_classic() +
  labs(x = "Height (cm)",
       y = "Count",
       title = "Density plot of height of male human Star Wars characters")
```



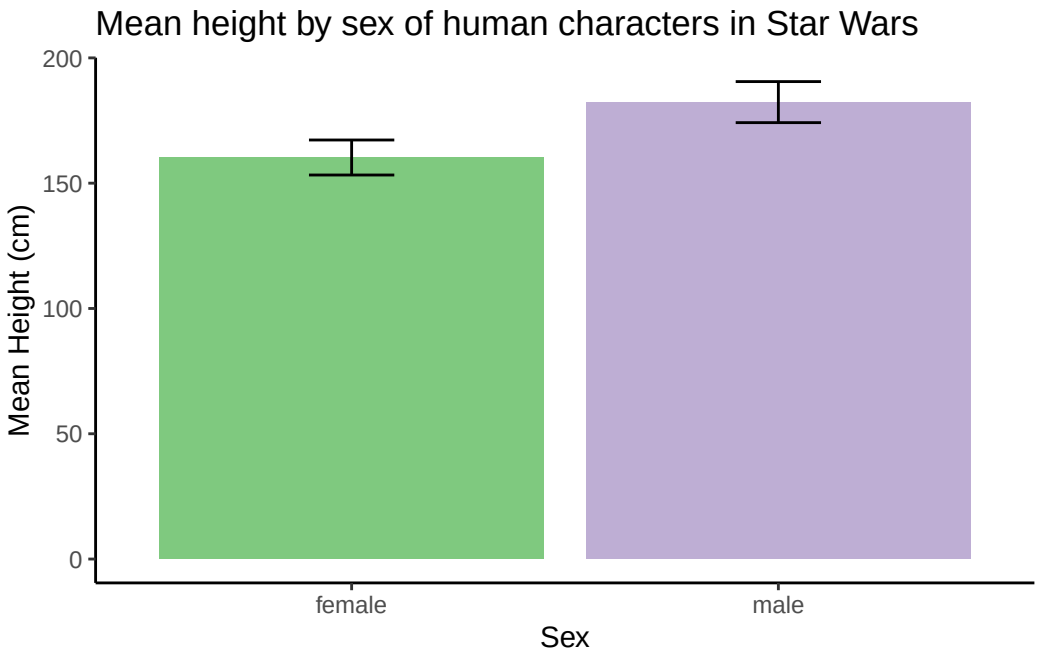
Boxplot of height

```
starwars |>
  filter(species == "Human") |>
  ggplot(aes(y = height, x = sex, colour = sex)) +
  geom_boxplot() +
  theme_classic() +
  scale_colour_brewer(palette = "Accent", name = "Sex") +
  labs(y = "Height (cm)",
       x = "Sex",
       title = "Boxplot of height by sex of human characters in Star Wars") +
  theme(legend.position ="none")
```



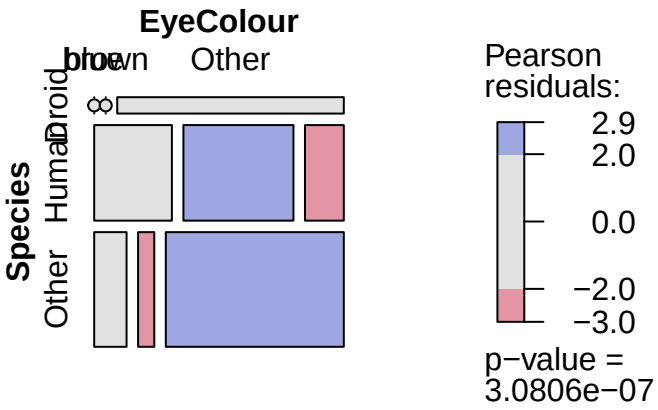
Mean height (bar chart)

```
starwars |>
  filter(species == "Human") |>
  group_by(sex) |>
  summarise(ht = mean(height, na.rm = TRUE),
            sd = sd(height, na.rm = TRUE)) |>
  ggplot(aes(x = sex, y = ht, fill = sex)) +
  geom_bar(stat = "identity") +
  geom_errorbar(aes(ymin = ht-sd, ymax = ht+sd), width = 0.2)+
  scale_fill_brewer(palette = "Accent") +
  labs(y = "Mean Height (cm)",
       x = "Sex",
       title = "Mean height by sex of human characters in Star Wars") +
  theme_classic() +
  theme(legend.position = "none")
```



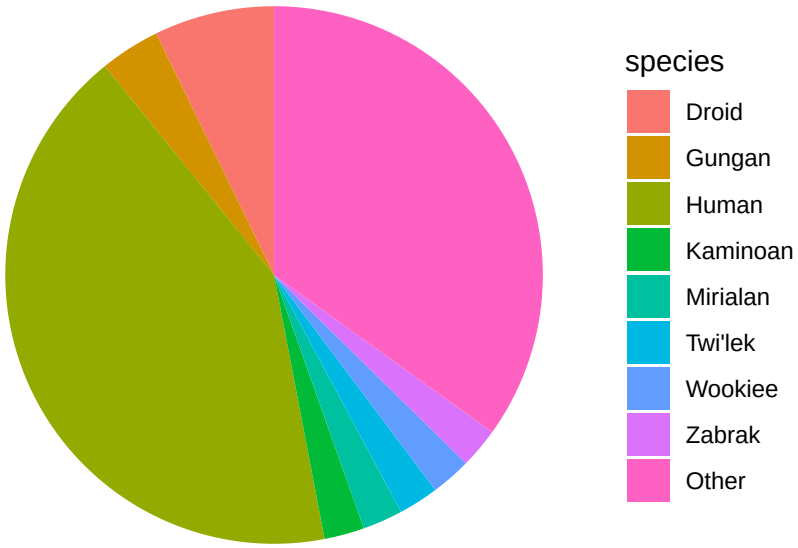
Mosaic Plot

```
library(vcd)


startbl <-  starwars |>
  mutate(Species = fct_lump_n(species, 2),
         EyeColour = fct_lump_n(eye_color,2))


mosaic(~ Species + EyeColour, data = startbl,shade = TRUE, legend = TRUE)
```

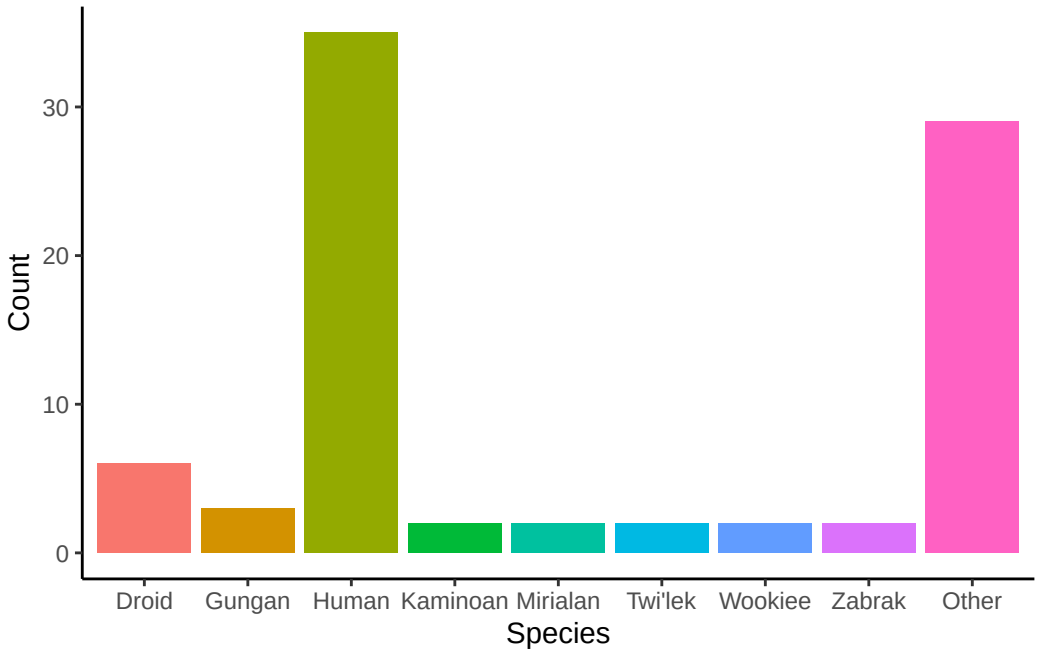


Pie Charts are Just Bad Bar Charts

```
starwars |>
  mutate(species = fct_lump_n(species, 4)) |>
  group_by(species) |>
  filter(!is.na(species)) |>
  tally() |>
  ggplot(aes(x = "", fill = species, y = n)) +
  geom_bar(stat = "identity", width = 1) +
  theme_void() +
  coord_polar("y", start = 0)
```




```
starwars |>
  mutate(species = fct_lump_n(species, 4)) |>
  group_by(species) |>
  filter(!is.na(species)) |>
  tally() |>
  ggplot(aes(x = species, fill = species, y = n)) +
  geom_bar(stat = "identity") +
  theme_classic() +
  labs(x = "Species", y = "Count") +
  theme(legend.position = "none")
```

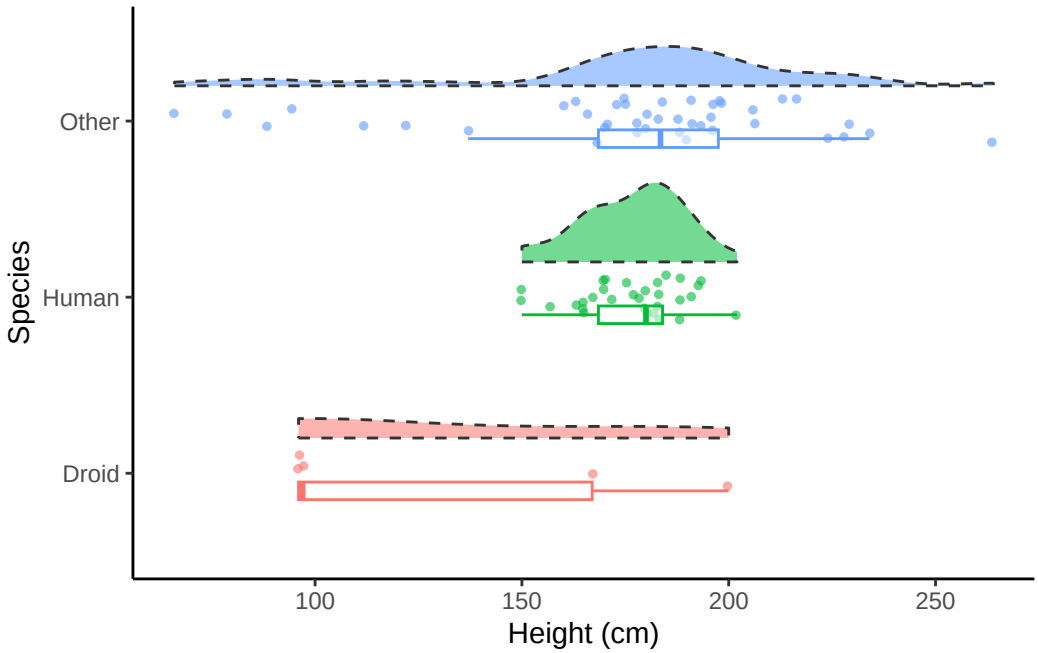


Wordclouds

```
library(wordcloud)
starwars |>
  count(homeworld) |>
  with(wordcloud(words = homeworld, freq = n, min.freq=1, random.order = FALSE, rot.per =
                 colors = brewer.pal(6, "Accent"), use.r.layout = FALSE))
```

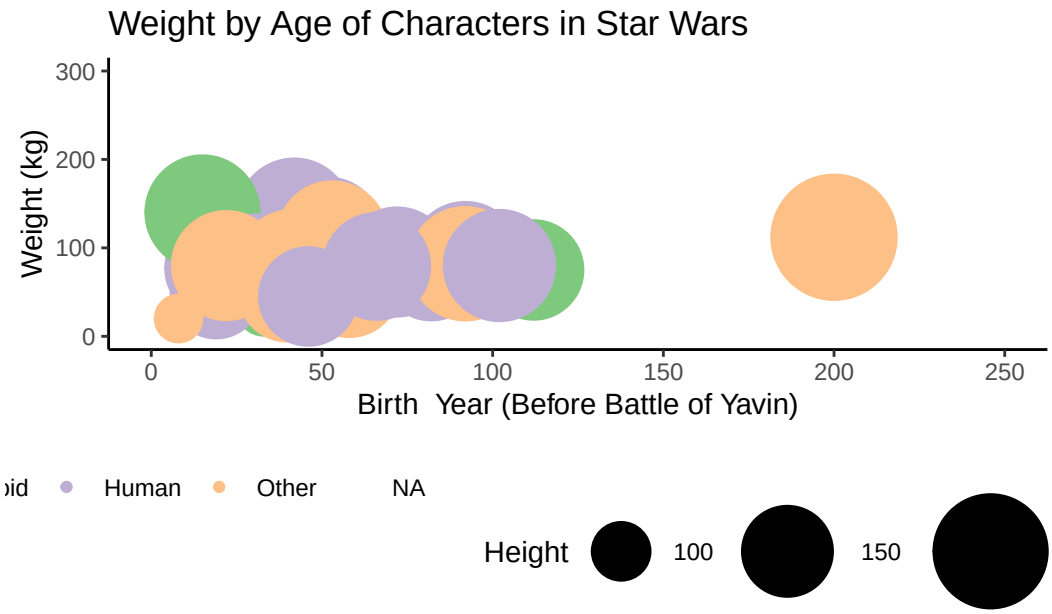
Raincloud Plots

```
starwars |>
  mutate(species = fct_lump_n(species,2)) |>
  filter(!is.na(species)) |>
  ggplot(aes(x = species)) +
  geom_point(aes(y = height, colour = species), position = position_jitter(width = .13), s
  see::geom_violinhalf(aes(y = height, alpha= 0.3, fill = species), linetype = "dashed", p
  geom_boxplot(aes(y = height, alpha = 0.3, colour = species), position = position_nudge(x
  theme_classic() +
  labs(x = "Species", y = "Height (cm)") +
  theme(legend.position = "none") +
  coord_flip()
```



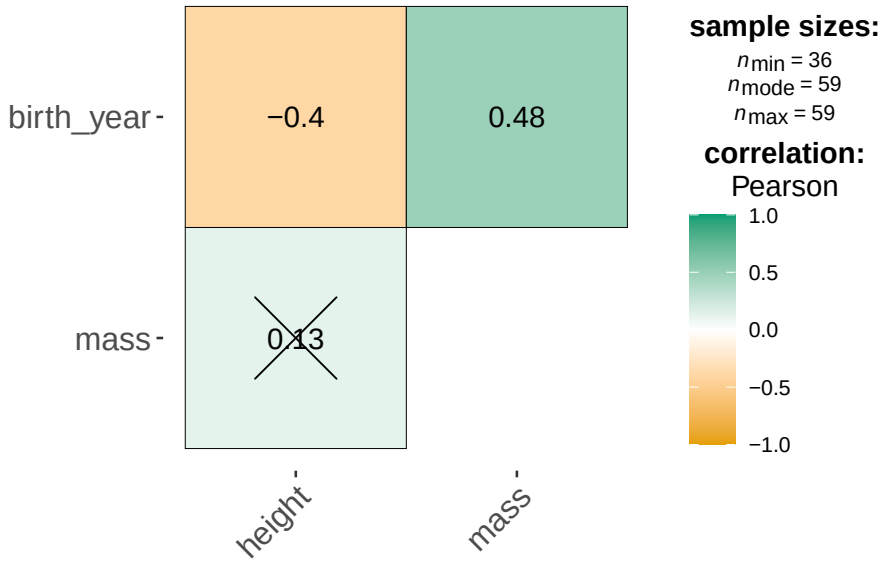
Bubble plots

```
starwars |>
  mutate(col = fct_lump_n(species, 2)) |>
  ggplot(aes(x = birth_year, y = mass, size = height, colour = col)) +
  geom_point() +
  scale_size(range = c(.1, 24), name="Height") +
  theme_classic() +
  scale_x_continuous(limits = c(0,250)) +
  scale_y_continuous(limits = c(0,300)) +
  scale_colour_brewer(palette = "Accent", name = "Species") +
  theme(legend.position = "bottom") +
  labs(x = "Birth  Year (Before Battle of Yavin)",
       y = "Weight (kg)",
       title = "Weight by Age of Characters in Star Wars")
```



Correlation plots

```
starwars |>
  select(height, mass, birth_year) |>
  ggcorrmat()
```



X = non−significant at p < 0.05 (Adjustment: Holm)

Lecture 3: The Mean as a Basic Model

Data and custom function for this lecture

```
library(tidyverse)

heifers <- tibble(heifers = c(211.3, 200.4, 220.1, 200.8, 222.0, 209.3,
              195.8, 220.4, 226.2, 218.7, 193.7, 209.7))



wage <- readxl::read_excel("assets/UKWageData2023ONS.xlsx",
                           skip = 5)

find_mode <- function(x) {
  ux <- unique(x)
  tab <- tabulate(match(x, ux))
  ux[tab == max(tab)]
}
```

Finding central tendency

```
heifers |>
  summarise(mean = mean(heifers),
            median = median(heifers),
            min = min(heifers),
            max = max(heifers),
            mode = find_mode(round(heifers, 0)))
```

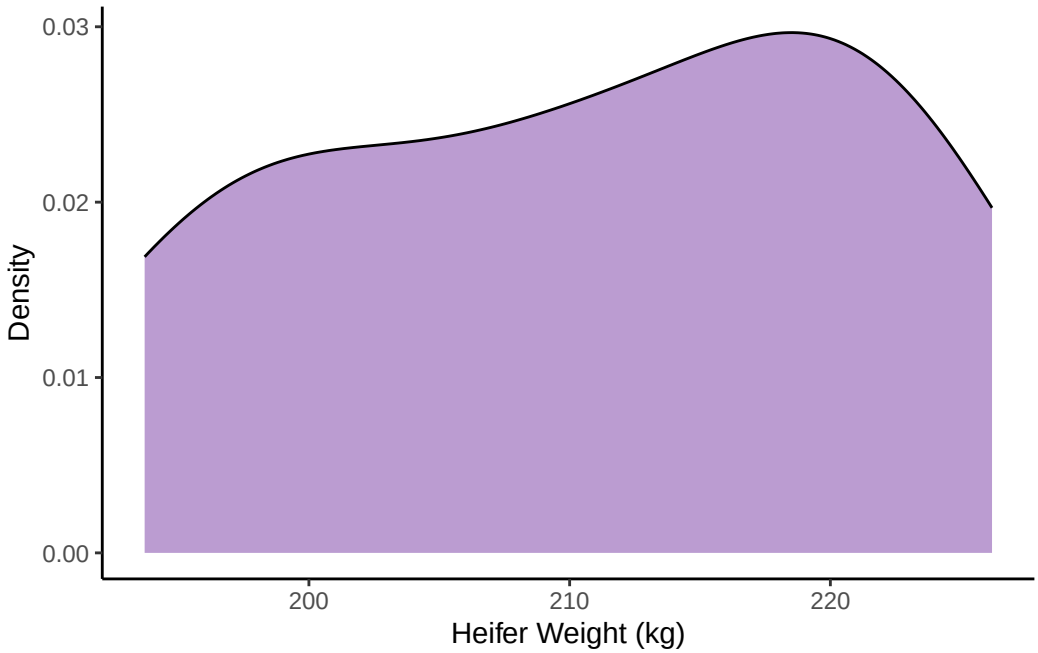
```
# A tibble: 1 x 5
   mean median   min   max  mode
  <dbl>  <dbl> <dbl> <dbl> <dbl>
1  211.   210.  194.  226.   220
```

