

# **Research Methods and Data Analysis (IAWEL)**

Jill R D MacKay

2024-02-27

# Table of contents

<b>1 Research Methods and Data Analysis (IAWEL)</b>	<b>5</b>
<b>Preface</b>	<b>6</b>
Packages in this textbook . . . . .	6
Licensing . . . . .	6
<b>2 Week 1: The Philosophy of Science</b>	<b>8</b>
<b>Lecture 3: The Replication Crisis</b>	<b>9</b>
<b>Lecture 5: Introduction to Research Methods</b>	<b>12</b>
2.1 Create data and plot . . . . .	12
2.2 Run an ANOVA on Plant data . . . . .	13
2.3 Read and Run Crude Chicken Correlations . . . . .	13
<b>3 Week 2: The Use and Abuse of Data</b>	<b>14</b>
<b>Lecture 2: Data Visualisation</b>	<b>15</b>
Height vs Weight by Sex . . . . .	15
Histogram of male height . . . . .	16
Density plot of male height . . . . .	17
Boxplot of height . . . . .	18
Mean height (bar chart) . . . . .	19
Mosaic Plot . . . . .	20
Pie Charts are Just Bad Bar Charts . . . . .	21
Wordclouds . . . . .	23
Raincloud Plots . . . . .	24
Bubble plots . . . . .	25
Correlation plots . . . . .	26
<b>Lecture 3: The Mean as a Basic Model</b>	<b>28</b>
Data and custom function for this lecture . . . . .	28
Finding central tendency . . . . .	28
The Mean and Outliers . . . . .	29
Mean UK Salary . . . . .	31
The Mode . . . . .	32

Multiple Modes . . . . .	33
The Median . . . . .	34
Median UK Salary . . . . .	35
<b>4 Week 3: Introduction to Analyses</b>	<b>37</b>
<b>Lecture 2: Introduction to statistics</b>	<b>38</b>
Set up your environment and packages . . . . .	38
Summarise example data . . . . .	38
Visualise example data . . . . .	39
A Linear Model . . . . .	39
A Bayesian Model . . . . .	42
A Linear model with a factor . . . . .	50
Bayesian Framework . . . . .	53
<b>Lecture: Calculating variance</b>	<b>62</b>
Why does it matter? . . . . .	62
Residuals . . . . .	63
Adding the residuals . . . . .	64
Compare Variances . . . . .	65
Compare Standard Deviations . . . . .	65
Compare Standard Errors . . . . .	66
<b>Lecture: Meta Analyses</b>	<b>67</b>
<b>5 Week 4: Considerations for Collecting Data</b>	<b>68</b>
<b>Lecture: Effect Sizes and Covariance</b>	<b>69</b>
Mock Data and visualisation . . . . .	69
Calculate Cohen's d . . . . .	70
Calculate Hedge's g . . . . .	70
Effects of Differences . . . . .	71
Correlation coefficient (r) . . . . .	72
Other Correlation Coefficients . . . . .	72
Cramer's V . . . . .	72
<b>6 Week 5: Sources of Data</b>	<b>74</b>
<b>7 Week 6: Analysing Qualitative Data</b>	<b>75</b>
<b>8 Week 7: Analysing Quantitative Data</b>	<b>76</b>
<b>9 Weeks 8 &amp; 9: Analytical Softwares</b>	<b>77</b>
<b>10 Week 10: Project Proposals</b>	<b>78</b>

<b>11 References</b>	<b>79</b>
<b>References</b>	<b>80</b>

# **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

This book is licensed under the Unlicense.

This is free and unencumbered software released into the public domain.

Anyone is free to copy, modify, publish, use, compile, sell, or distribute this software, either in source code form or as a compiled binary, for any purpose, commercial or non-commercial, and by any means.

In jurisdictions that recognize copyright laws, the author or authors of this software dedicate any and all copyright interest in the software to the public domain. We make this dedication