```r
wage |>
  summarise(mean = mean(Median),
            median = median(Median),
            min = min(Median),
            max = max(Median),
            mode = find_mode(round(Median,0)))
```

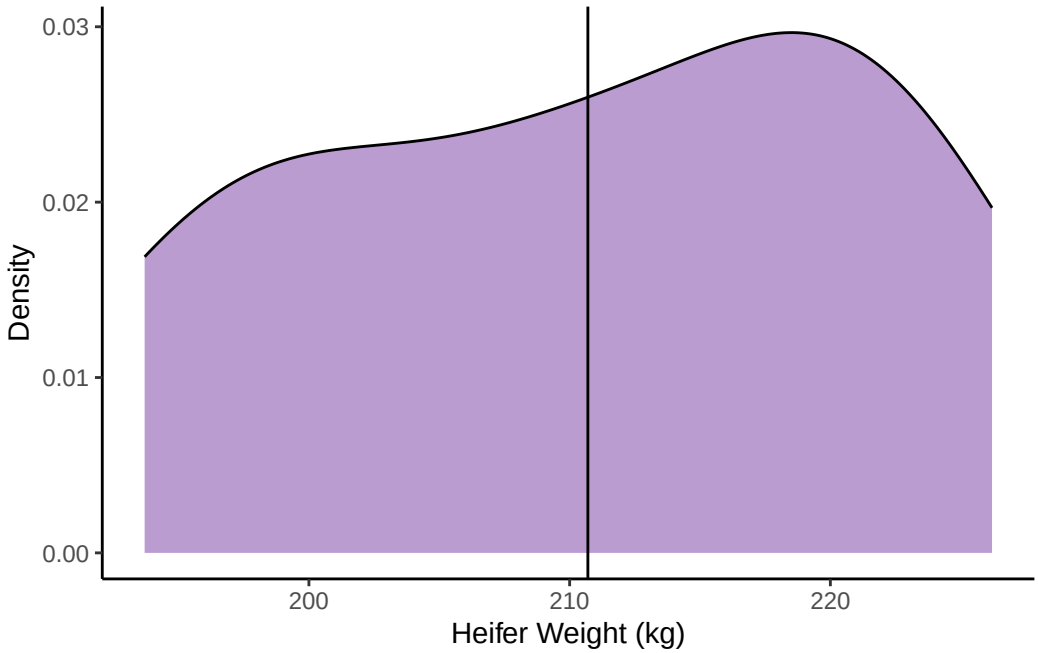
```
# A tibble: 4 x 5
    mean median   min   max  mode
   <dbl>  <dbl> <dbl> <dbl> <dbl>
1 34475.  31988 17859 84131 28216
2 34475.  31988 17859 84131 35248
3 34475.  31988 17859 84131 26000
4 34475.  31988 17859 84131 25000
```

The Mean and Outliers

```r
heifers |>
  ggplot(aes(x = heifers)) +
  geom_density(fill = "#bb9cd1") +
  theme_classic() +
  labs(x = "Heifer Weight (kg)",
       y = "Density")
```

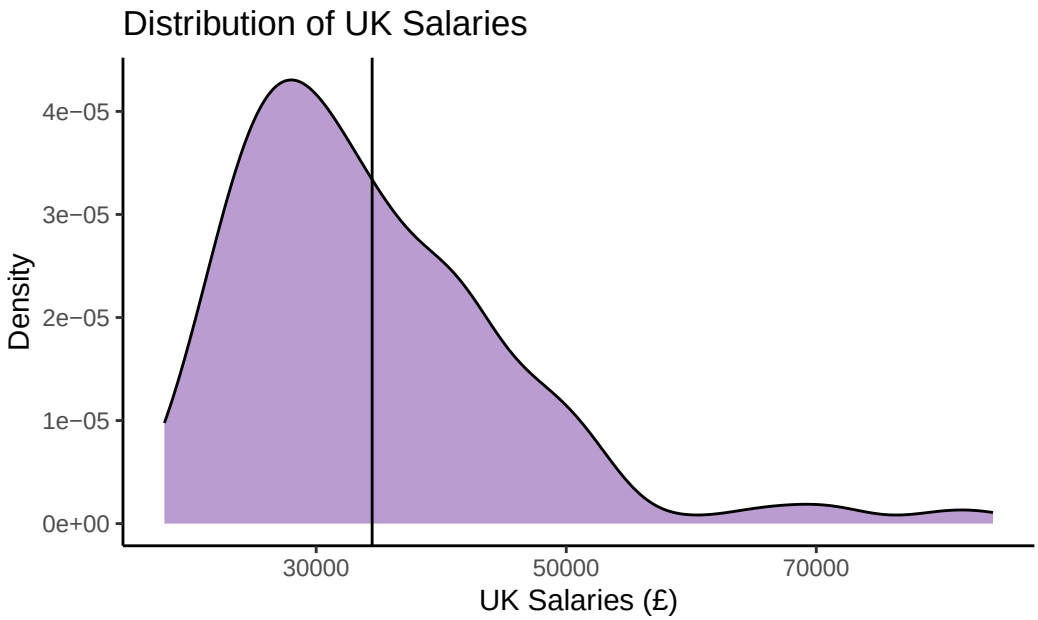



```
heifers |>
  ggplot(aes(x = heifers)) +
  geom_density(fill = "#bb9cd1") +
  geom_vline(aes(xintercept = 210.7)) +
  theme_classic() +
  labs(x = "Heifer Weight (kg)",
       y = "Density")
```



Mean UK Salary

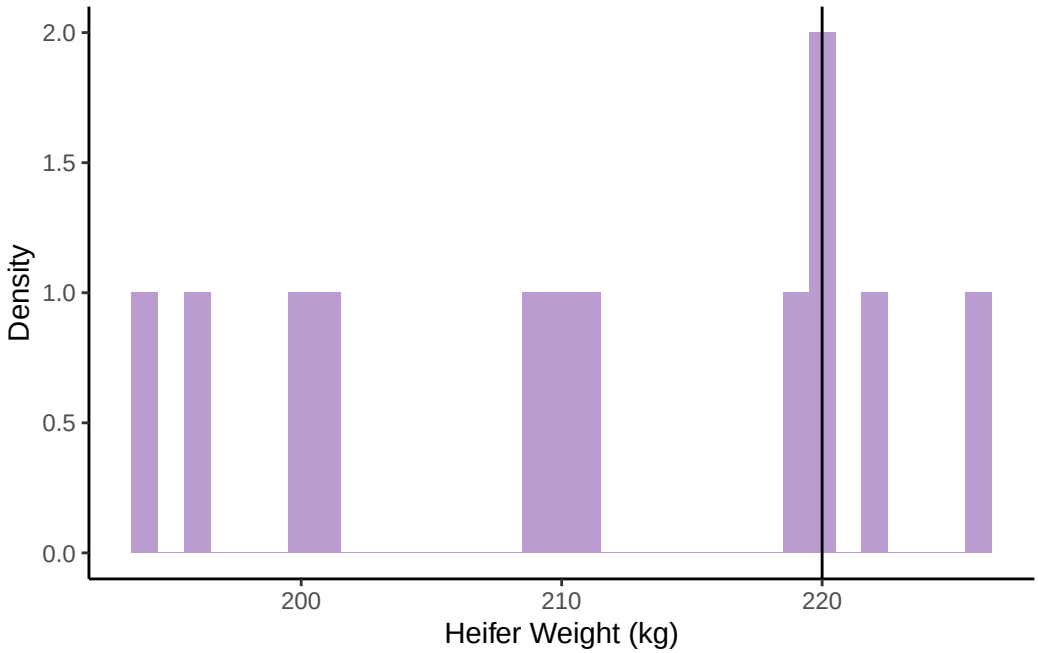
```
wage |>
  ggplot(aes(x = Median)) +
  geom_density(fill = "#bb9cd1") +
  theme_classic() +
  geom_vline(aes(xintercept = 34475)) +
  labs(x = "UK Salaries (£)",
       y = "Density",
       title = "Distribution of UK Salaries",
       caption = "Data taken from ONS 2023 Median Salaries by Field, n = 329 fields")
```



Data taken from ONS 2023 Median Salaries by Field, n = 329 fields

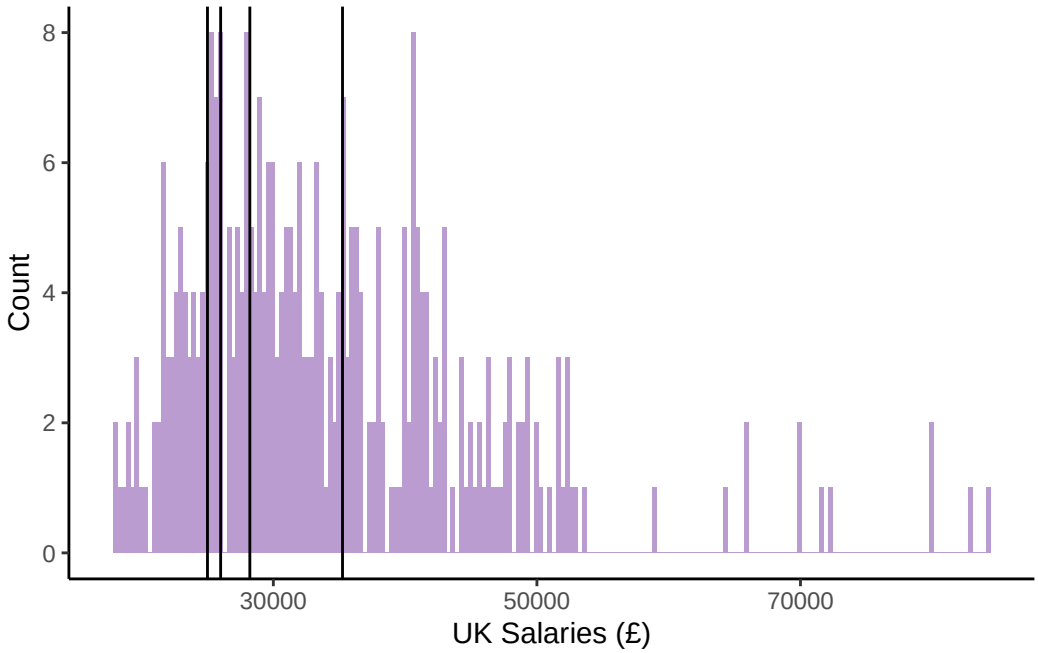
The Mode

```
heifers |>
  ggplot(aes(x = heifers)) +
  geom_histogram(fill = "#bb9cd1", binwidth = 1) +
  geom_vline(aes(xintercept = 220)) +
  theme_classic() +
  labs(x = "Heifer Weight (kg)",
       y = "Density")
```



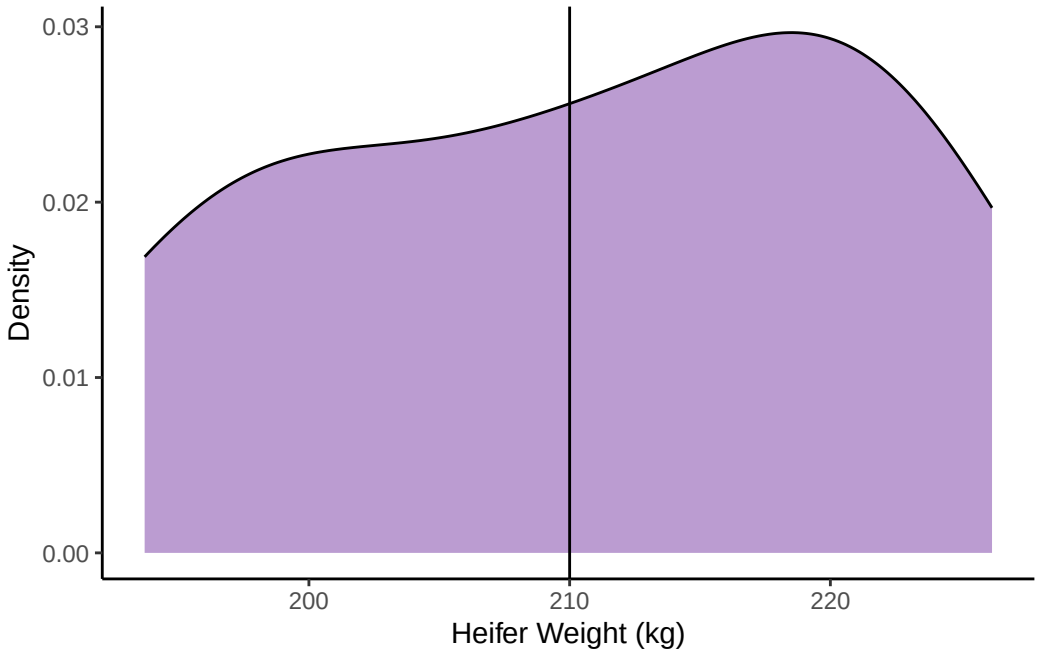
Multiple Modes

```
wage |>
  ggplot(aes(x = Median)) +
  geom_histogram(fill = "#bb9cd1", bins = 200) +
  geom_vline(aes(xintercept = 25000)) +
  geom_vline(aes(xintercept = 26000)) +
  geom_vline(aes(xintercept = 28216)) +
  geom_vline(aes(xintercept = 35248)) +
  theme_classic() +
  labs(x = "UK Salaries (£)",
       y = "Count")
```



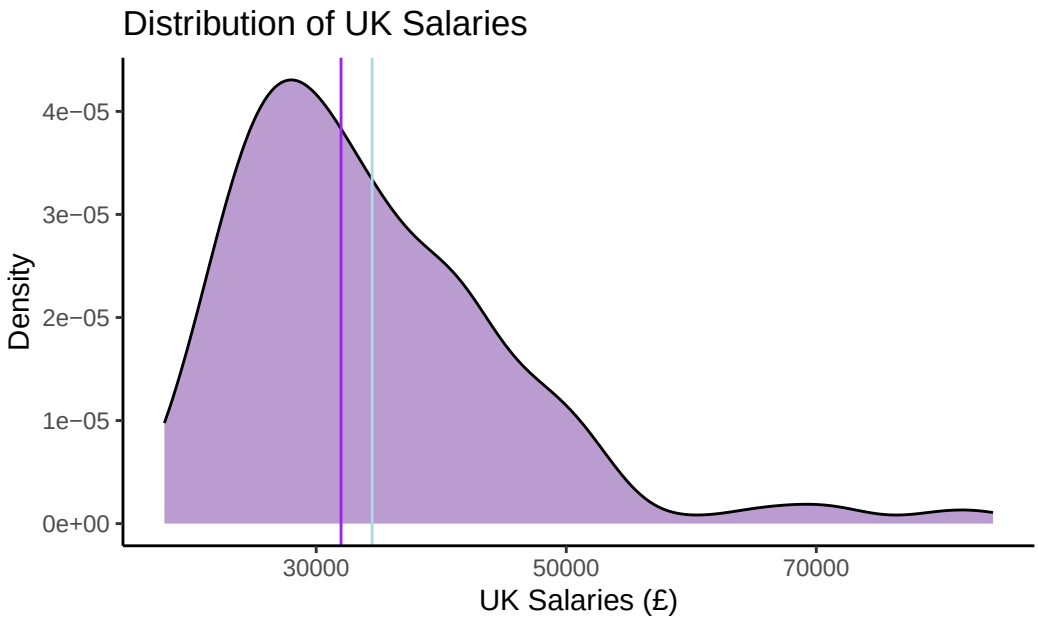
The Median

```
heifers |>
  ggplot(aes(x = heifers)) +
  geom_density(fill = "#bb9cd1") +
  geom_vline(aes(xintercept = 210)) +
  theme_classic() +
  labs(x = "Heifer Weight (kg)",
       y = "Density")
```



Median UK Salary

```
wage |>
  ggplot(aes(x = Median)) +
  geom_density(fill = "#bb9cd1") +
  theme_classic() +
  geom_vline(aes(xintercept = 34475), colour = "lightblue") +
  geom_vline(aes(xintercept = 31988),colour = "purple") +
  labs(x = "UK Salaries (£)",
       y = "Density",
       title = "Distribution of UK Salaries",
       caption = "Data taken from ONS 2023 Median Salaries by Field, n = 329 fields")
```



Data taken from ONS 2023 Median Salaries by Field, n = 329 fields

4 Week 3: Introduction to Analyses

Lecture 2: Introduction to statistics

Set up your environment and packages

```
library(tidyverse)
library(easystats)
library(rstan)
library(rstanarm)

cat_weights <- tibble(avg_daily_snacks  = c(3, 2, 4, 2, 3, 1, 1, 0, 1, 0, 2, 3, 1, 2, 1, 3
                      weight = c(3.8, 3.9, 5, 3.7,  4.1, 3.6, 3.7, 3.6, 3.8, 4.1, 4.3, 3.9
                      environ = c("Indoor", "Indoor", "Outdoor", "Indoor",
                                  "Outdoor", "Indoor", "Outdoor", "Indoor",
                                  "Indoor", "Indoor", "Outdoor", "Indoor",
                                  "Outdoor", "Indoor", "Indoor", "Outdoor"))
```

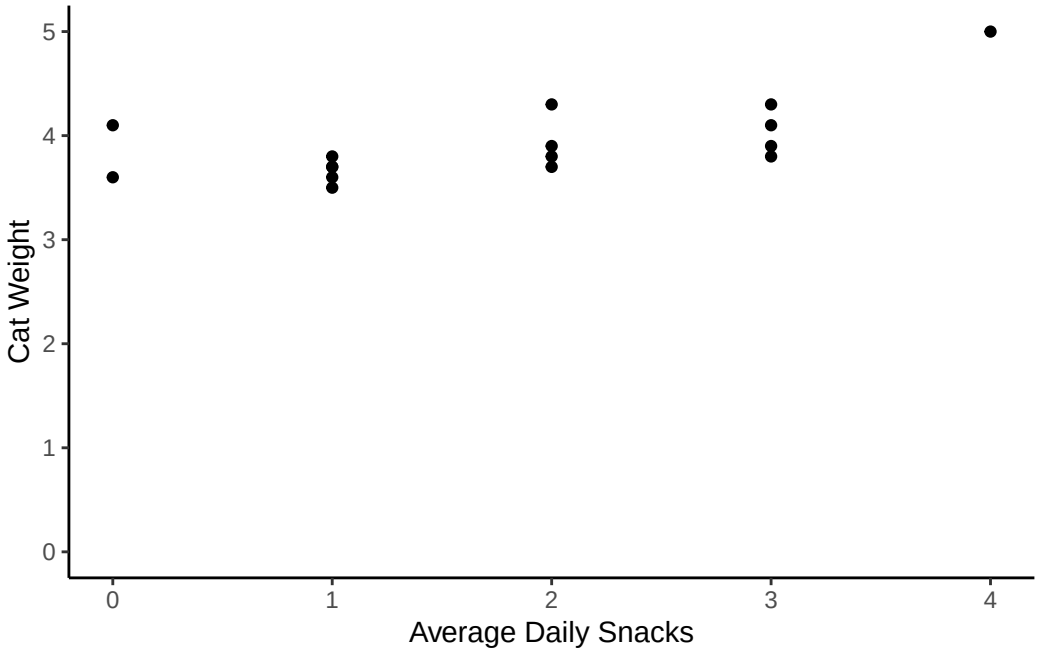
Summarise example data

```
cat_weights |>
  summarise("Mean Weight (kg)" = mean(weight),
            "SD Weight (kg)" = sd(weight),
            "Mean Daily Snacks" = mean (avg_daily_snacks),
            )
```

```
# A tibble: 1 x 3
  `Mean Weight (kg)` `SD Weight (kg)` `Mean Daily Snacks`
               <dbl>            <dbl>               <dbl>
1               3.92            0.373                1.81
```

Visualise example data

```
cat_weights |>
  ggplot(aes(x = avg_daily_snacks, y = weight)) +
  geom_point() +
  labs(x = "Average Daily Snacks", y = "Cat Weight") +
  theme_classic() +
  scale_y_continuous(limits = c(0,5))
```



A Linear Model

```
model_fcat <- lm(weight ~ avg_daily_snacks, data = cat_weights)
summary(model_fcat)
```

```
Call:
lm(formula = weight ~ avg_daily_snacks, data = cat_weights)

Residuals:
     Min       1Q   Median       3Q      Max
```

```
-0.36758 -0.18723 -0.06116  0.06705  0.62813

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)       3.55474    0.14045  25.309 4.33e-13 ***
avg_daily_snacks  0.20428    0.06576   3.107  0.00773 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.2973 on 14 degrees of freedom
Multiple R-squared:  0.4081,    Adjusted R-squared:  0.3658
F-statistic: 9.652 on 1 and 14 DF,  p-value: 0.007729
```

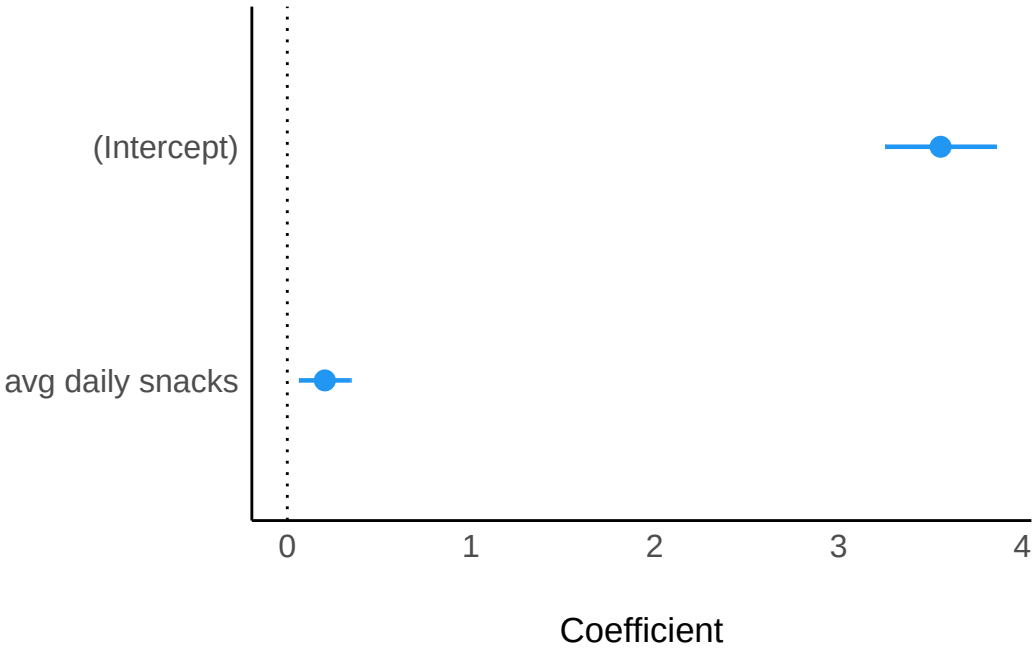
```
report::report(model_fcat)
```

```
We fitted a linear model (estimated using OLS) to predict weight with
avg_daily_snacks (formula: weight ~ avg_daily_snacks). The model explains a
statistically significant and substantial proportion of variance (R2 = 0.41,
F(1, 14) = 9.65, p = 0.008, adj. R2 = 0.37). The model's intercept,
corresponding to avg_daily_snacks = 0, is at 3.55 (95% CI [3.25, 3.86], t(14) =
25.31, p < .001). Within this model:

  - The effect of avg daily snacks is statistically significant and positive
(beta = 0.20, 95% CI [0.06, 0.35], t(14) = 3.11, p = 0.008; Std. beta = 0.64,
95% CI [0.20, 1.08])

Standardized parameters were obtained by fitting the model on a standardized
version of the dataset. 95% Confidence Intervals (CIs) and p-values were
computed using a Wald t-distribution approximation.
```

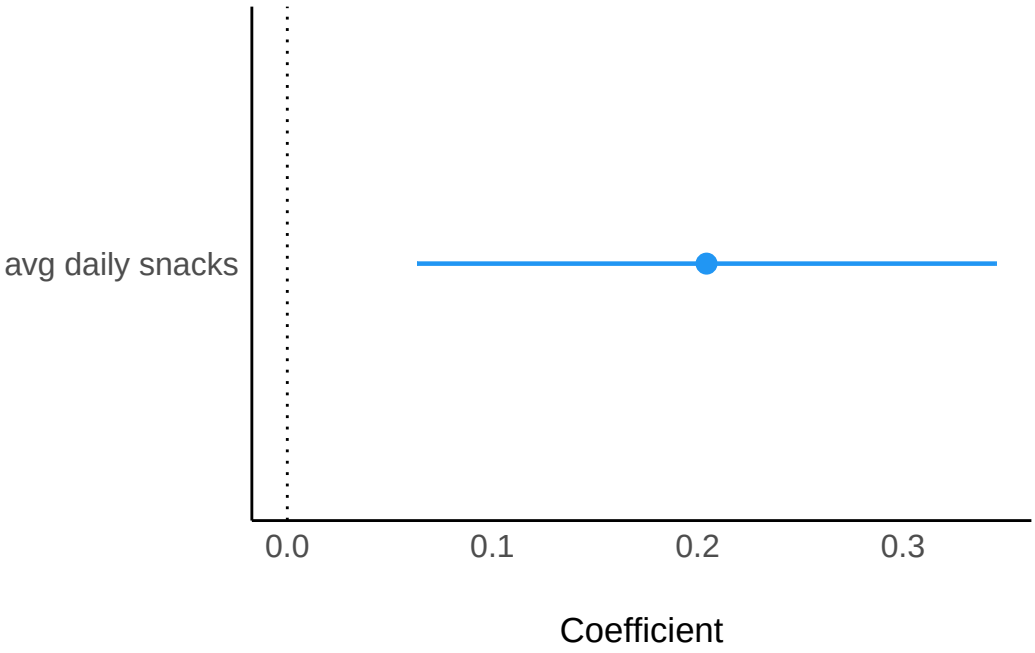
```
parameters(model_fcat)
```

```
Parameter        | Coefficient |   SE |       95% CI | t(14) |      p
-------------------------------------------------------------------------
(Intercept)      |        3.55 | 0.14 | [3.25, 3.86] | 25.31 | < .001
avg daily snacks |        0.20 | 0.07 | [0.06, 0.35] |  3.11 | 0.008
```

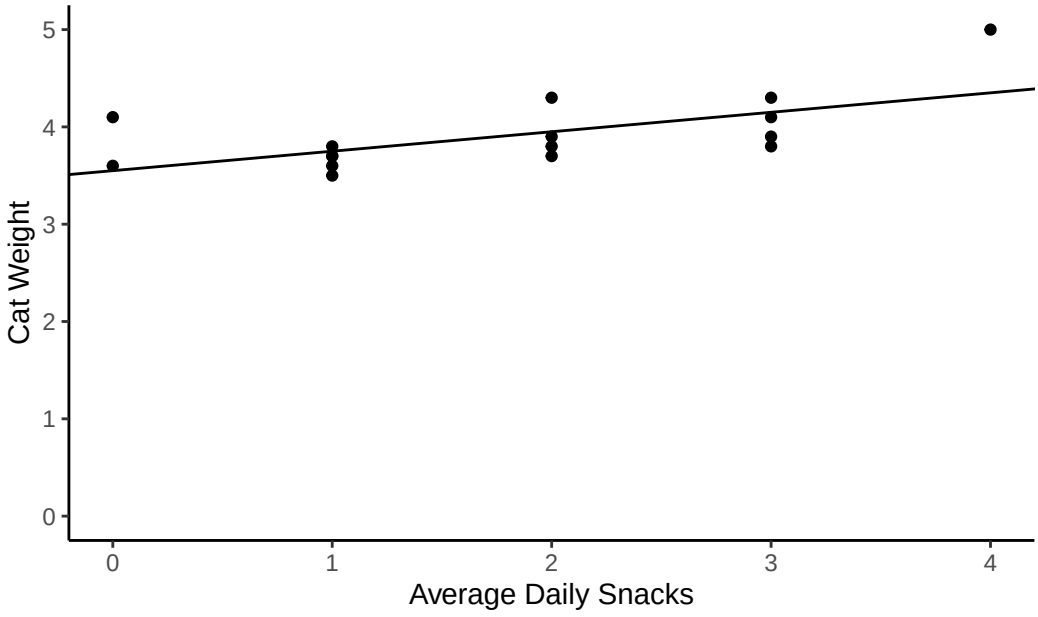
```
plot(model_parameters(model_fcat), show_intercept = TRUE)
```




```
plot(model_parameters(model_fcat))
```



```r
cat_weights |>
  ggplot(aes(x = avg_daily_snacks, y = weight)) +
  geom_point() +
  labs(x = "Average Daily Snacks", y = "Cat Weight",
       caption = "Weight ~ Average Daily Snacks shown") +
  theme_classic() +
  scale_y_continuous(limits = c(0,5)) +
  geom_abline(slope = 0.20, intercept = 3.55)
```



Weight ~ Average Daily Snacks shown

A Bayesian Model

```r
set.seed(10)

model_bcat <- stan_glm(weight ~ avg_daily_snacks, data = cat_weights)
```

```
SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.002048 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 20.48 seconds.
```

```
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 0.053 seconds (Warm-up)
Chain 1:                0.04 seconds (Sampling)
Chain 1:                0.093 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 1.4e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.14 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 0.037 seconds (Warm-up)
Chain 2:                0.042 seconds (Sampling)
```

```
Chain 2:                0.079 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 1.4e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.14 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 0.038 seconds (Warm-up)
Chain 3:                0.038 seconds (Sampling)
Chain 3:                0.076 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 1.1e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.11 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
```

```
Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 0.039 seconds (Warm-up)
Chain 4:                0.039 seconds (Sampling)
Chain 4:                0.078 seconds (Total)
Chain 4:
```

```r
summary(model_bcat)
```

```
Model Info:
 function:     stan_glm
 family:       gaussian [identity]
 formula:      weight ~ avg_daily_snacks
 algorithm:    sampling
 sample:       4000 (posterior sample size)
 priors:       see help('prior_summary')
 observations: 16
 predictors:   2

Estimates:
                   mean   sd   10%   50%   90%
(Intercept)      3.6    0.2  3.4   3.6   3.7
avg_daily_snacks 0.2    0.1  0.1   0.2   0.3
sigma            0.3    0.1  0.2   0.3   0.4

Fit Diagnostics:
           mean   sd   10%   50%   90%
mean_PPD 3.9    0.1  3.8   3.9   4.1

The mean_ppd is the sample average posterior predictive distribution of the outcome variable

MCMC diagnostics
                 mcse Rhat n_eff
(Intercept)      0.0  1.0  3066
avg_daily_snacks 0.0  1.0  2981
sigma            0.0  1.0  2539
mean_PPD         0.0  1.0  3573
```

```
log-posterior    0.0  1.0  1450

For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective
```

```
describe_posterior(model_bcat)
```

```
Summary of Posterior Distribution

Parameter        | Median |       95% CI |     pd |          ROPE | % in ROPE |  Rhat |     E
-------------------------------------------------------------------------------------------
(Intercept)      |   3.55 | [3.26, 3.86] |   100% | [-0.04, 0.04] |        0% | 1.001 | 3066
avg_daily_snacks |   0.20 | [0.06, 0.35] | 99.62% | [-0.04, 0.04] |        0% | 1.000 | 2981
```

```
report::report(model_bcat)
```

```
SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 1.4e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.14 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 0.039 seconds (Warm-up)
Chain 1:                0.051 seconds (Sampling)
Chain 1:                0.09 seconds (Total)
Chain 1:
```

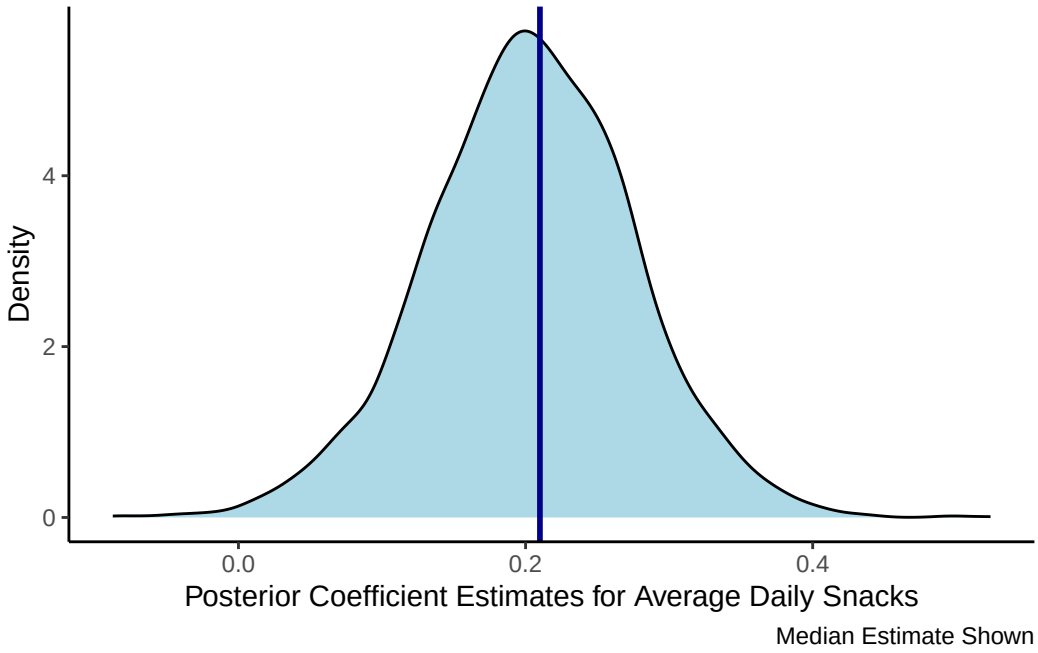
```
SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 1.2e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.12 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 0.037 seconds (Warm-up)
Chain 2:                0.047 seconds (Sampling)
Chain 2:                0.084 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 9e-06 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.09 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
```

```
Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 0.039 seconds (Warm-up)
Chain 3:                0.05 seconds (Sampling)
Chain 3:                0.089 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 1.2e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.12 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 0.039 seconds (Warm-up)
Chain 4:                0.058 seconds (Sampling)
Chain 4:                0.097 seconds (Total)
Chain 4:


We fitted a Bayesian linear model (estimated using MCMC sampling with 4 chains
of 2000 iterations and a warmup of 1000) to predict weight with
avg_daily_snacks (formula: weight ~ avg_daily_snacks). Priors over parameters
were set as normal (mean = 0.00, SD = 0.80) distributions. The model's
explanatory power is substantial (R2 = 0.38, 95% CI [6.54e-06, 0.62], adj. R2 =
0.15). The model's intercept, corresponding to avg_daily_snacks = 0, is at 3.55
(95% CI [3.26, 3.86]). Within this model:

  - The effect of avg daily snacks (Median = 0.20, 95% CI [0.06, 0.35]) has a
```

99.62% probability of being positive (> 0), 99.30% of being significant (> 0.02), and 90.10% of being large (> 0.11). The estimation successfully converged (Rhat = 1.000) and the indices are reliable (ESS = 2981)

Following the Sequential Effect eXistence and sIgnificance Testing (SEXIT) framework, we report the median of the posterior distribution and its 95% CI (Highest Density Interval), along the probability of direction (pd), the probability of significance and the probability of being large. The thresholds beyond which the effect is considered as significant (i.e., non-negligible) and large are |0.02| and |0.11| (corresponding respectively to 0.05 and 0.30 of the outcome's SD). Convergence and stability of the Bayesian sampling has been assessed using R-hat, which should be below 1.01 (Vehtari et al., 2019), and Effective Sample Size (ESS), which should be greater than 1000 (Burkner, 2017).