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**



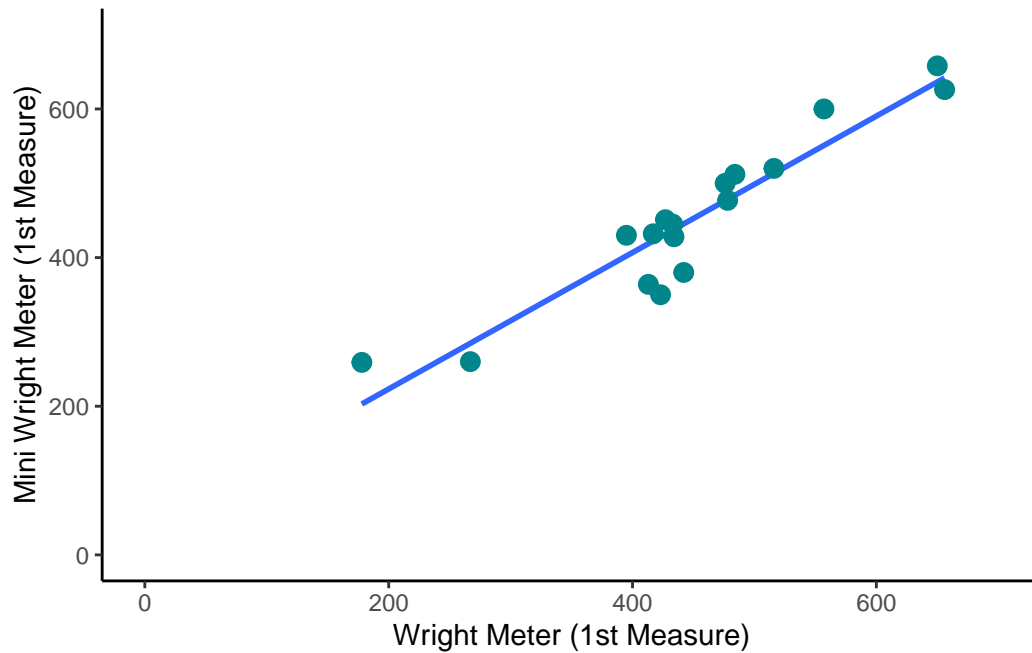
## Lecture 3: The Replication Crisis

Bland-Altman Plots are generated with the following code.

```
library(tidyverse)

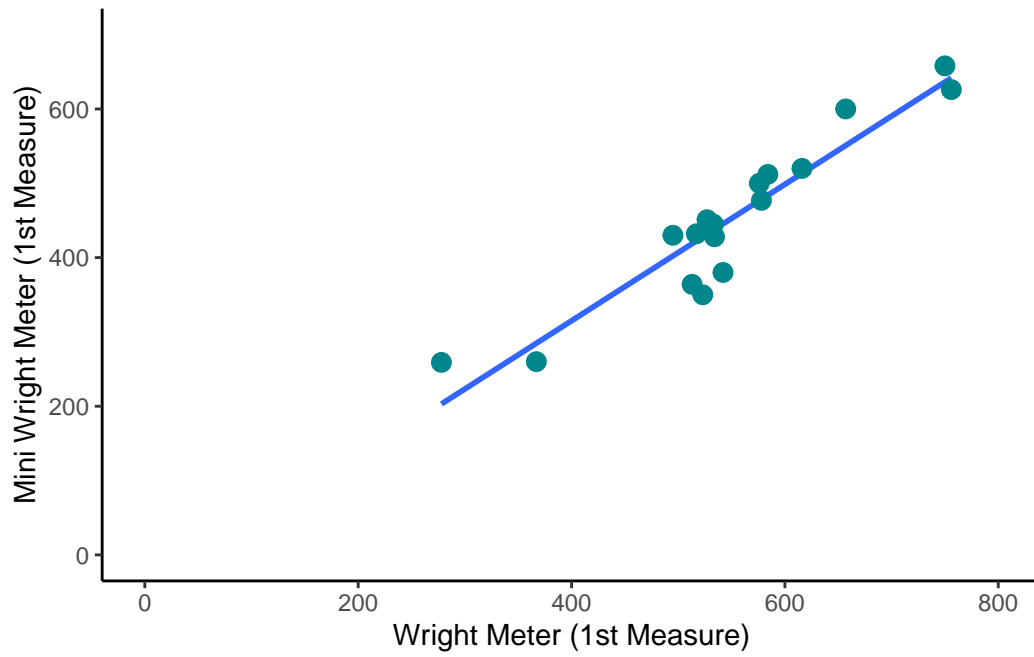
bland <- tibble(
  subject = c(1:17),
  Wright1 = c(484,395,516,434,476,557,413,442,650,433,417,656,267,478,178,423,427),
  Wright2 = c(490,397,512,401,470,611,415,431,638,429,420,633,275,492,165,372,421),
  Mini1 = c(512,430,520,428,500,600,364,380,658,445,432,626,260,477,259,350,451),
  Mini2 = c(525,415,508,444,500,625,460,390,642,432,420,605,227,467,268,370,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

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)")
```



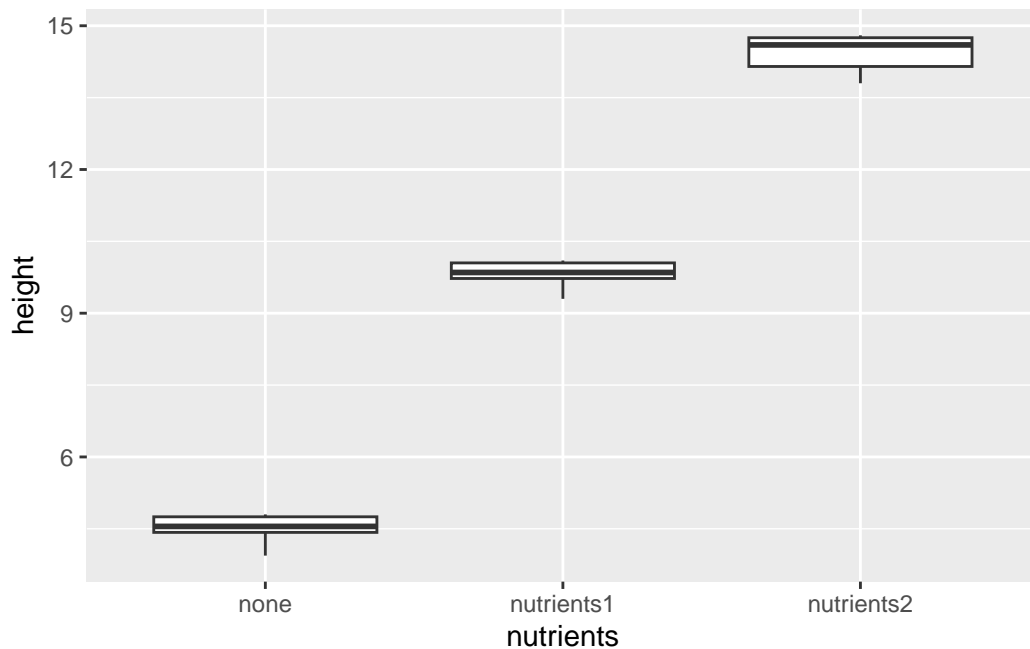
# Lecture 5: Introduction to Research Methods