```
posteriors <- get_parameters(model_bcat)

posteriors |>
  ggplot(aes(x = avg_daily_snacks)) +
  geom_density(fill = "lightblue") +
  theme_classic() +
  labs(x = "Posterior Coefficient Estimates for Average Daily Snacks",
       y = "Density",
       caption = "Median Estimate Shown") +
  geom_vline(xintercept = 0.21, color = "darkblue", linewidth = 1)
```



A Linear model with a factor

```
model_fcat2 <- lm(weight ~ avg_daily_snacks + environ, data = cat_weights)
summary(model_fcat2)
```

```
Call:
lm(formula = weight ~ avg_daily_snacks + environ, data = cat_weights)

Residuals:
    Min      1Q  Median      3Q     Max
-0.2678 -0.1897 -0.0700  0.0821  0.5725

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)       3.52748    0.13078  26.972 8.49e-13 ***
avg_daily_snacks  0.16168    0.06512   2.483   0.0275 *
environOutdoor    0.27860    0.15203   1.833   0.0899 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 0.275 on 13 degrees of freedom
Multiple R-squared:  0.5296,    Adjusted R-squared:  0.4572
F-statistic: 7.318 on 2 and 13 DF,  p-value: 0.007431
```

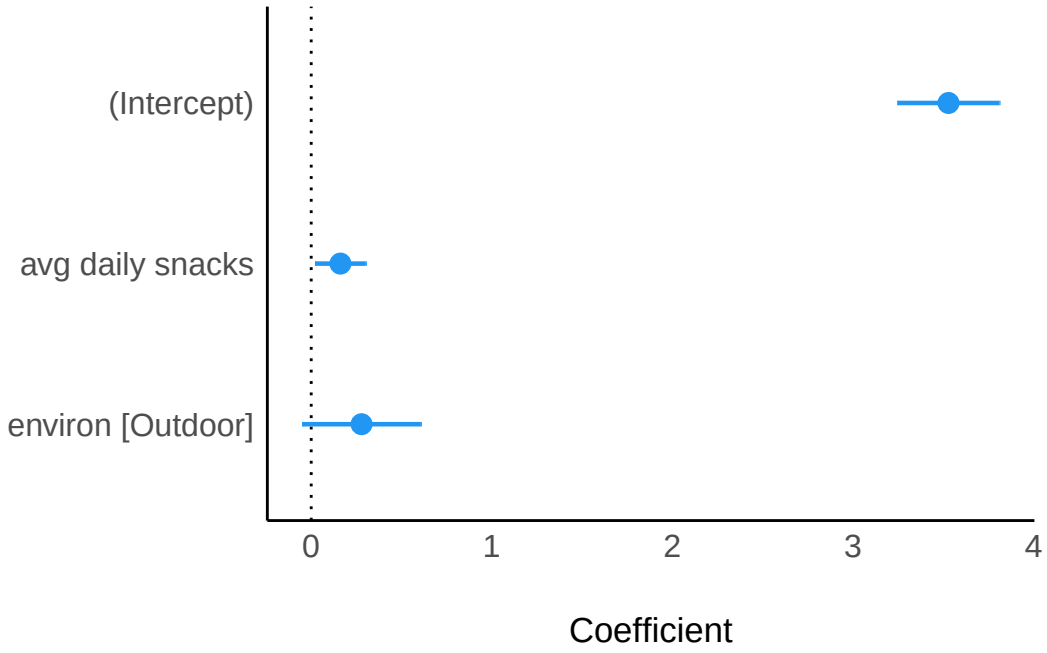
```
report::report(model_fcat2)
```

```
We fitted a linear model (estimated using OLS) to predict weight with
avg_daily_snacks and environ (formula: weight ~ avg_daily_snacks + environ).
The model explains a statistically significant and substantial proportion of
variance (R2 = 0.53, F(2, 13) = 7.32, p = 0.007, adj. R2 = 0.46). The model's
intercept, corresponding to avg_daily_snacks = 0 and environ = Indoor, is at
3.53 (95% CI [3.24, 3.81], t(13) = 26.97, p < .001). Within this model:

  - The effect of avg daily snacks is statistically significant and positive
(beta = 0.16, 95% CI [0.02, 0.30], t(13) = 2.48, p = 0.027; Std. beta = 0.51,
95% CI [0.07, 0.95])
  - The effect of environ [Outdoor] is statistically non-significant and positive
(beta = 0.28, 95% CI [-0.05, 0.61], t(13) = 1.83, p = 0.090; Std. beta = 0.75,
95% CI [-0.13, 1.63])

Standardized parameters were obtained by fitting the model on a standardized
version of the dataset. 95% Confidence Intervals (CIs) and p-values were
computed using a Wald t-distribution approximation.
```

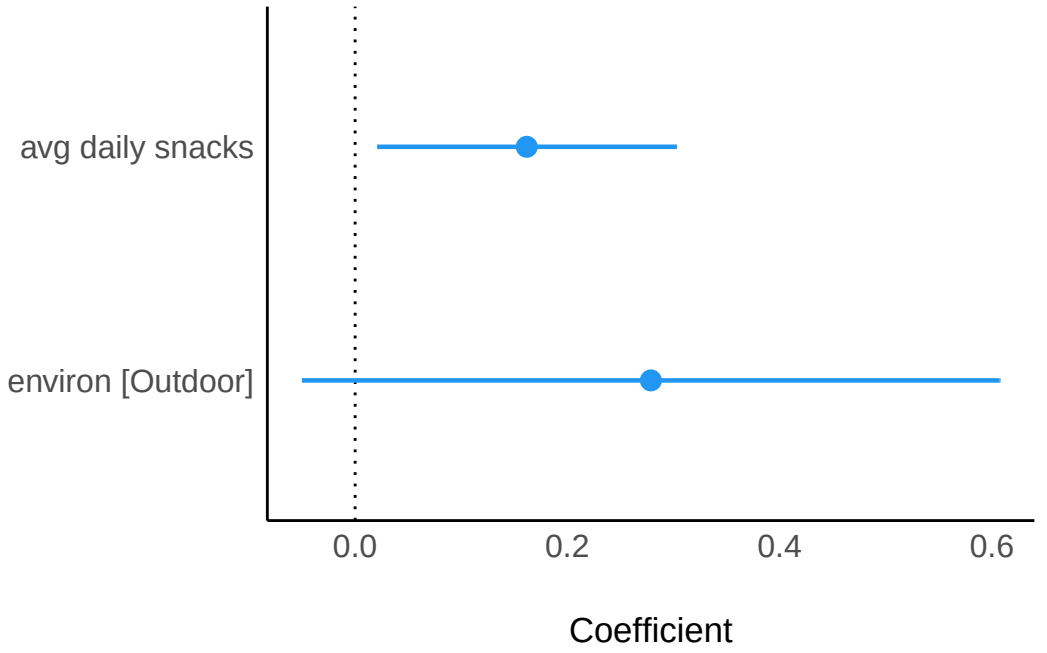
```
parameters(model_fcat2)
```

```
Parameter         | Coefficient |   SE |        95% CI | t(13) |      p
-------------------------------------------------------------------------
(Intercept)       |        3.53 | 0.13 | [ 3.24, 3.81] | 26.97 | < .001
avg daily snacks  |        0.16 | 0.07 | [ 0.02, 0.30] |  2.48 | 0.027
environ [Outdoor] |        0.28 | 0.15 | [-0.05, 0.61] |  1.83 | 0.090
```

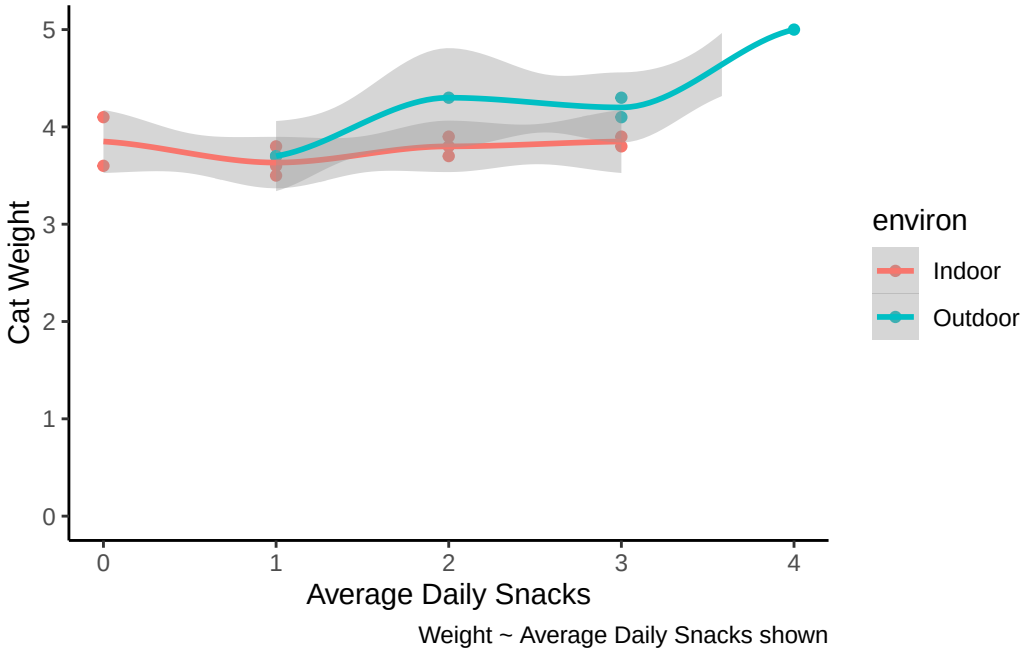
```
plot(model_parameters(model_fcat2), show_intercept = TRUE)
```




```
plot(model_parameters(model_fcat2))
```



```r
cat_weights |>
  ggplot(aes(x = avg_daily_snacks, y = weight, colour = environ)) +
  geom_point() +
  labs(x = "Average Daily Snacks", y = "Cat Weight",
       caption = "Weight ~ Average Daily Snacks shown") +
  theme_classic() +
  scale_y_continuous(limits = c(0,5)) +
  geom_smooth()
```



Bayesian Framework

```r
model_bcat2 <- stan_glm(weight ~ avg_daily_snacks + environ, data = cat_weights)
```

```
SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 1.8e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.18 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
```

```
Chain 1:
Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 0.059 seconds (Warm-up)
Chain 1:                0.065 seconds (Sampling)
Chain 1:                0.124 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 1.3e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.13 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 0.052 seconds (Warm-up)
Chain 2:                0.047 seconds (Sampling)
Chain 2:                0.099 seconds (Total)
Chain 2:
```

```
SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 1.1e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.11 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 0.047 seconds (Warm-up)
Chain 3:                0.042 seconds (Sampling)
Chain 3:                0.089 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 2.1e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.21 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
```

```
Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 0.058 seconds (Warm-up)
Chain 4:                0.046 seconds (Sampling)
Chain 4:                0.104 seconds (Total)
Chain 4:
```

```
summary(model_bcat2)
```

```
Model Info:
 function:     stan_glm
 family:       gaussian [identity]
 formula:      weight ~ avg_daily_snacks + environ
 algorithm:    sampling
 sample:       4000 (posterior sample size)
 priors:       see help('prior_summary')
 observations: 16
 predictors:   3

Estimates:
                   mean   sd   10%   50%   90%
(Intercept)      3.5    0.1  3.4   3.5   3.7
avg_daily_snacks 0.2    0.1  0.1   0.2   0.2
environOutdoor   0.3    0.2  0.1   0.3   0.5
sigma            0.3    0.1  0.2   0.3   0.4

Fit Diagnostics:
           mean   sd   10%   50%   90%
mean_PPD 3.9    0.1  3.8   3.9   4.1

The mean_ppd is the sample average posterior predictive distribution of the outcome variable

MCMC diagnostics
                 mcse Rhat n_eff
(Intercept)      0.0  1.0  3785
avg_daily_snacks 0.0  1.0  3047
environOutdoor   0.0  1.0  2937
sigma            0.0  1.0  2680
mean_PPD         0.0  1.0  3807
```

```
log-posterior    0.0  1.0  1463

For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective
```

```
describe_posterior(model_bcat2)
```

```
Summary of Posterior Distribution

Parameter        | Median |        95% CI |     pd |          ROPE | % in ROPE |  Rhat |
-------------------------------------------------------------------------------------------
(Intercept)      |   3.53 | [ 3.25, 3.81] |   100% | [-0.04, 0.04] |        0% | 1.000 | 378
avg_daily_snacks |   0.16 | [ 0.02, 0.30] | 98.90% | [-0.04, 0.04] |     1.55% | 1.001 | 304
environOutdoor   |   0.28 | [-0.04, 0.62] | 95.65% | [-0.04, 0.04] |     4.18% | 1.000 | 293
```

```
report::report(model_bcat2)
```

```
SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 1.6e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.16 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 0.057 seconds (Warm-up)
Chain 1:                0.043 seconds (Sampling)
Chain 1:                0.1 seconds (Total)
```

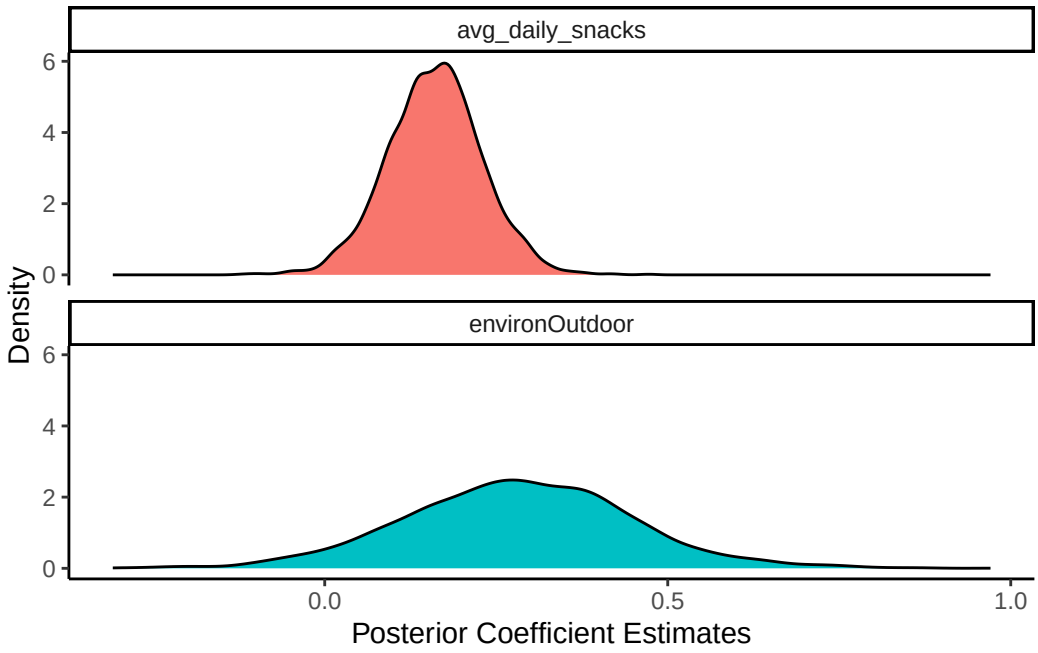
```
Chain 1:

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 1.1e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.11 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 0.062 seconds (Warm-up)
Chain 2:                0.071 seconds (Sampling)
Chain 2:                0.133 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 1.9e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.19 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
```

```
Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 0.089 seconds (Warm-up)
Chain 3:                0.052 seconds (Sampling)
Chain 3:                0.141 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'continuous' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 1e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.1 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 0.073 seconds (Warm-up)
Chain 4:                0.098 seconds (Sampling)
Chain 4:                0.171 seconds (Total)
Chain 4:
```

We fitted a Bayesian linear model (estimated using MCMC sampling with 4 chains of 2000 iterations and a warmup of 1000) to predict weight with avg_daily_snacks and environ (formula: weight ~ avg_daily_snacks + environ). Priors over parameters were all set as normal (mean = 0.00, SD = 0.80; mean = 0.00, SD = 1.87) distributions. The model's explanatory power is substantial (R2 = 0.50, 95% CI [0.17, 0.74], adj. R2 = 0.26). The model's intercept, corresponding to avg_daily_snacks = 0 and environ = Indoor, is at 3.53 (95% CI [3.25, 3.81]). Within this model:

```
  - The effect of avg daily snacks (Median = 0.16, 95% CI [0.02, 0.30]) has a
98.90% probability of being positive (> 0), 97.72% of being significant (>
0.02), and 76.35% of being large (> 0.11). The estimation successfully
converged (Rhat = 1.001) and the indices are reliable (ESS = 3047)
  - The effect of environ [Outdoor] (Median = 0.28, 95% CI [-0.04, 0.62]) has a
95.65% probability of being positive (> 0), 94.53% of being significant (>
0.02), and 85.28% of being large (> 0.11). The estimation successfully
converged (Rhat = 1.000) and the indices are reliable (ESS = 2937)

Following the Sequential Effect eXistence and sIgnificance Testing (SEXIT)
framework, we report the median of the posterior distribution and its 95% CI
(Highest Density Interval), along the probability of direction (pd), the
probability of significance and the probability of being large. The thresholds
beyond which the effect is considered as significant (i.e., non-negligible) and
large are |0.02| and |0.11| (corresponding respectively to 0.05 and 0.30 of the
outcome's SD). Convergence and stability of the Bayesian sampling has been
assessed using R-hat, which should be below 1.01 (Vehtari et al., 2019), and
Effective Sample Size (ESS), which should be greater than 1000 (Burkner, 2017).
```

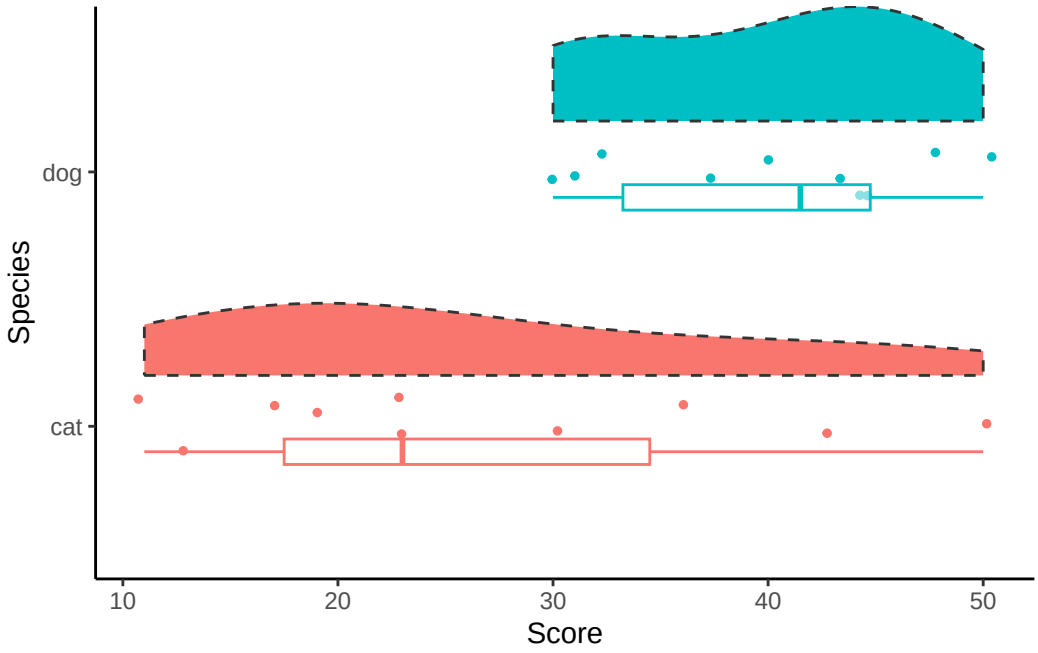
```r
posteriors2 <- get_parameters(model_bcat2)


posteriors2 |>
  pivot_longer(cols = c(avg_daily_snacks, environOutdoor),
               names_to = "Parameter",
               values_to="estimate") |>
  ggplot() +
  geom_density(aes(x = estimate, fill = Parameter)) +
  theme_classic() +
  labs(x = "Posterior Coefficient Estimates",
       y = "Density") +
  facet_wrap(facets = ~Parameter, ncol = 1) +
  theme(legend.position = "none")
```


avg_daily_snacks
environOutdoor
Density
6
4
2
0
0.0
0.5
1.0
Posterior Coefficient Estimates

Lecture: Calculating variance

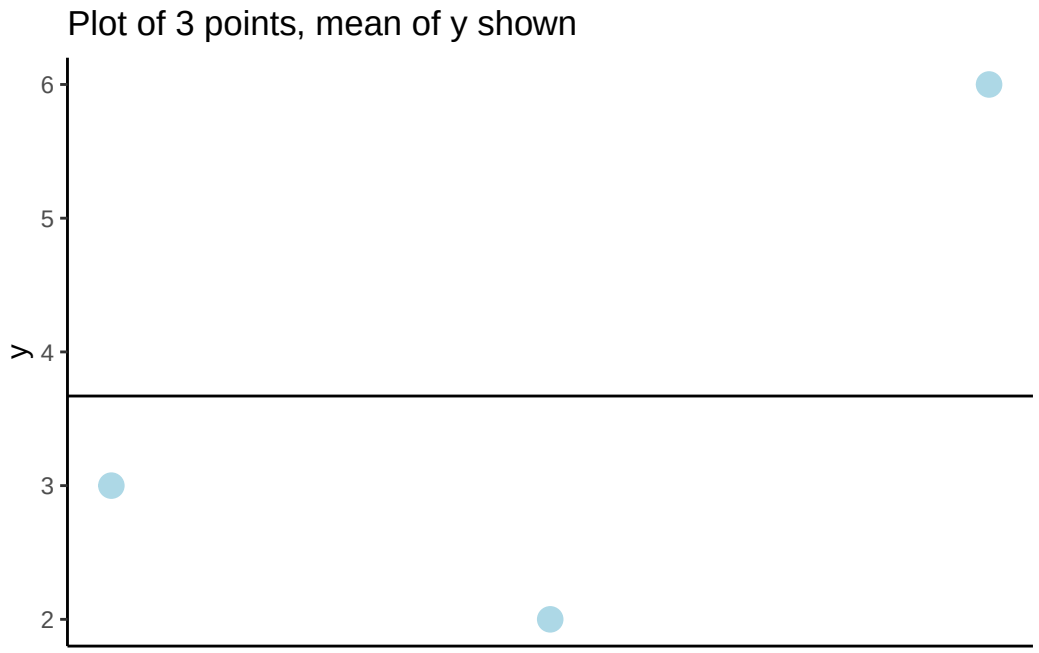
Why does it matter?

```r
vardat <- tibble(cat = c(13, 17, 30, 36, 11, 43, 23, 50, 19, 23),
                 dog = c(30, 31, 45, 43, 48, 50, 37, 32, 40, 44))



vardat |>
  pivot_longer(cols = c(cat, dog),
               names_to = "Species",
               values_to = "Score") |>
  ggplot(aes(x = Species)) +
  geom_point(aes(y = Score, colour = Species), position = position_jitter(width = .13), si
  see::geom_violinhalf(aes(y = Score, fill = Species), linetype = "dashed", position = pos
  geom_boxplot(aes(y = Score, alpha = 0.3, colour = Species), position = position_nudge(x
  theme_classic() +
  labs(x = "Species", y = "Score") +
  theme(legend.position = "none") +
  coord_flip()
```



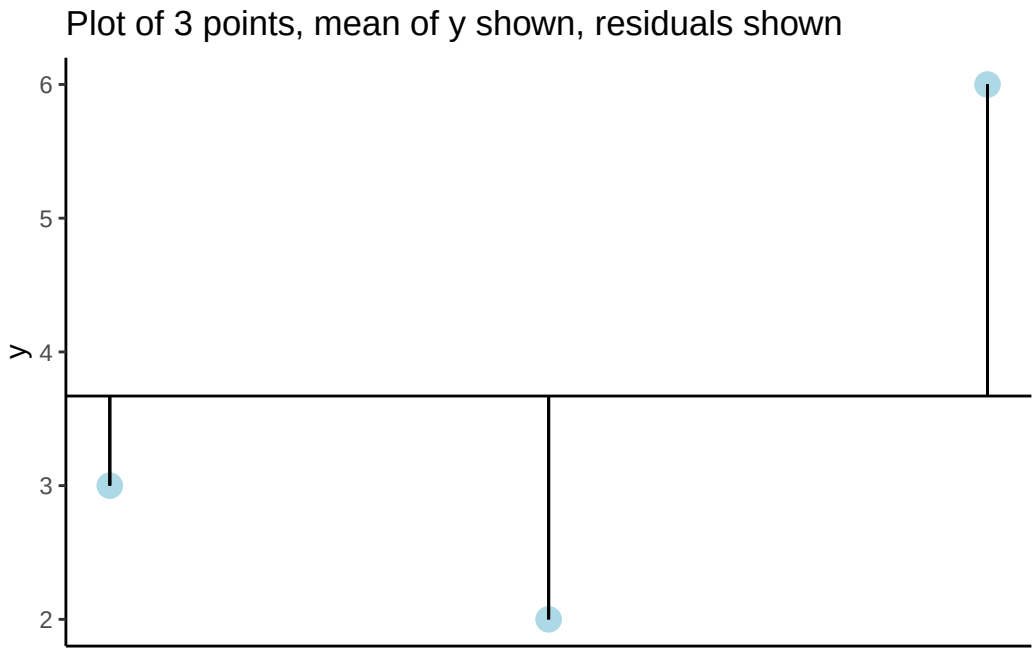
Residuals

```
resid <- tibble(x = c(1, 2, 3),
                y = c(3, 2, 6))

resid |>
  ggplot(aes(x, y)) +
  geom_point(size = 4, colour = "lightblue") +
  theme_classic() +
  geom_hline(yintercept = 3.67) +
  labs(title = "Plot of 3 points, mean of y shown") +
  theme(axis.text.x=element_blank(),
        axis.ticks.x=element_blank(),
        axis.title.x = element_blank())
```



Adding the residuals

```
resid |>
  ggplot(aes(x, y)) +
  geom_point(size = 4, colour = "lightblue") +
  theme_classic() +
  geom_hline(yintercept = 3.67) +
  geom_segment(aes(x = 1, y = 3.67, xend = 1, yend = 3)) +
  geom_segment(aes(x = 2, y = 3.67, xend = 2, yend = 2)) +
  geom_segment(aes(x = 3, y = 3.67, xend = 3, yend = 6)) +
  labs(title = "Plot of 3 points, mean of y shown, residuals shown") +
  theme(axis.text.x=element_blank(),
        axis.ticks.x=element_blank(),
        axis.title.x = element_blank())
```



Compare Variances

```
vardat |>
  summarise(var_dogs = var(dog),
            var_cat = var(cat))
```

```
# A tibble: 1 x 2
  var_dogs var_cat
     <dbl>   <dbl>
1       52    169.
```

Compare Standard Deviations

```
vardat |>
  summarise(sd_dogs = sd(dog),
            sd_cat = sd(cat))
```

```
# A tibble: 1 x 2
  sd_dogs sd_cat
    <dbl>  <dbl>
1    7.21   13.0
```

Compare Standard Errors

```
std.error <- function(x) sd(x)/sqrt(length(x))

vardat |>
  summarise(se_dogs = std.error(dog),
            se_cat = std.error(cat))
```

```
# A tibble: 1 x 2
  se_dogs se_cat
    <dbl>  <dbl>
1    2.28   4.11
```

Lecture: Meta Analyses

Calculate rs from R2

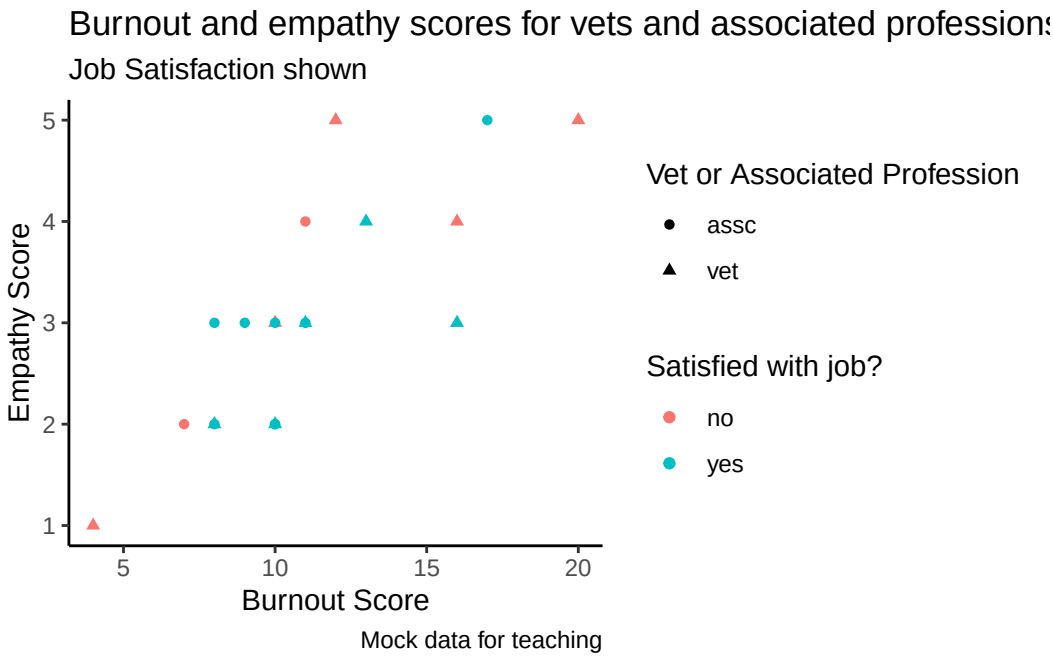
```
sqrt(0.11)
```

5 Week 4: Considerations for Collecting Data

Lecture: Effect Sizes and Covariance

Mock Data and visualisation

```
job_dat <- tibble(job = c("vet", "vet", "vet","vet", "vet", "vet", "vet", "vet", "vet", "v
                          "assc", "assc", "assc", "assc", "assc", "assc", "assc", "assc",
                  burnout = c(13, 12, 4, 16, 16, 20, 8, 10, 11, 10,
                              10, 11, 8, 7, 8, 10, 9, 11, 17, 10),
                  empathy = c(4, 5, 1, 4,3, 5, 2, 3,3,2,
                              2, 3, 3, 2, 2, 3, 3, 4, 5, 2),
                  satisfaction = c("yes", "no", "no", "no", "yes", "no", "yes", "no", "yes
                                   "yes", "yes", "yes", "no", "yes", "yes", "yes","no", "y


job_dat |>
  ggplot(aes(x = burnout, y = empathy, shape = job, colour = satisfaction)) +
  geom_point() +
  theme_classic() +
  labs(title = "Burnout and empathy scores for vets and associated professions",
       subtitle = "Job Satisfaction shown",
       caption = "Mock data for teaching",
       x = "Burnout Score",
       y = "Empathy Score") +
  scale_shape_discrete(name = "Vet or Associated Profession") +
  scale_color_discrete(name = "Satisfied with job?")
```



Calculcate Cohen's d

```
library(effsize)

cohen.d(d = job_dat$burnout, f = job_dat$job)
```

```
Cohen's d

d estimate: -0.5048995 (medium)
95 percent confidence interval:
     lower      upper
-1.4593128  0.4495138
```

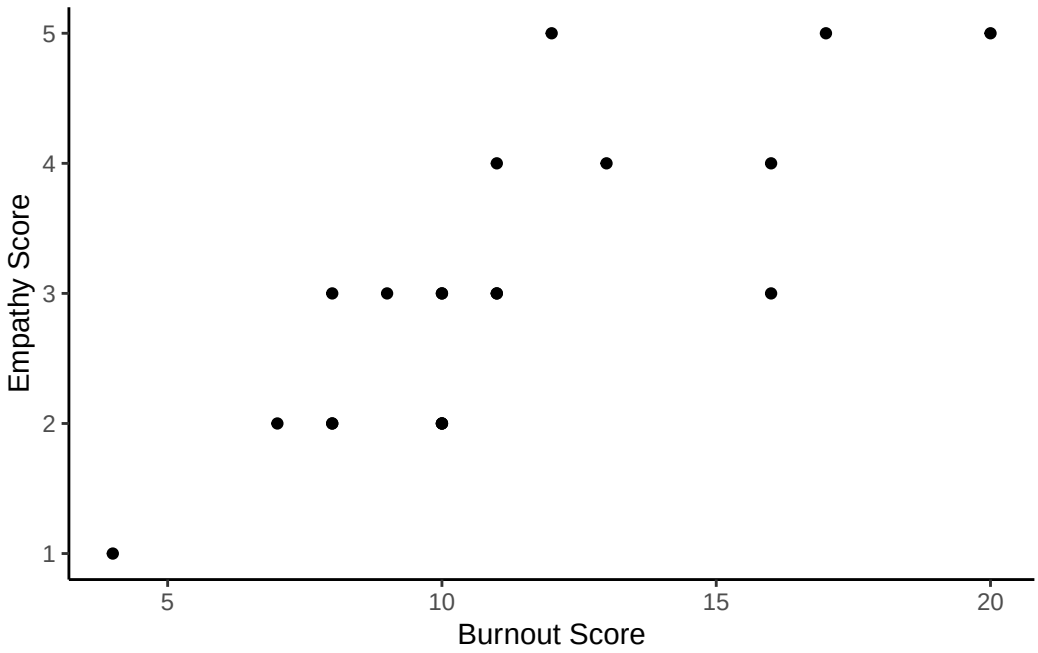
Calculcate Hedge's g

```
cohen.d(d = job_dat$burnout, f = job_dat$job, hedges.correction = TRUE)
```

```
Hedges's g

g estimate: -0.4835657 (small)
95 percent confidence interval:
     lower      upper
-1.3964834  0.4293519
```

Effects of Differences

```
job_dat |>
  ggplot(aes(x = burnout, y = empathy)) +
  geom_point() +
  theme_classic() +
  labs(x = "Burnout Score", y = "Empathy Score")
```



Correlation coefficient (r)

```
cor(job_dat$burnout, job_dat$empathy, method = "pearson")
```

```
[1] 0.7991678
```

Other Correlation Coefficients

```
cor(job_dat$burnout, job_dat$empathy, method = "spearman")
```

```
[1] 0.8202187
```

```
cor.test(job_dat$burnout, as.numeric(as.factor(job_dat$job)))
```

```

    Pearson's product-moment correlation

data:  job_dat$burnout and as.numeric(as.factor(job_dat$job))
t = 1.129, df = 18, p-value = 0.2737
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.2091671  0.6281908
sample estimates:
      cor
0.2571562
```

Cramer's V

```
library(lsr)

job_dat <- job_dat |>
  mutate(burnoutcat = case_when(burnout > 10 ~ "burnout",
                                TRUE ~ "no burnout"))
```

```
job_tbl <- xtabs(~job_dat$job +  job_dat$satisfaction + job_dat$burnoutcat)
ftable(job_tbl)
```

```
                                job_dat$burnoutcat burnout no burnout
job_dat$job job_dat$satisfaction
assc        no                                              1         1
            yes                                             2         6
vet         no                                              3         2
            yes                                             3         2
```

```
chisq.test(ftable(job_tbl))
```

```

    Pearson's Chi-squared test

data:  ftable(job_tbl)
X-squared = 2.2222, df = 3, p-value = 0.5276
```

```
cramersV(ftable(job_tbl))
```

```
[1] 0.3333333
```

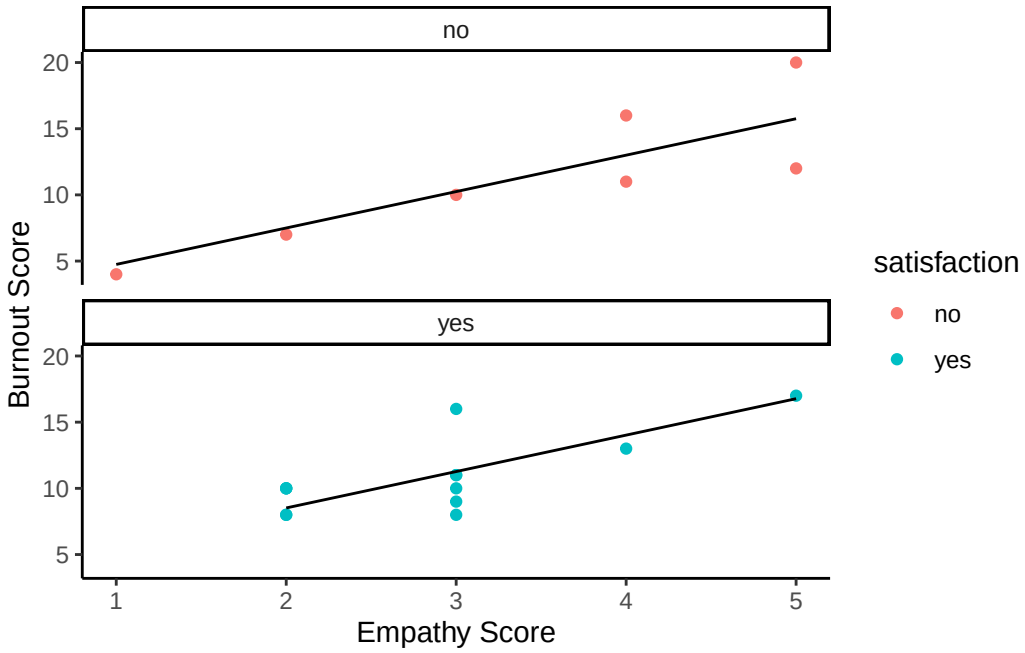
R^2adj Example

```
jobmod <- lm(burnout ~ empathy + satisfaction, data = job_dat)
summary(jobmod)
```