## 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
```

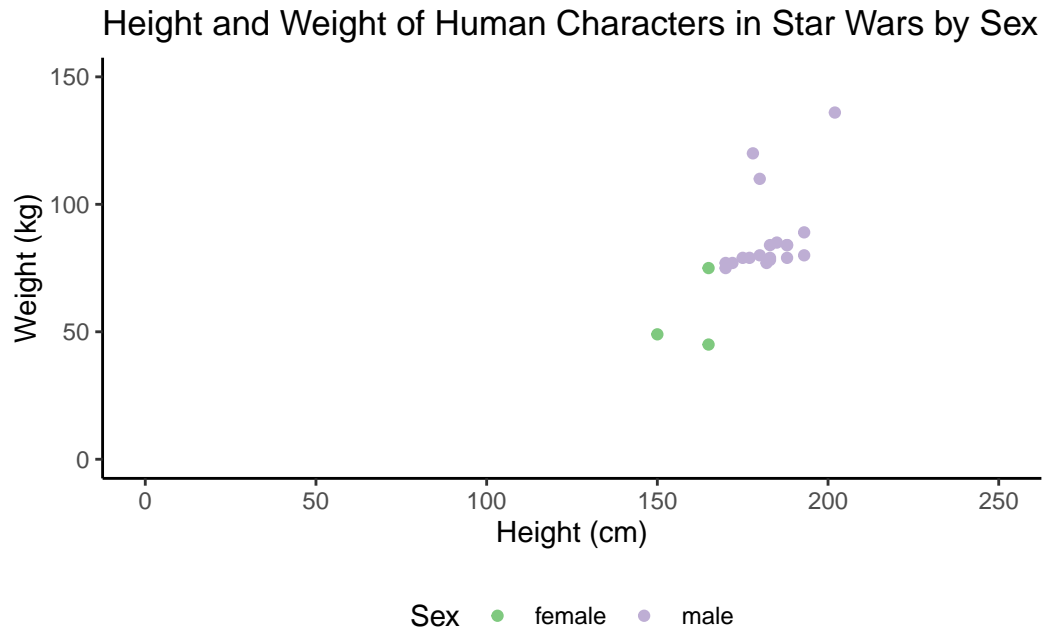
## **3 Week 2: The Use and Abuse of Data**

## Lecture 2: Data Visualisation

This code will help you replicate the charts in Lecture 2

### Height vs Weight by Sex

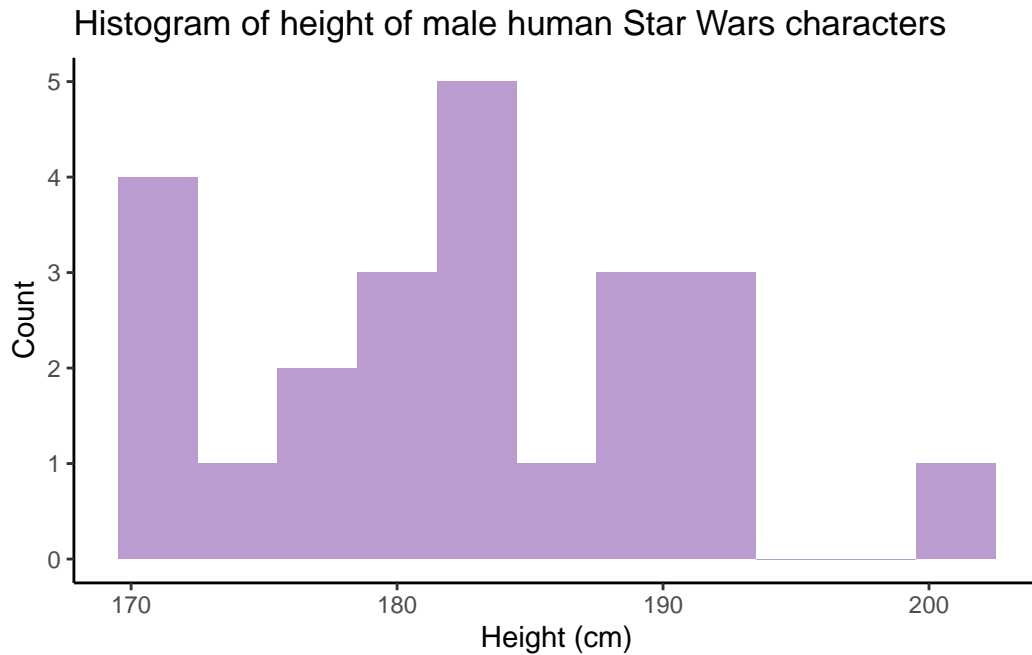
```
starwars |>
  filter(species == "Human") |>
  ggplot(aes(x = height, y = mass, colour = sex)) +
  geom_point() +
  theme_classic() +
  scale_x_continuous(limits = c(0,250)) +
  scale_y_continuous(limits = c(0,150)) +
  scale_colour_brewer(palette = "Accent", name = "Sex") +
  theme(legend.position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



## Histogram of male height

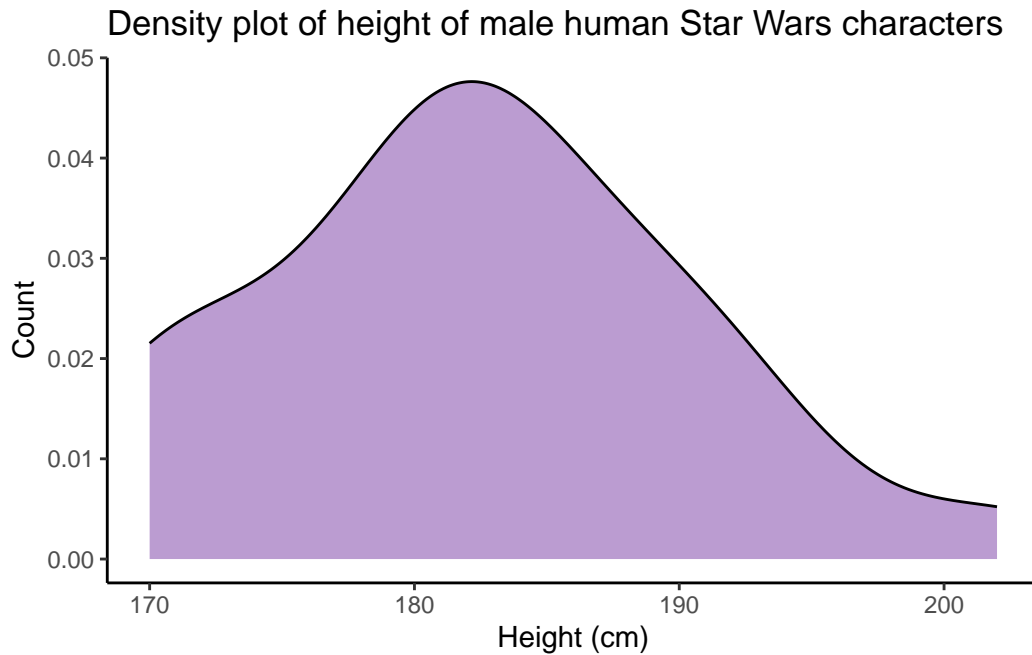
```
starwars |>
  filter(species == "Human",
         sex == "male") |>
  ggplot(aes(x = height)) +
  geom_histogram(binwidth = 3, fill = "#bb9cd1") +
  theme_classic() +
  labs(x = "Height (cm)",
       y = "Count",
       title = "Histogram of height of male human Star Wars characters")
```