```

Call:
lm(formula = burnout ~ empathy + satisfaction, data = job_dat)

Residuals:
    Min      1Q  Median      3Q     Max
-3.7526 -1.0832 -0.3836  1.4822  4.7305

Coefficients:
```

```
                Estimate Std. Error t value Pr(>|t|)
(Intercept)       1.9944     1.8904   1.055    0.306
empathy           2.7516     0.4865   5.656 2.85e-05 ***
satisfactionyes   1.0202     1.1393   0.895    0.383
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.354 on 17 degrees of freedom
Multiple R-squared:  0.6549,    Adjusted R-squared:  0.6144
F-statistic: 16.13 on 2 and 17 DF,  p-value: 0.000118
```

```r
job_dat |>
  mutate(mod = predict(jobmod)) |>
  ggplot() +
  geom_point(aes(x = empathy, y = burnout, colour = satisfaction)) +
  geom_line(aes(x = empathy, y = mod)) +
  theme_classic() +
  facet_wrap(facets = ~ satisfaction, ncol = 1) +
  labs(x = "Empathy Score", y = "Burnout Score")
```

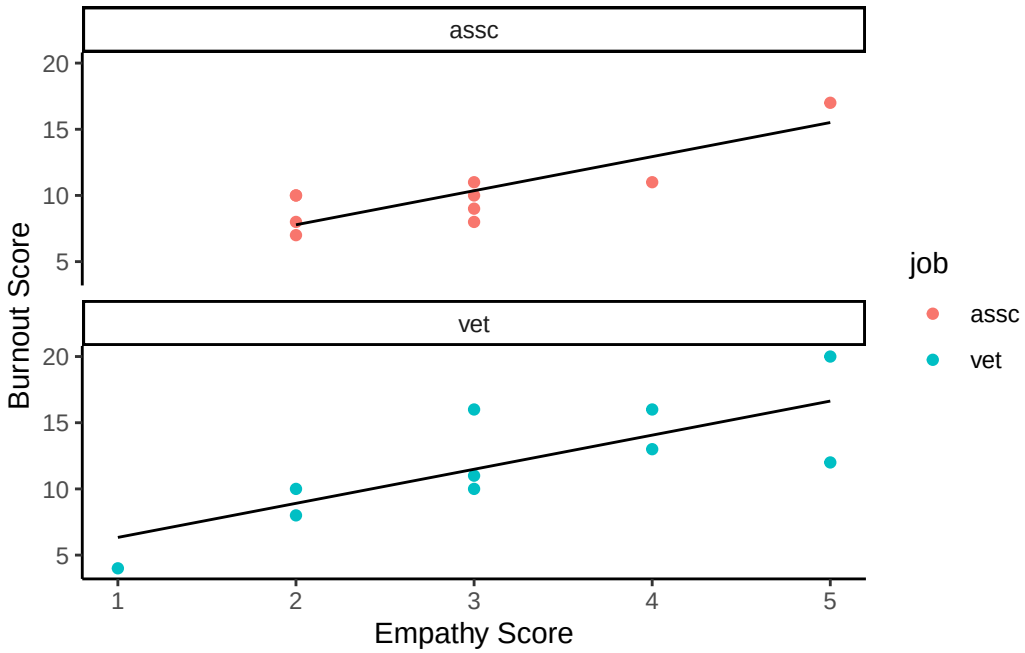


5.1 Covariance {. unnumbered}

```r
jobmod2 <- lm(burnout ~ empathy + job, data = job_dat)
summary(jobmod2)
```

```
Call:
lm(formula = burnout ~ empathy + job, data = job_dat)

Residuals:
    Min      1Q  Median      3Q     Max
-4.6359 -1.3894 -0.4212  1.6035  4.5151

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    2.631      1.552   1.695    0.108
empathy        2.575      0.471   5.469 4.16e-05 ***
jobvet         1.127      1.052   1.072    0.299
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.331 on 17 degrees of freedom
Multiple R-squared:  0.6615,    Adjusted R-squared:  0.6217
F-statistic: 16.61 on 2 and 17 DF,  p-value: 0.0001002
```

```r
job_dat |>
  mutate(mod = predict(jobmod2)) |>
  ggplot() +
  geom_point(aes(x = empathy, y = burnout, colour = job)) +
  geom_line(aes(x = empathy, y = mod)) +
  theme_classic() +
  facet_wrap(facets = ~ job, ncol = 1) +
  labs(x = "Empathy Score", y = "Burnout Score")
```

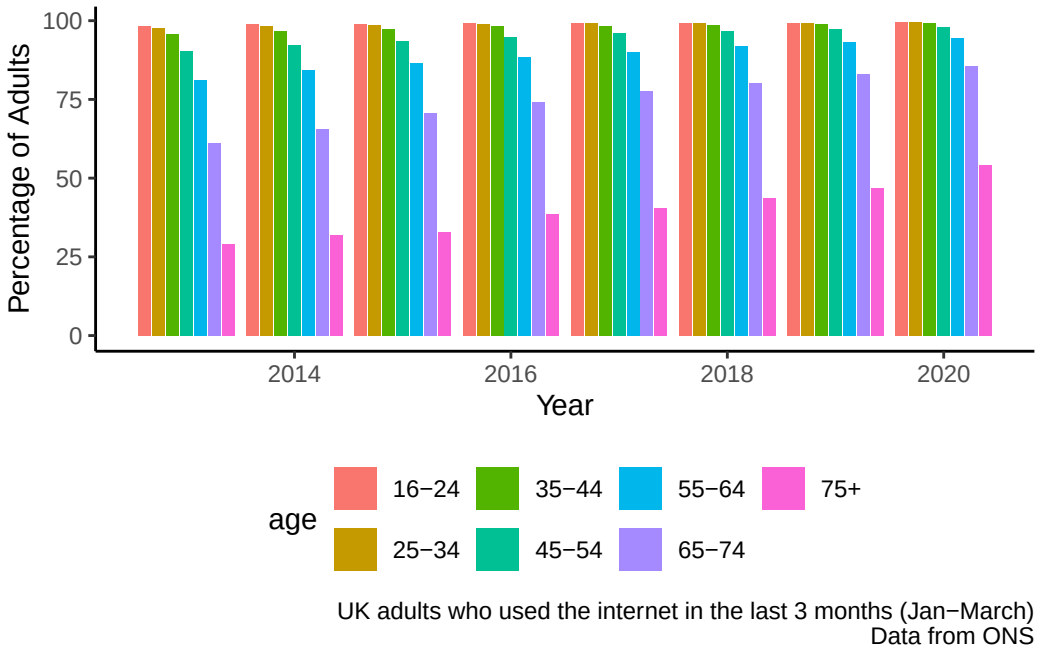

assc
vet
Burnout Score
Empathy Score
job
assc
vet
20
15
10
5
1
2
3
4
5

6 Week 5: Sources of Data

Lecture 1 Digital Media Research

6.1 Who's in digital spaces

```
library(tidyverse)

internet <- tibble(year = c(2013, 2014, 2015, 2016, 2017,   2018,   2019,   2020),
                   "16-24" = c(98.3,    98.9,   98.8,   99.2,   99.2,   99.3,   99.2,   99
                   "25-34" = c(97.7,    98.3,   98.6,   98.9,   99.1,   99.2,   99.4,   99
                   "35-44" = c(95.8,    96.7,   97.3,   98.2,   98.4,   98.6,   98.9,   99
                   "45-54" = c(90.2,    92.3,   93.6,   94.9,   96.2,   96.8,   97.5,   97
                   "55-64" = c(81.3,    84.2,   86.7,   88.3,   90.0,   91.8,   93.2,   94
                   "65-74" = c(61.1,    65.5,   70.6,   74.1,   77.5,   80.2,   83.2,   85
                  "75+" = c(29.1,   31.9,   33.0,   38.7,   40.5,   43.6,   46.8,   54.0))
  pivot_longer(cols = -year,
               names_to = "age",
               values_to = "perc")


internet |>
  ggplot(aes(x = year, y = perc, fill = age)) +
  geom_bar(stat = "identity", position = "dodge2") +
  theme_classic() +
  theme(legend.position = "bottom") +
  labs(x = "Year", y = "Percentage of Adults",
       caption = "UK adults who used the internet in the last 3 months (Jan-March)\nData f
```



UK adults who used the internet in the last 3 months (Jan−March)
Data from ONS

7 Week 6: Analysing Qualitative Data

There's no content in this week this year :)

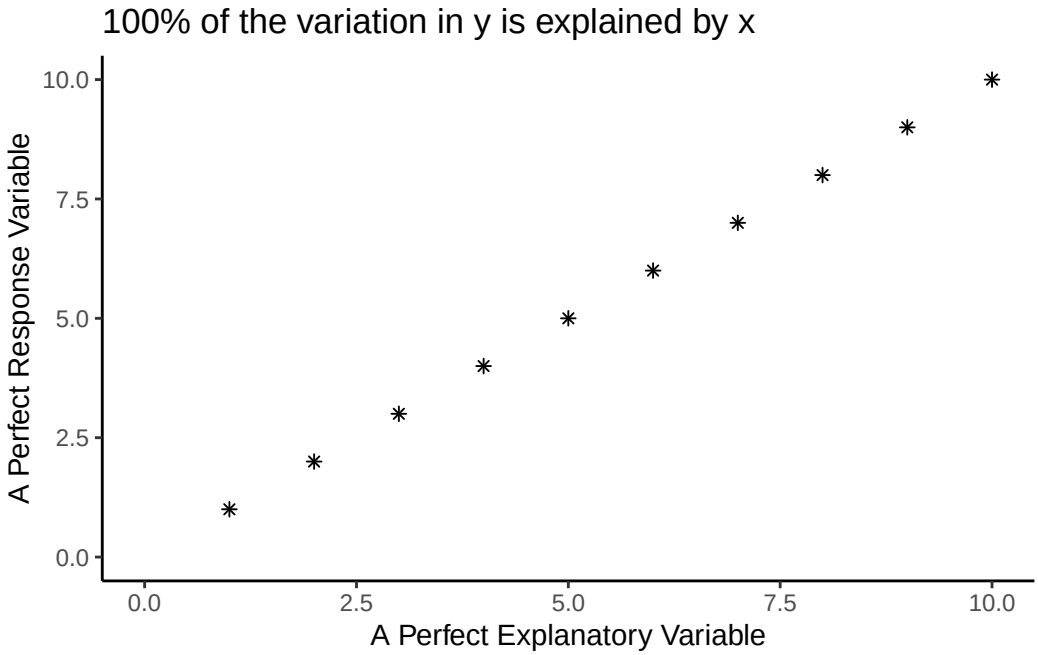
8 Week 7: Analysing Quantitative Data

Lecture 1: Partitioning Variation

```
library(tidyverse)


examples <- tibble (x.perfect = c(1,2,3,4,5,6,7,8,9,10),
                    y.perfect = c(1,2,3,4,5,6,7,8,9,10),
                    x.realistic = c(1,2,3,4,4,6,8,8,9,11),
                    y.realistic = c(2, 2, 4, 5, 5, 5, 7, 8, 9, 9))
```

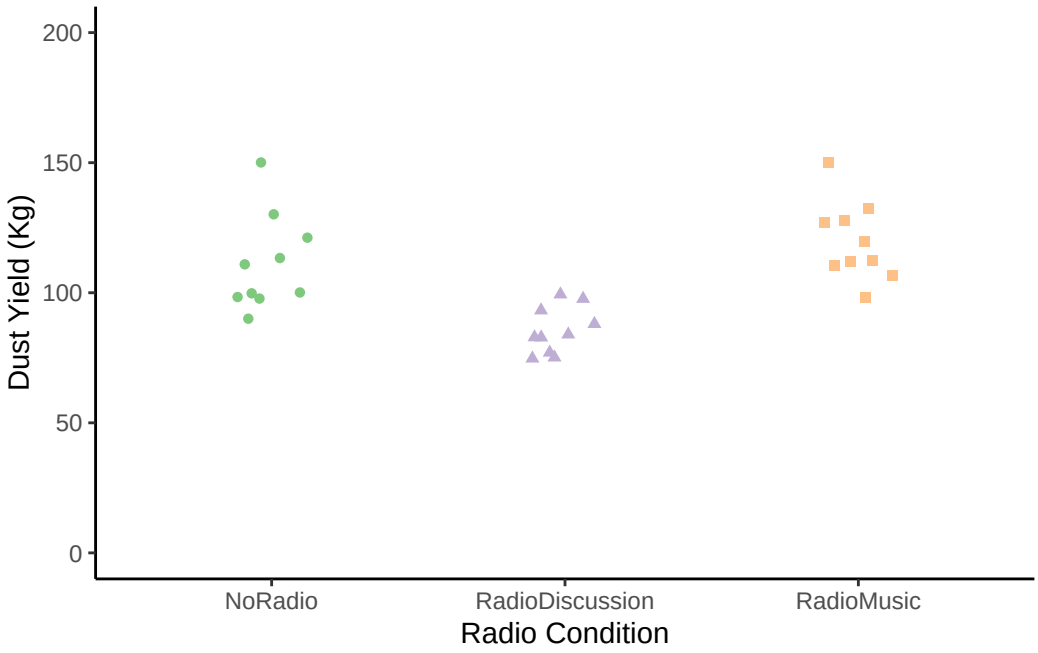
A Perfect World

```
examples |>
  ggplot (aes(x = x.perfect, y = y.perfect))+
  geom_point (shape = 8) +
  scale_y_continuous(limits =c(0,10)) +
  scale_x_continuous(limits = c(0,10)) +
  labs (title = "100% of the variation in y is explained by x",
        x = "A Perfect Explanatory Variable",
        y = "A Perfect Response Variable") +
  theme_classic()
```



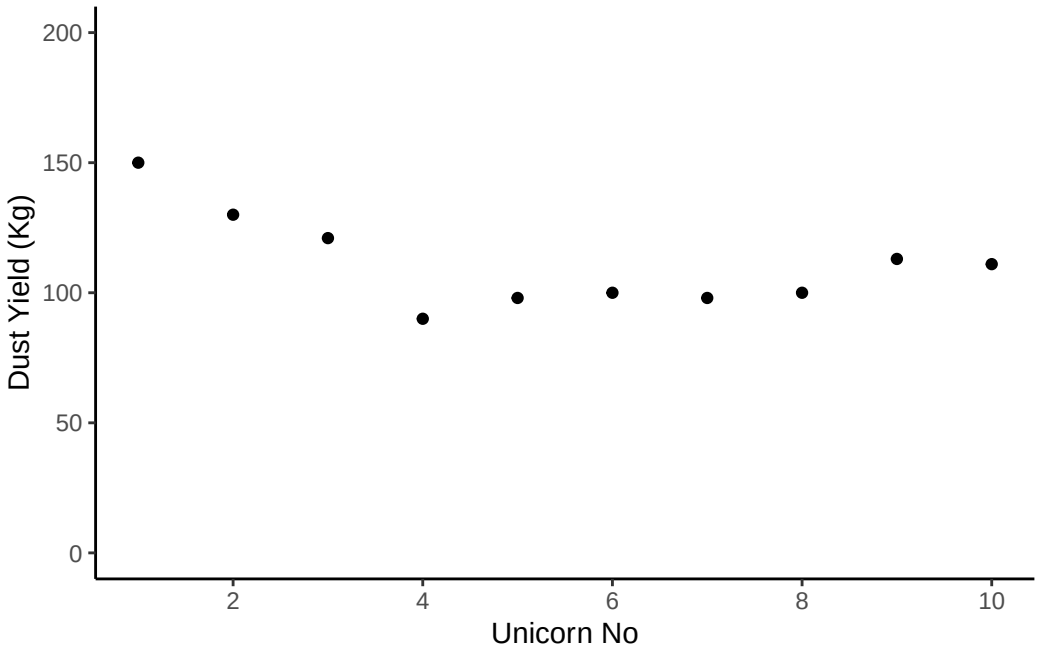
Our Unicorn Farm

```
unicorns <- tibble (NoRadio = c(150, 130, 121, 90, 98, 100, 98, 100, 113, 111),
                    RadioMusic = c(112, 127, 132, 150, 112, 128, 110, 120, 98, 107),
                    RadioDiscussion = c(75, 98, 88, 83, 83, 77, 75, 84, 93, 99)) |>
  pivot_longer(cols = c(NoRadio:RadioDiscussion),
               names_to = "Radio",
               values_to = "DustYield")


unicorns |>
  ggplot (aes (x = Radio, y = DustYield)) +
  geom_point(aes(shape = Radio, colour = Radio),  position = position_jitter(width = .13)) +
  labs (y = "Dust Yield (Kg)", x = "Radio Condition") +
  scale_color_brewer(palette = "Accent") +
  scale_y_continuous(limits = c(0, 200)) +
  theme_classic () +
  theme(legend.position = "none")
```



8.1 No Radio Group .{unnumbered}

```
unicorns |>
  filter(Radio == "NoRadio") |>
  mutate(UnicornNo = c(1,2,3,4,5,6,7,8,9,10)) |>
  # The mutate function adds a new variable just to plot this one specific chart
  # And then we pipe it directly into ggplot, so we're not changing the unicorns data
  # Remember you can check this with `View(unicorns)`, you'll see 'UnicornNo' doesn't exis
  ggplot (aes (x = UnicornNo, y = DustYield)) +
  geom_point() +
  labs (y = "Dust Yield (Kg)", x = "Unicorn No") +
  scale_y_continuous(limits = c(0, 200)) +
  scale_x_continuous(breaks = c(0, 2, 4, 6, 8, 10)) +
  theme_classic ()
```



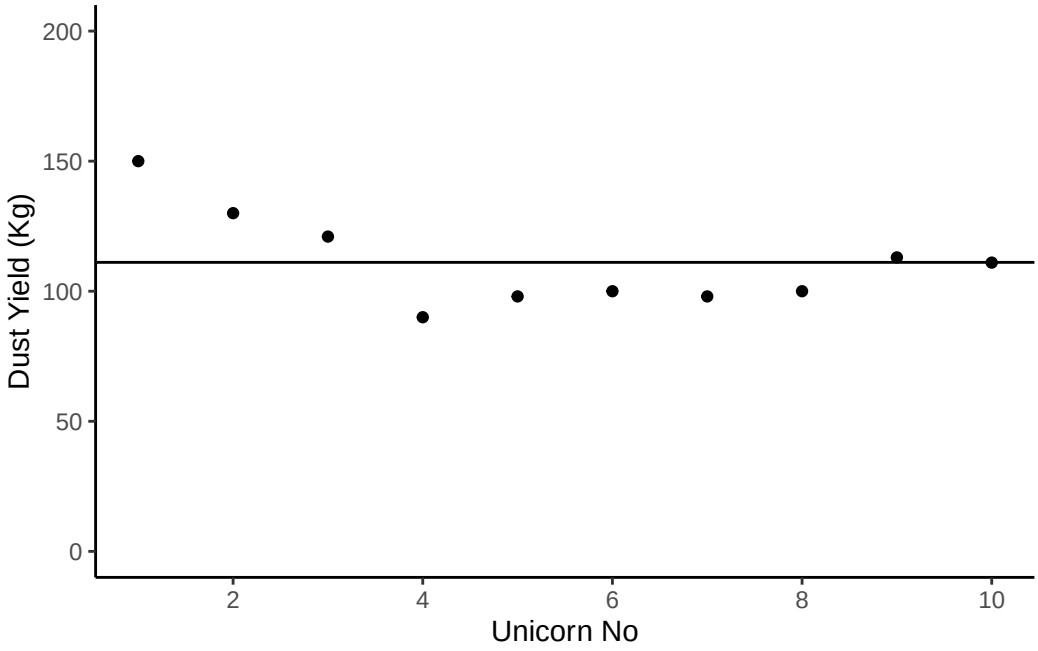
8.2 No Radio Mean .{unnumbered}

```
unicorns |>
  group_by(Radio) |>
  filter(Radio == "NoRadio") |>
  summarise(mean = mean(DustYield))
```

```
# A tibble: 1 x 2
  Radio    mean
  <chr>   <dbl>
1 NoRadio  111.
```

```
unicorns |>
  filter(Radio == "NoRadio") |>
  mutate(UnicornNo = c(1,2,3,4,5,6,7,8,9,10)) |>
  # The mutate function adds a new variable just to plot this one specific chart
  # And then we pipe it directly into ggplot, so we're not changing the unicorns data
  # Remember you can check this with `View(unicorns)`, you'll see 'UnicornNo' doesn't exis
  ggplot (aes (x = UnicornNo, y = DustYield)) +
  geom_point() +
```

```
  geom_hline(yintercept = 111.1)+
  labs (y = "Dust Yield (Kg)", x = "Unicorn No") +
  scale_y_continuous(limits = c(0, 200)) +
  scale_x_continuous(breaks = c(0, 2, 4, 6, 8, 10)) +
  theme_classic ()
```



8.3 Deviations from the mean .{unnumbered}

```
unicorns |>
  filter(Radio == "NoRadio") |>
  mutate(Diff = DustYield-111.1)
```

```
# A tibble: 10 x 3
   Radio   DustYield    Diff
   <chr>       <dbl>   <dbl>
 1 NoRadio       150  38.9
 2 NoRadio       130  18.9
 3 NoRadio       121   9.9
 4 NoRadio        90 -21.1
```

```
 5 NoRadio        98 -13.1
 6 NoRadio       100 -11.1
 7 NoRadio        98 -13.1
 8 NoRadio       100 -11.1
 9 NoRadio       113   1.90
10 NoRadio       111  -0.100
```

```
unicorns |>
  filter(Radio == "NoRadio") |>
  mutate(Diff = DustYield-111.1) |>
  summarise(`Sum of Differences` = sum(Diff))
```

```
# A tibble: 1 x 1
  `Sum of Differences`
                 <dbl>
1             5.68e-14
```

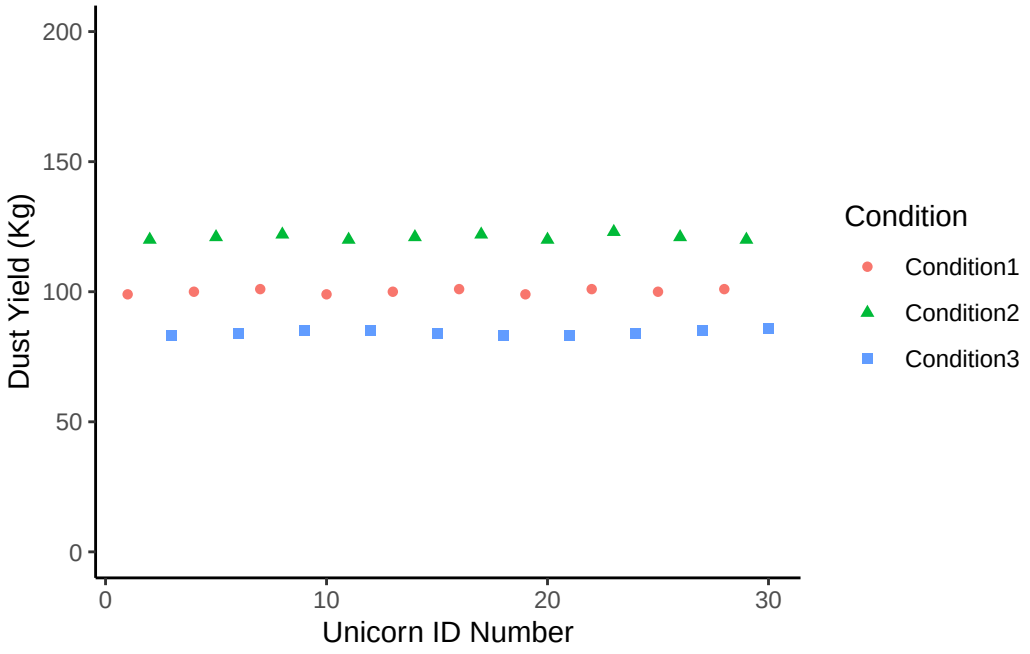
8.4 Variance .{unnumbered}

```
#: eval: true


unicorns |>
  group_by(Radio) |>
  summarise (Variance = var(DustYield))
```

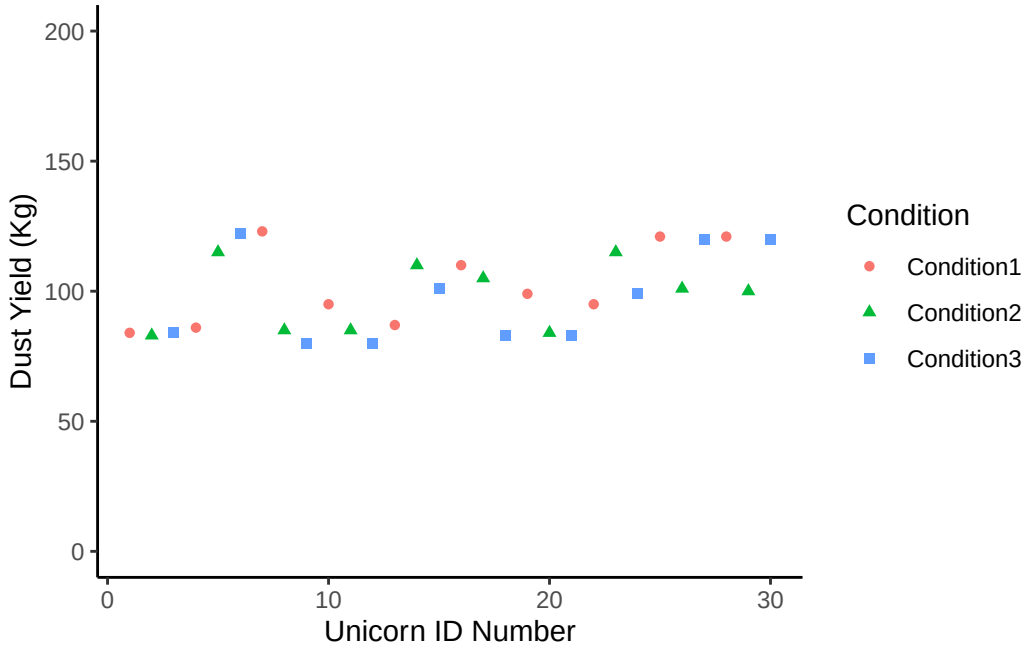
8.5 Imaginary Scenarions .{unnumbered}

```
Sce1 <- tibble (Condition1 = c(99,100,101,99,100,101,99,101,100,101),
                Condition2 = c(120,121,122,120,121,122,120,123,121,120),
                Condition3 = c(83,84,85,85,84,83,83,84,85,86)) |>
  pivot_longer (cols = c(Condition1:Condition3), names_to = "Condition", values_to = "Dust
  mutate (Count = c(1:30))

Sce2 <-tibble (Condition1 = c(84,86,123,95,87,110,99,95,121,121),
               Condition2 = c(83,115,85,85, 110,105,84,115,101,100),
               Condition3 = c(84,122,80,80,101,83,83,99, 120,120)) |>
```

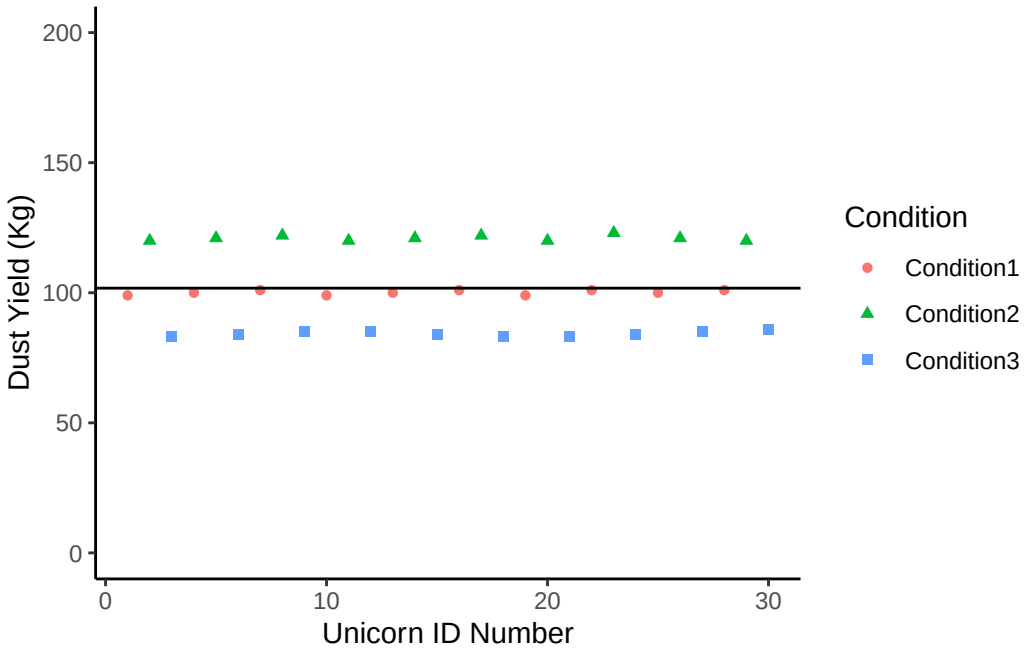
```
  pivot_longer (cols = c(Condition1:Condition3), names_to = "Condition", values_to = "Dust
  mutate (Count = c(1:30))



ImaginaryScenario1 <- Sce1 |>
  ggplot (aes (x = Count, y = DustYield)) +
  geom_point(aes(shape = Condition, colour = Condition)) +
  labs (y = "Dust Yield (Kg)", x = "Unicorn ID Number") +
  scale_y_continuous(limits = c(0, 200)) +
  theme_classic ()

# I have also made this chart an object because we're going to update it
# It's quicker to do this as an object
# You can compare how we update this chart with how we update the ones above.

ImaginaryScenario1
```



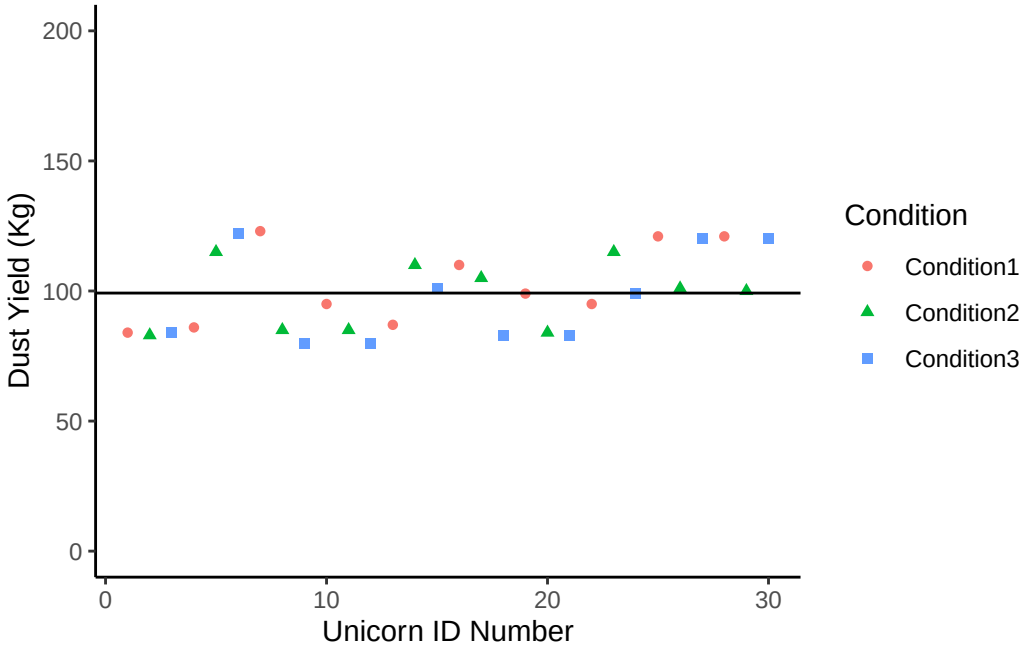
```
experiment <- Sce2 |>
  ggplot (aes (x = Count, y = DustYield)) +
  geom_point(aes(shape = Condition, colour = Condition)) +
  labs (y = "Dust Yield (Kg)", x = "Unicorn ID Number") +
  scale_y_continuous(limits = c(0, 200)) +
  theme_classic ()

experiment
```



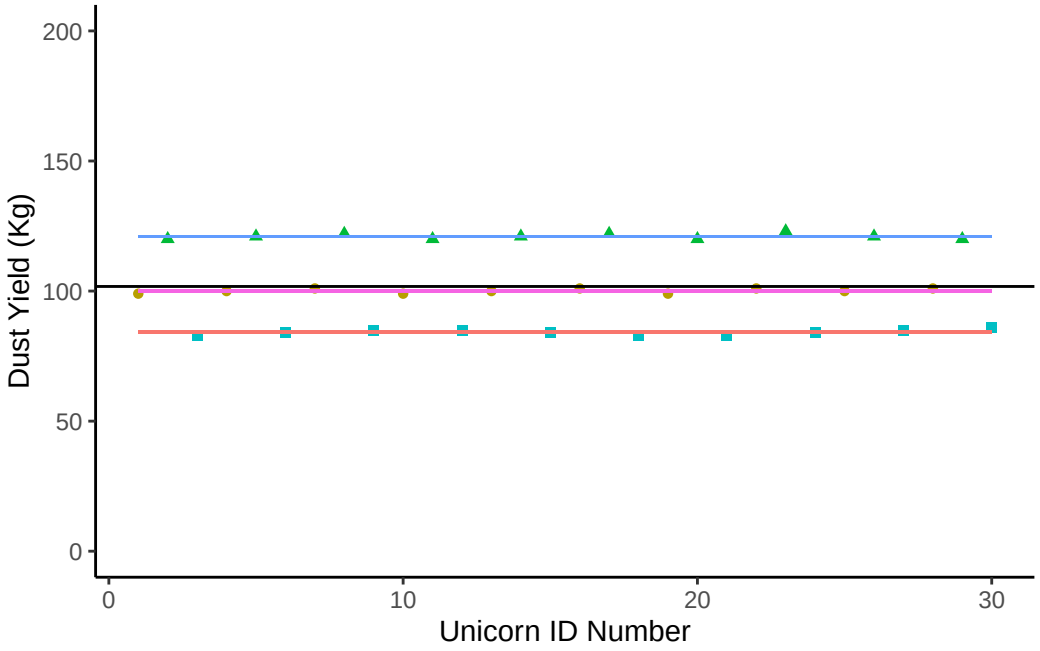

```
ImaginaryScenario1 +
  geom_hline(yintercept = mean(Sce1$DustYield))
```



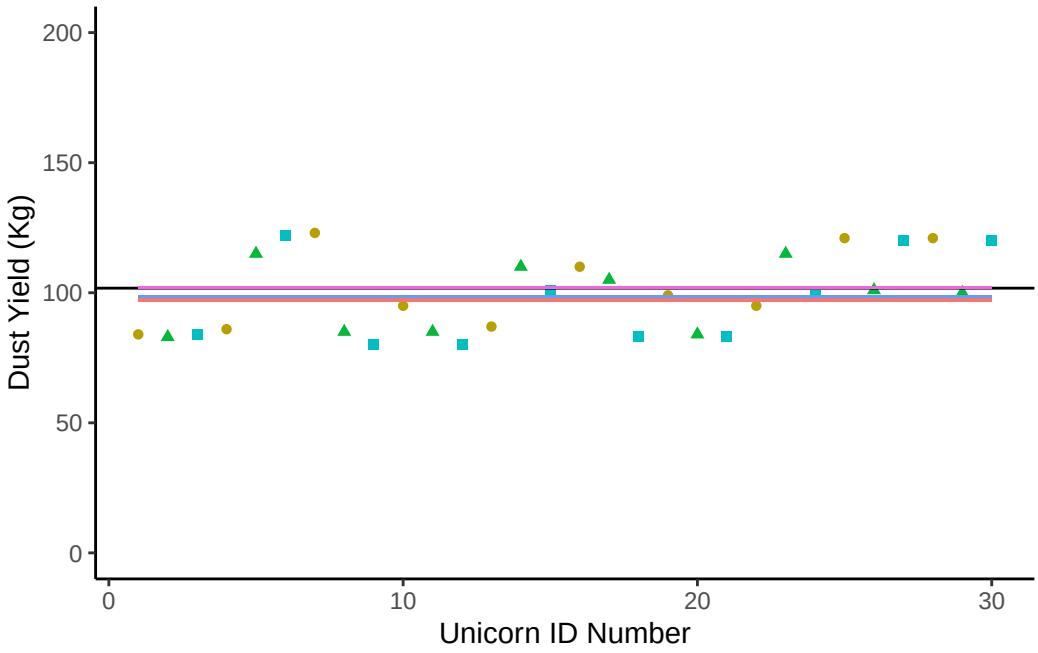