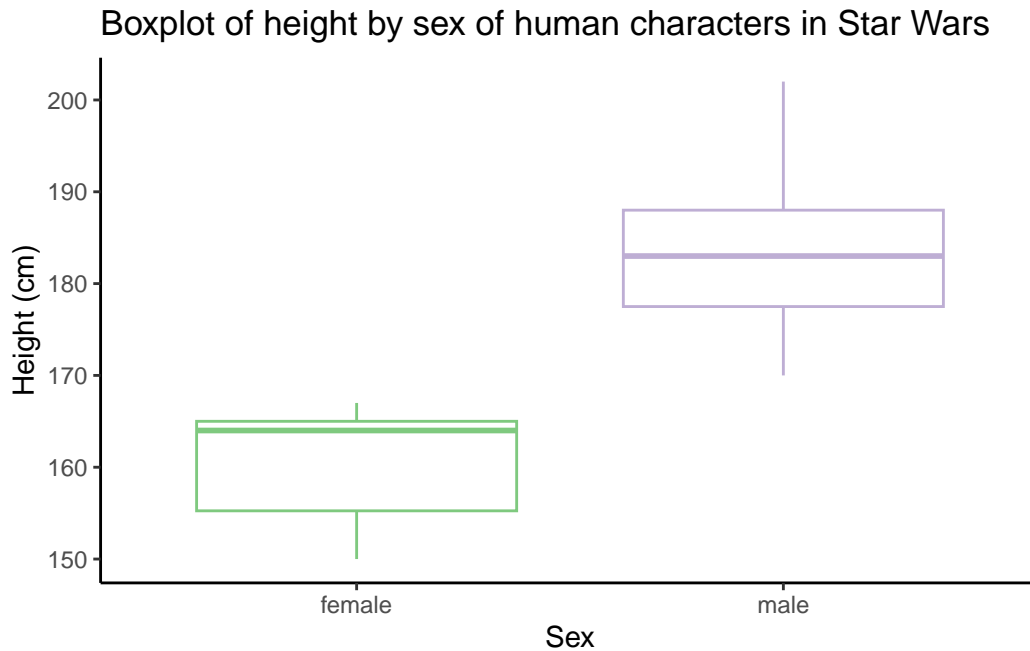
## Density plot of male height

```
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")
```



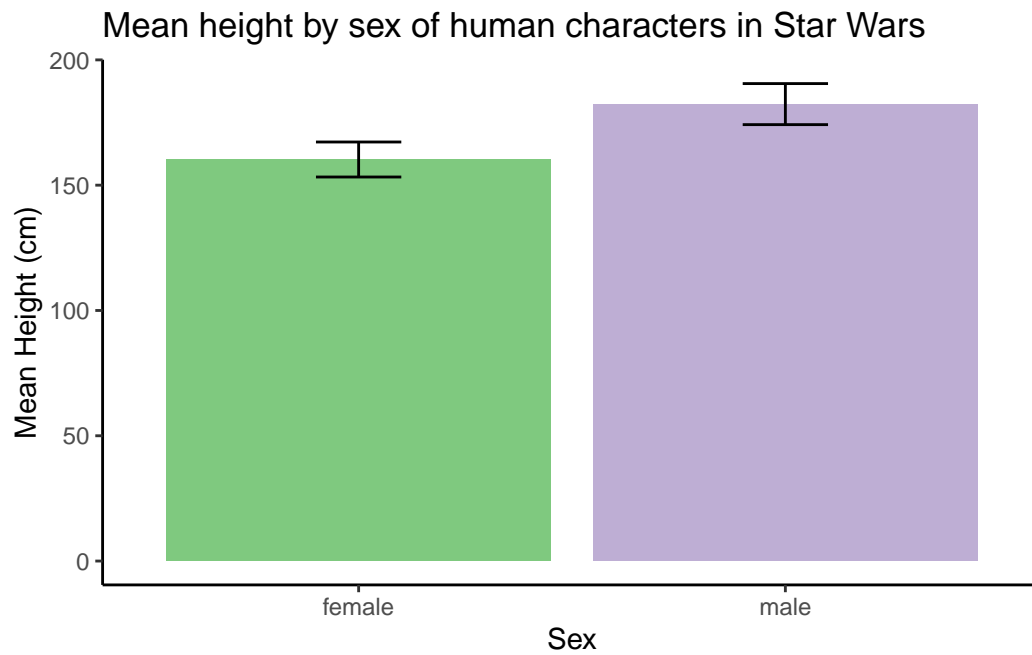
## Boxplot of height

```
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")
```

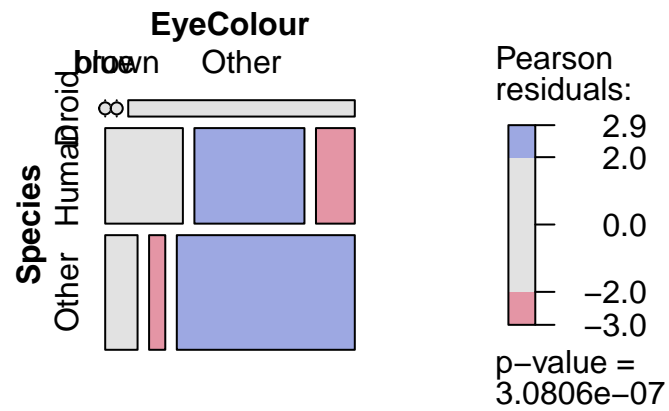


## Mosaic Plot

```
library(vcd)

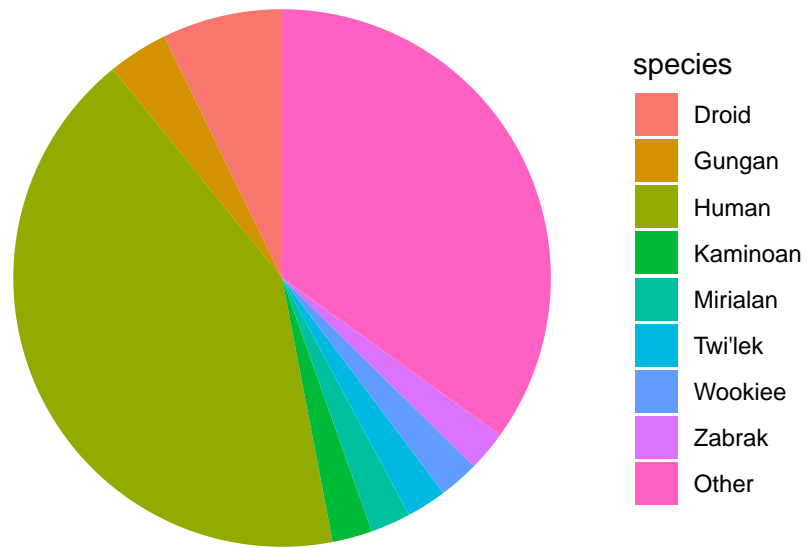
startbl <- starwars |>
  mutate(Species = fct_lump_n(species, 2),
         EyeColour = fct_lump_n(eye_color, 2))

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

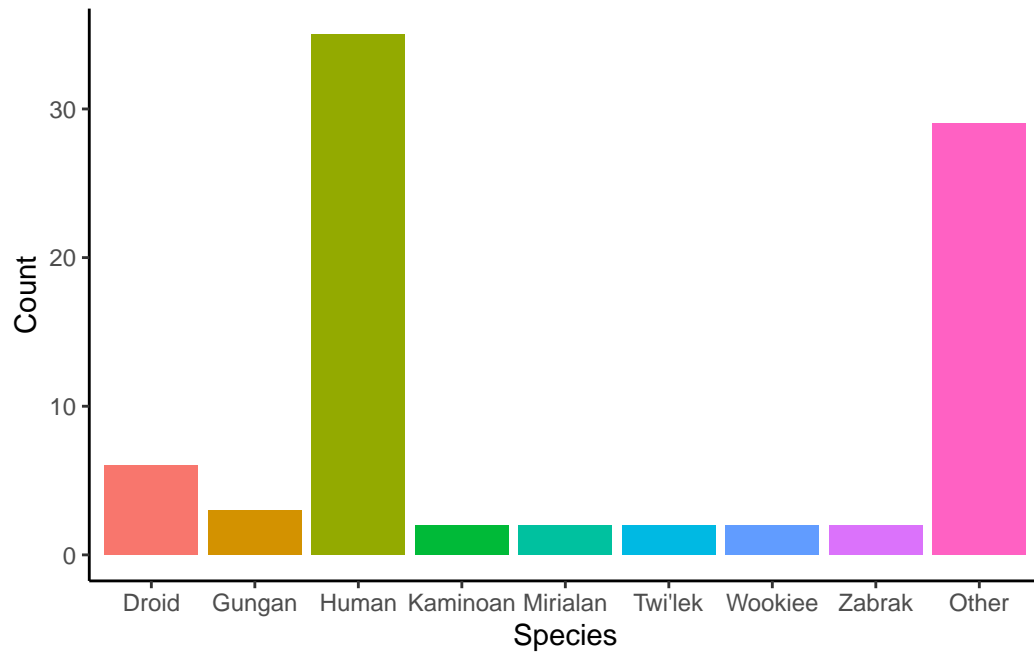


## Pie Charts are Just Bad Bar Charts

```
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

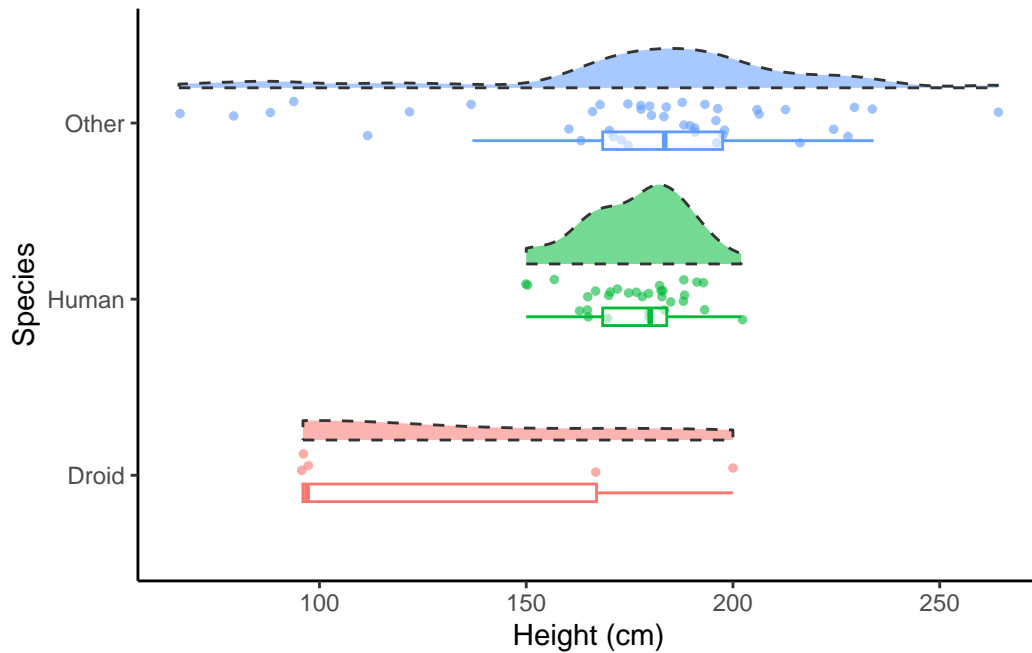
```
library(wordcloud)
starwars |>
  count(homeworld) |>
  with(wordcloud(words = homeworld, freq = n, min.freq=1, random.order = FALSE, rot.per =
    colors = brewer.pal(6, "Accent"), use.r.layout = FALSE))
```



## 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), size = 100) +
  see::geom_violinhalf(aes(y = height, alpha = 0.3, fill = species), linetype = "dashed", position = position_nudge(x = 1)) +
  geom_boxplot(aes(y = height, alpha = 0.3, colour = species), position = position_nudge(x = 1)) +
  theme_classic() +
  labs(x = "Species", y = "Height (cm)") +
  theme(legend.position = "none") +
  coord_flip()
```

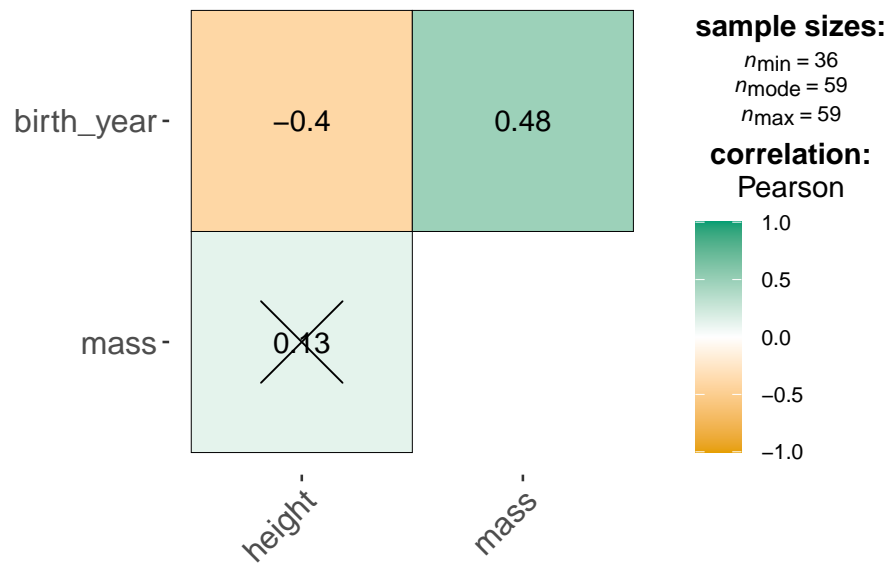




## Bubble plots