```
experiment +
  geom_hline(yintercept = mean(Sce2$DustYield))
```



```
ImaginaryScenario1 +
  geom_hline(yintercept = mean(Sce1$DustYield)) +
  geom_segment(aes(x =1, y = 100.1, xend =30, yend = 100.1, color = "red")) +
  geom_segment(aes(x = 1, y = 121.0, xend = 30, yend = 121.0, color = "green")) +
  geom_segment (aes(x = 1, y = 84.2, xend = 30, yend = 84.2, color = "blue")) +
  theme (legend.position = "none")
```




```
experiment +
  geom_hline(yintercept = mean(Sce1$DustYield)) +
  geom_segment(aes(x =1, y = 102, xend =30, yend = 102, color = "red")) +
  geom_segment(aes(x = 1, y = 98.3, xend = 30, yend = 98.3, color = "green")) +
  geom_segment (aes(x = 1, y = 97.2, xend = 30, yend = 97.2, color = "blue")) +
  theme (legend.position = "none")
```




```
Sce2 |>
  group_by(Condition) |>
  summarise(mean = mean (DustYield))
```

```
# A tibble: 3 x 2
  Condition   mean
  <chr>      <dbl>
1 Condition1 102.
2 Condition2  98.3
3 Condition3  97.2
```

8.6 MFY .{unnumbered}

```
MFY <- unicorns |>
  mutate ("Y" = DustYield) |>
  mutate ("M" = mean(Y)) |>
  # If we now ask R to group the data, it will calculate the mean per group:
  group_by(Radio) |>
  mutate ("F" = mean(Y)) |>
```

```
  # Remember to ungroup after!
  ungroup()



MFY <- MFY |>
  mutate (MY = (Y-M),
          MF = (F-M),
          FY = (Y - F))


MFY <- MFY |>
  mutate (MY2 = (MY*MY),
          MF2 = (MF*MF),
          FY2 = (FY*FY))


MFY |>
  summarise(SumSquareMY = sum(MY2),
            SumSquareMF = sum(MF2),
            SumSquareFY = sum(FY2))
```

```
# A tibble: 1 x 3
  SumSquareMY SumSquareMF SumSquareFY
        <dbl>       <dbl>       <dbl>
1      12053.       6301.       5752.
```

```
MFY |>
  summarise(SumSquareMY = sum(MY2),
            SumSquareMF = sum(MF2),
            SumSquareFY = sum(FY2),
            MeanSquareMY = sum(MY2)/29,
            MeanSquareMF = sum(MF2)/2,
            MeanSquareFY = sum(FY2)/27)
```

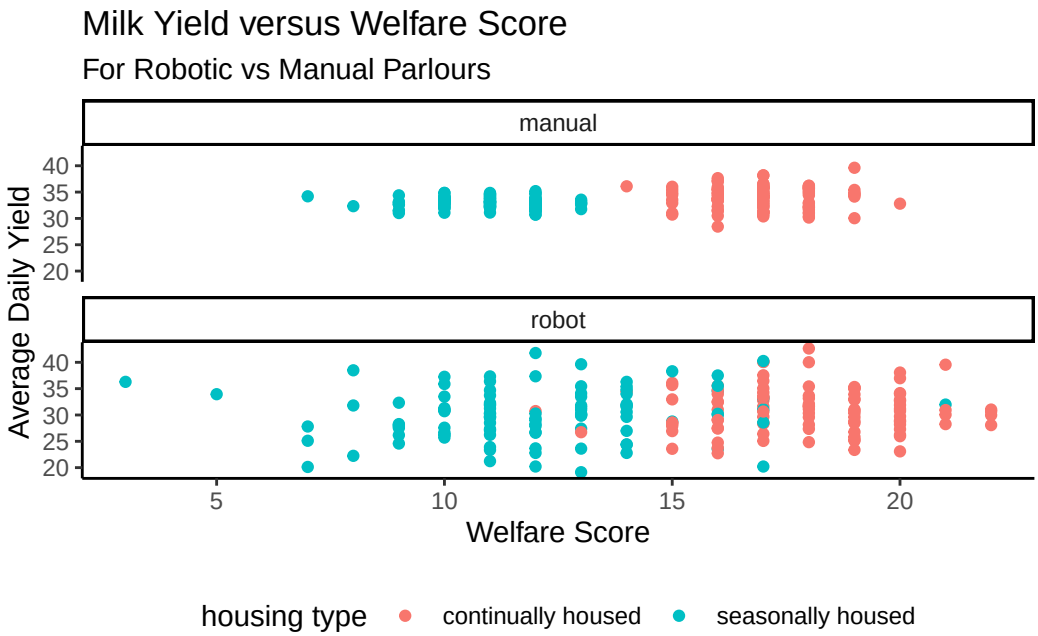
```
# A tibble: 1 x 6
  SumSquareMY SumSquareMF SumSquareFY MeanSquareMY MeanSquareMF MeanSquareFY
        <dbl>       <dbl>       <dbl>        <dbl>        <dbl>        <dbl>
1      12053.       6301.       5752.         416.        3151.         213.
```

```r
ANOVA <- aov(DustYield ~ Radio, data = unicorns)
summary(ANOVA)
```

```
            Df Sum Sq Mean Sq F value  Pr(>F)
Radio        2   6301    3151   14.79 4.6e-05 ***
Residuals   27   5752     213
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Lecture 2: Choosing a Statistical Test

```
library(tidyverse)

cows <- readxl::read_excel("cows.xlsx") |>
  mutate(parlour = as.factor(parlour),
         `housing type` = as.factor(`housing type`))


cows |>
  ggplot(aes(x = `Welfare Score`, y= `Average Daily Yield`, colour = `housing type`)) +
  geom_point() +
  theme_classic() +
  labs(x  = "Welfare Score", y = "Average Daily Yield",
       title = "Milk Yield versus Welfare Score",
       subtitle = "For Robotic vs Manual Parlours") +
  facet_wrap(facets = ~parlour, ncol = 1) +
  theme(legend.position = "bottom")
```



Numerical Response

```
cows |>
  summarise(mean_yield = mean(`Average Daily Yield`),
            sd_yield = sd(`Average Daily Yield`))
```

```
# A tibble: 1 x 2
  mean_yield sd_yield
       <dbl>    <dbl>
1         32     3.73
```

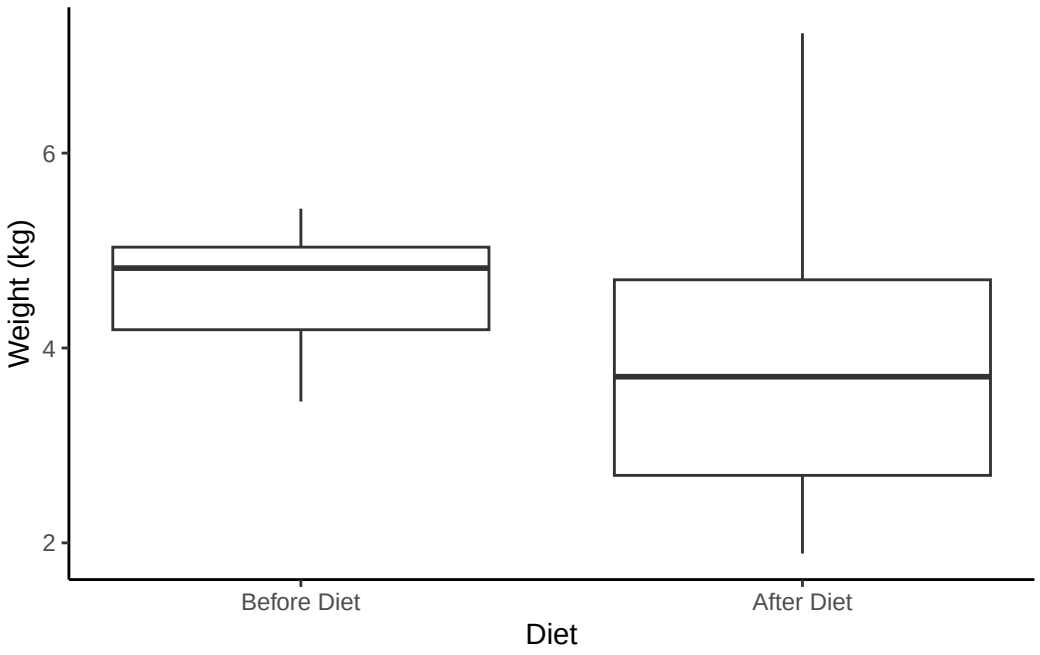
```
t.test(cows$`Average Daily Yield`, mu = 28)
```

```
    One Sample t-test

data:  cows$`Average Daily Yield`
t = 21.448, df = 399, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 28
```

```
95 percent confidence interval:
 31.63336 32.36664
sample estimates:
mean of x
       32
```

Numerical Response Categorical Explanatory

```r
diet <- tibble (before = c(5.04, 4.63, 4.04, 5.10, 5.43, 4.83, 3.45, 3.49, 5.02, 4.81),
                after = c( 4.78, 2.49, 4.46, 2.03, 5.13, 7.23, 3.50, 1.89, 3.30, 3.91))


diet |>
  ggplot(aes(y = before)) +
  geom_boxplot() +
  geom_boxplot(aes(y = after, x =1)) +
  theme_classic() +
  scale_x_continuous(labels =c("Before Diet", "After Diet"), breaks = c(0,1)) +
  labs(x = "Diet", y = "Weight (kg)")
```



```r
diet |>
  summarise(before_mean = mean(before),
            after_mean = mean(after),
            before_sd = sd(before),
            after_sd = sd(after))
```

```
# A tibble: 1 x 4
  before_mean after_mean before_sd after_sd
        <dbl>      <dbl>     <dbl>    <dbl>
1        4.58       3.87     0.689     1.62
```

```r
t.test(diet$before, diet$after, paired = TRUE, alternative = "two.sided")
```

```
    Paired t-test

data:  diet$before and diet$after
t = 1.4615, df = 9, p-value = 0.1779
alternative hypothesis: true mean difference is not equal to 0
95 percent confidence interval:
 -0.3900511  1.8140511
sample estimates:
mean difference
          0.712
```

2-Way ANOVA

```r
model1 <-  aov(`Average Daily Yield` ~ parlour +  `housing type`, data = cows)
summary(model1)
```

```
                Df Sum Sq Mean Sq F value  Pr(>F)
parlour          1    900   900.0  78.511 < 2e-16 ***
`housing type`   1    100   100.0   8.723 0.00333 **
Residuals      397   4551    11.5
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Linear model

```r
model2 <- lm(`Average Daily Yield` ~ parlour +  `housing type`, data = cows)
summary(model2)
```

```
Call:
lm(formula = `Average Daily Yield` ~ parlour + `housing type`,
    data = cows)

Residuals:
     Min       1Q   Median       3Q      Max
-10.8651  -1.7814   0.0787   1.6805  11.7654

Coefficients:
                                Estimate Std. Error t value Pr(>|t|)
(Intercept)                      34.0000     0.2932 115.956  < 2e-16 ***
parlourrobot                     -3.0000     0.3386  -8.861  < 2e-16 ***
`housing type`seasonally housed  -1.0000     0.3386  -2.954  0.00333 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.386 on 397 degrees of freedom
Multiple R-squared:  0.1801,    Adjusted R-squared:  0.176
F-statistic: 43.62 on 2 and 397 DF,  p-value: < 2.2e-16
```

Numerical Response Numerical Explanatory

Correlation

```r
cor.test(cows$`Average Daily Yield`, cows$`Welfare Score`)
```

```
    Pearson's product-moment correlation

data:  cows$`Average Daily Yield` and cows$`Welfare Score`
t = 1.7643, df = 398, p-value = 0.07845
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.01004724  0.18454894
sample estimates:
```

```
       cor
0.08809126
```

Linear Model

Genearlised Linear Model

```
#| eval: true


model4 <- glm(`Average Daily Yield` ~ `Welfare Score`
              + `housing type` + parlour, data = cows)
summary(model4)
```

Categorical Response

Proportion Test

```
prop.test(x = 3, n = 20, p = 0.18, alternative = "two.sided")
```

```
    1-sample proportions test with continuity correction

data:  3 out of 20, null probability 0.18
X-squared = 0.0033875, df = 1, p-value = 0.9536
alternative hypothesis: true p is not equal to 0.18
95 percent confidence interval:
 0.03956627 0.38862512
sample estimates:
   p
0.15
```

Categorical Response Categorical Explanatory

McNemar's Test

```
work <- matrix(c(2, 5, 38, 35),
               ncol=2,
               byrow=TRUE,
```

```
                dimnames = list(c("Dysplasia", "No Dysplasia"),
                                c("Before Work Season", "After Work Season")))


mcnemar.test(work)
```

```
    McNemar's Chi-squared test with continuity correction

data:  work
McNemar's chi-squared = 23.814, df = 1, p-value = 1.061e-06
```

Fisher's Exact Test

```
fisher.test(x = c(4,16), y = c(2,18))
```

```
    Fisher's Exact Test for Count Data

data:  c(4, 16) and c(2, 18)
p-value = 1
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
 0.02564066        Inf
sample estimates:
odds ratio
       Inf
```

```
# Calculate Odds Ratio
(4+18) / (16 + 2)
```

```
[1] 1.222222
```

Chi2 Test

```
gsd <- matrix(c(16, 12, 84, 86),
              ncol=2,
              byrow=TRUE,
              dimnames = list(c("Dysplasia", "No Dysplasia"),
```

```
                                c("Inbred", "Less Inbred")))


chisq.test(gsd)
```

```
    Pearson's Chi-squared test with Yates' continuity correction

data:  gsd
X-squared = 0.30714, df = 1, p-value = 0.5794
```

```
library(vcd)
assocstats(gsd)
```

```
                     X^2 df P(> X^2)
Likelihood Ratio 0.57672  1  0.44760
Pearson          0.57481  1  0.44835

Phi-Coefficient   : 0.054
Contingency Coeff.: 0.054
Cramer's V        : 0.054
```

Categorical Response Numerical Explanatory

Logistic Regression

```
dysp <- tibble(dysplasia = c(1, 1, 1, 1, 1, 1, 1,
                             0, 0, 0, 0, 0, 0, 0,
                             0, 0, 0, 0, 0, 0, 0),
               inflammation =c(0.91, 0.79, 1.40, 0.71, 1.01, 0.77, 0.85,
                               0.42, 1.02, 0.31, 0.05, 1.17, 0.04, 0.36,
                               0.12, 0.02, 0.05, 0.42, 0.92, 0.72,  1.05))

logit <- glm(dysplasia ~ inflammation, data = dysp, family = "binomial")
summary(logit)
```

```
Call:
```

```
glm(formula = dysplasia ~ inflammation, family = "binomial",
    data = dysp)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.5715  -0.5727  -0.3094   1.0397   1.4914

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)    -3.190      1.484  -2.150   0.0316 *
inflammation    3.488      1.740   2.004   0.0450 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 26.734  on 20  degrees of freedom
Residual deviance: 20.534  on 19  degrees of freedom
AIC: 24.534

Number of Fisher Scoring iterations: 5
```

```
library(easystats)
report(logit)
```

```
We fitted a logistic model (estimated using ML) to predict dysplasia with
inflammation (formula: dysplasia ~ inflammation). The model's explanatory power
is moderate (Tjur's R2 = 0.24). The model's intercept, corresponding to
inflammation = 0, is at -3.19 (95% CI [-7.09, -0.88], p = 0.032). Within this
model:

  - The effect of inflammation is statistically significant and positive (beta =
3.49, 95% CI [0.65, 7.91], p = 0.045; Std. beta = 1.47, 95% CI [0.27, 3.33])

Standardized parameters were obtained by fitting the model on a standardized
version of the dataset. 95% Confidence Intervals (CIs) and p-values were
computed using a Wald z-distribution approximation.
```

```
parameters(logit)
```

```
Parameter    | Log-Odds |   SE |         95% CI |     z |     p
-------------------------------------------------------------------
(Intercept)  |    -3.19 | 1.48 | [-7.09, -0.88] | -2.15 | 0.032
inflammation |     3.49 | 1.74 | [ 0.65,  7.91] |  2.00 | 0.045
```

```
exp(cbind(OddsRatio = coef(logit), confint(logit)))
```

```
                OddsRatio        2.5 %       97.5 %
(Intercept)   0.04118154 0.0008355444    0.4146414
inflammation 32.71638289 1.9151146706 2735.6121035
```

Parametric vs Non Parametric

Power Calculations

Assumptions

Residuals

```
residuals <- tibble(resids = resid(model4))

residuals |>
  ggplot(aes(x = resids)) +
  geom_density()+
  theme_classic()


parameters::describe_distribution(residuals)
```

9 Weeks 8 & 9: Analytical Softwares

This week has no content yet, please check back later!

10 Week 10: Project Proposals

This week has no content yet, please check back later!

11 References

References

The cover image duck comes from Pixabay, as a Creative Commons 0 image by Clker-Free-Vector-Images-3736

Brilleman, SL, MJ Crowther, M Moreno-Betancur, J Buros Novik, and R Wolfe. 2018. "Joint Longitudinal and Time-to-Event Models via Stan." https://github.com/stan-dev/stancon_talks/.

Fellows, Ian. 2018. *Wordcloud: Word Clouds*. https://CRAN.R-project.org/package=wordcloud.

Lüdecke, Daniel, Mattan S. Ben-Shachar, Indrajeet Patil, Brenton M. Wiernik, and Dominique Makowski. 2022. "Easystats: Framework for Easy Statistical Modeling, Visualization, and Reporting." *CRAN*. https://easystats.github.io/easystats/.

Patil, Indrajeet. 2021. "Visualizations with statistical details: The 'ggstatsplot' approach." *Journal of Open Source Software* 6 (61): 3167. https://doi.org/10.21105/joss.03167.

Stan Development Team. 2023. "RStan: The R Interface to Stan." https://mc-stan.org/.

Torchiano, Marco. 2020. *Effsize: Efficient Effect Size Computation*. https://doi.org/10.5281/zenodo.1480624.

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, et al. 2019. "Welcome to the tidyverse." *Journal of Open Source Software* 4 (43): 1686. https://doi.org/10.21105/joss.01686.

Zeileis, Achim, David Meyer, and Kurt Hornik. 2007. "Residual-Based Shadings for Visualizing (Conditional) Independence." *Journal of Computational and Graphical Statistics* 16 (3): 507–25. https://doi.org/10.1198/106186007X237856.