```
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")
```





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

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

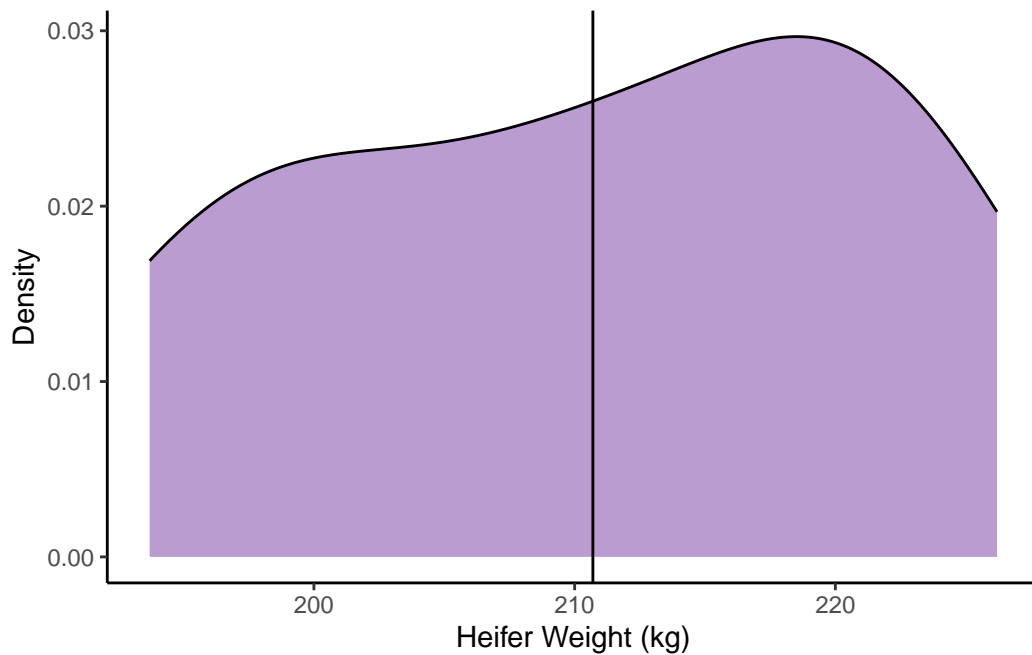
```
# A tibble: 4 x 5
   mean median   min   max mode
  <dbl>  <dbl> <dbl> <dbl> <dbl>
1 34475.   31988 17859 84131 28216
2 34475.   31988 17859 84131 35248
3 34475.   31988 17859 84131 26000
4 34475.   31988 17859 84131 25000
```

## The Mean and Outliers

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

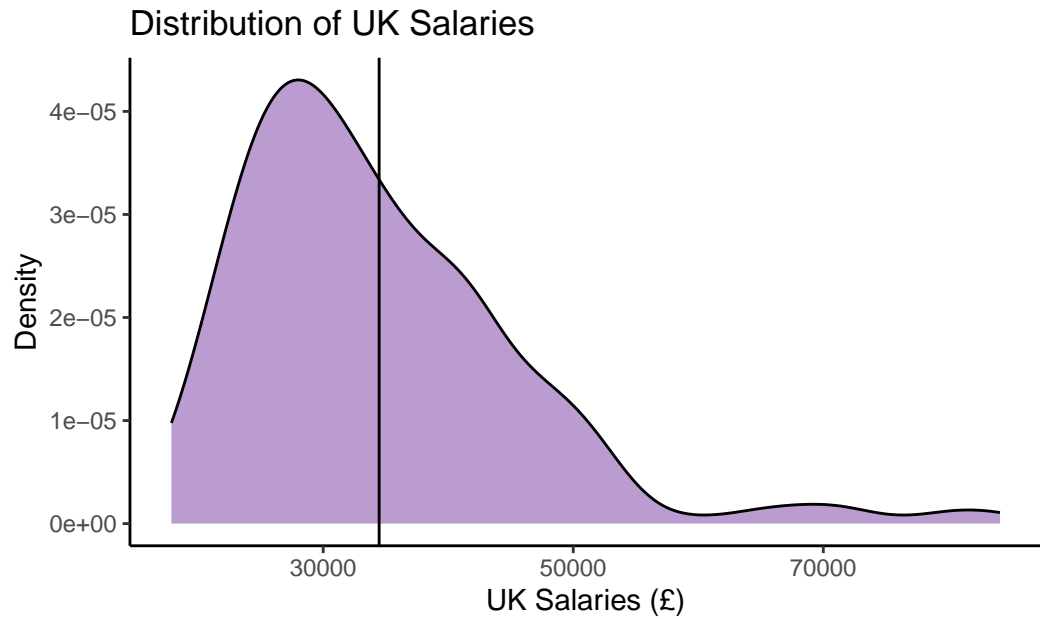


```
heifers |>
  ggplot(aes(x = heifers)) +
  geom_density(fill = "#bb9cd1") +
  geom_vline(aes(xintercept = 210.7)) +
  theme_classic() +
  labs(x = "Heifer Weight (kg)",
       y = "Density")
```



## 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")
```

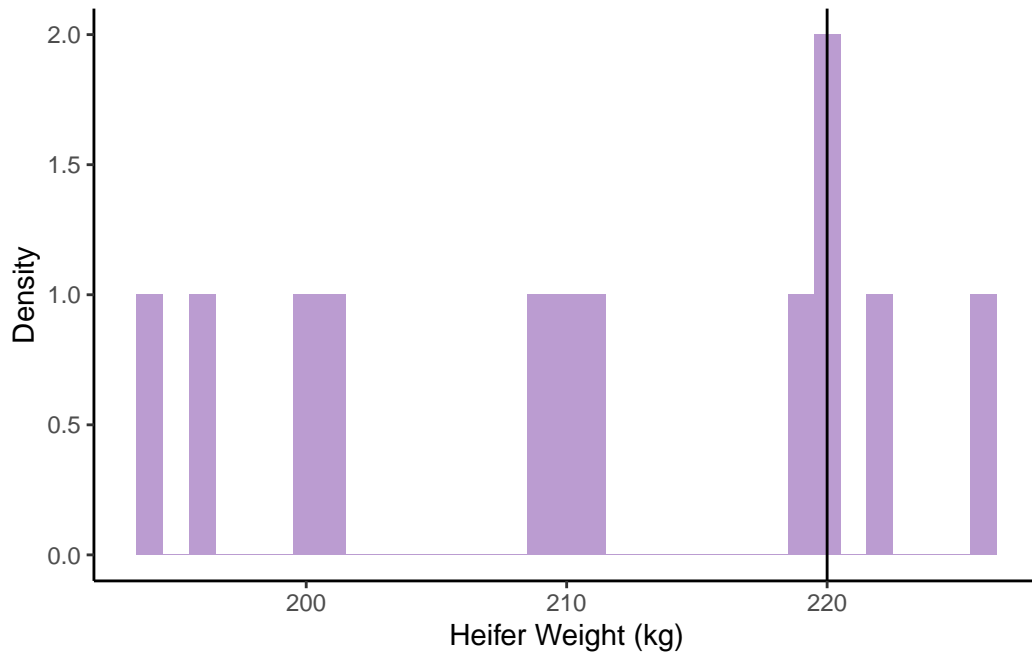


Data taken from ONS 2023 Median Salaries by Field, n = 329 fields

## The Mode

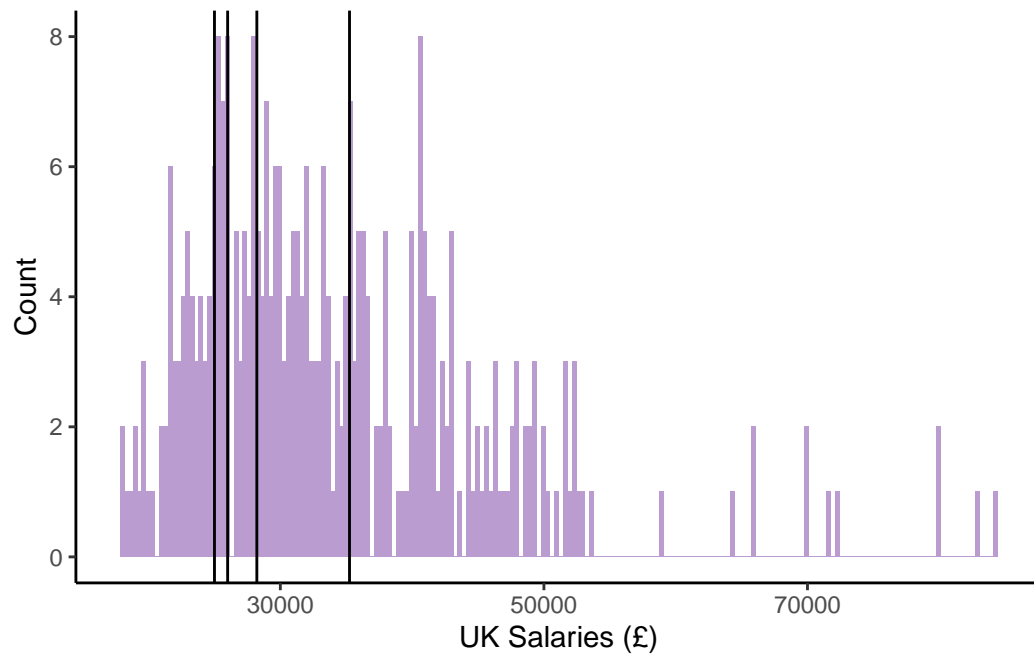
```
heifers |>
  ggplot(aes(x = heifers)) +
  geom_histogram(fill = "#bb9cd1", binwidth = 1) +
  geom_vline(aes(xintercept = 220)) +
  theme_classic() +
  labs(x = "Heifer Weight (kg)",
       y = "Density")
```





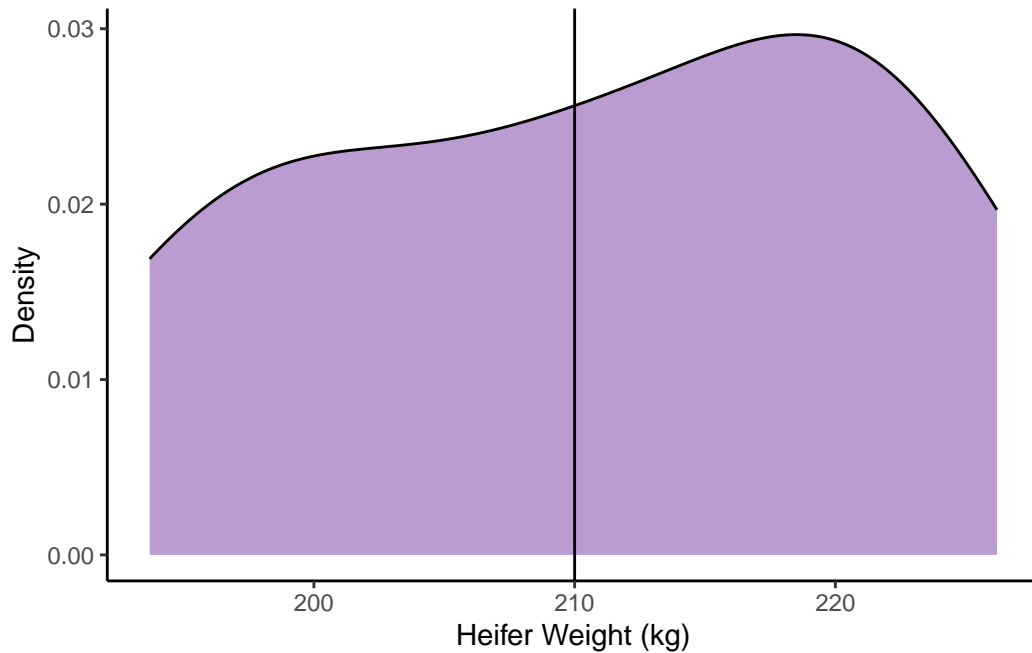
## Multiple Modes

```
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")
```



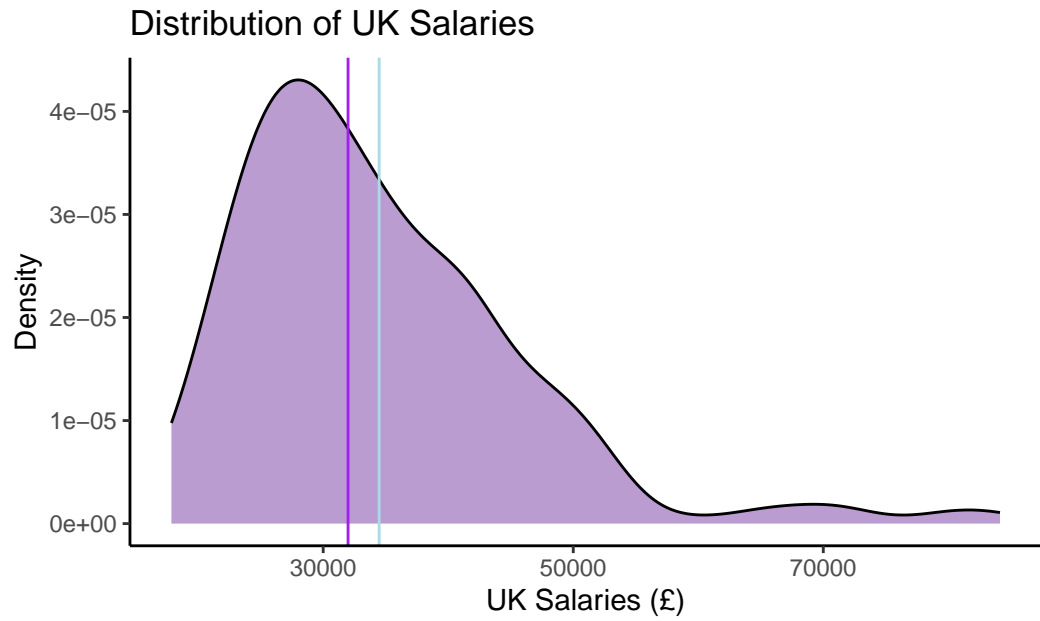
## The Median

```
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, 3.7, 3.8, 3.9),
  environ = c("Indoor", "Indoor", "Outdoor", "Indoor",
    "Outdoor", "Indoor", "Outdoor", "Indoor",
    "Indoor", "Indoor", "Outdoor", "Indoor",
    "Outdoor", "Indoor", "Indoor", "Outdoor"))
```

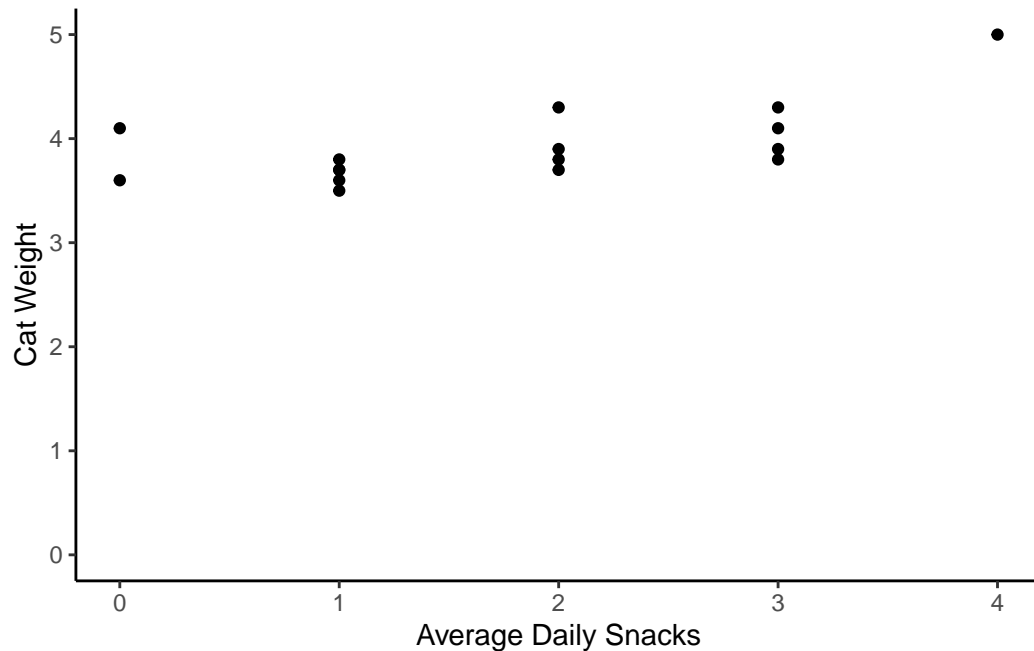
## Summarise example data

```
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 ( $R^2 = 0.41$ ,  $F(1, 14) = 9.65$ ,  $p = 0.008$ , adj.  $R^2 = 0.37$ ). 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])

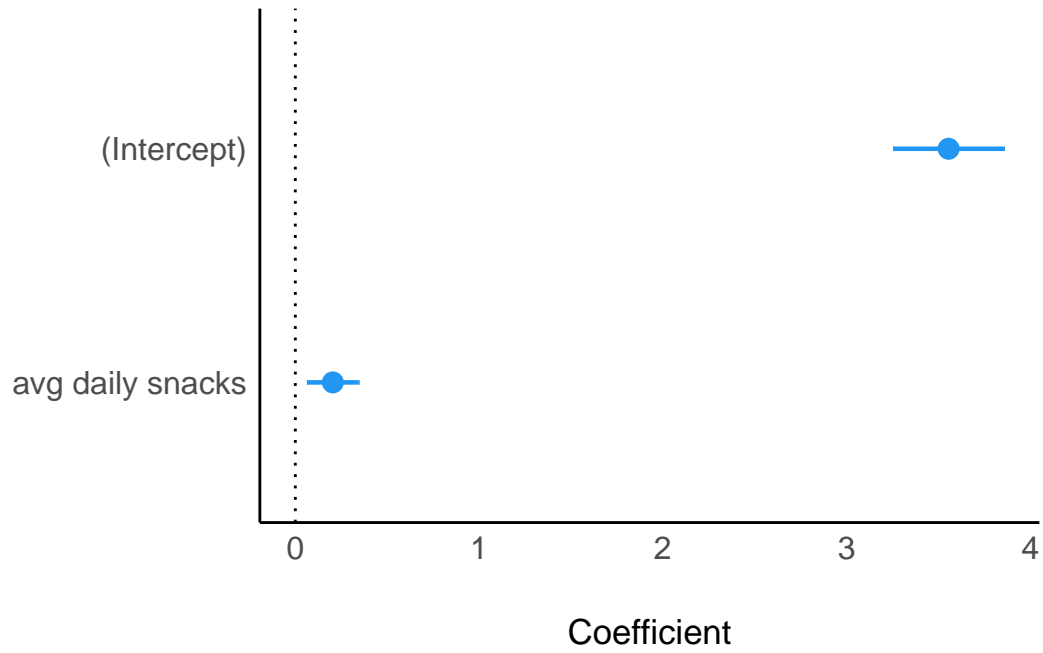
Standardized parameters were obtained by fitting the model on a standardized version of the dataset. 95% Confidence Intervals (CIs) and p-values were computed using a Wald t-distribution approximation.

```
parameters(model_fcat)
```

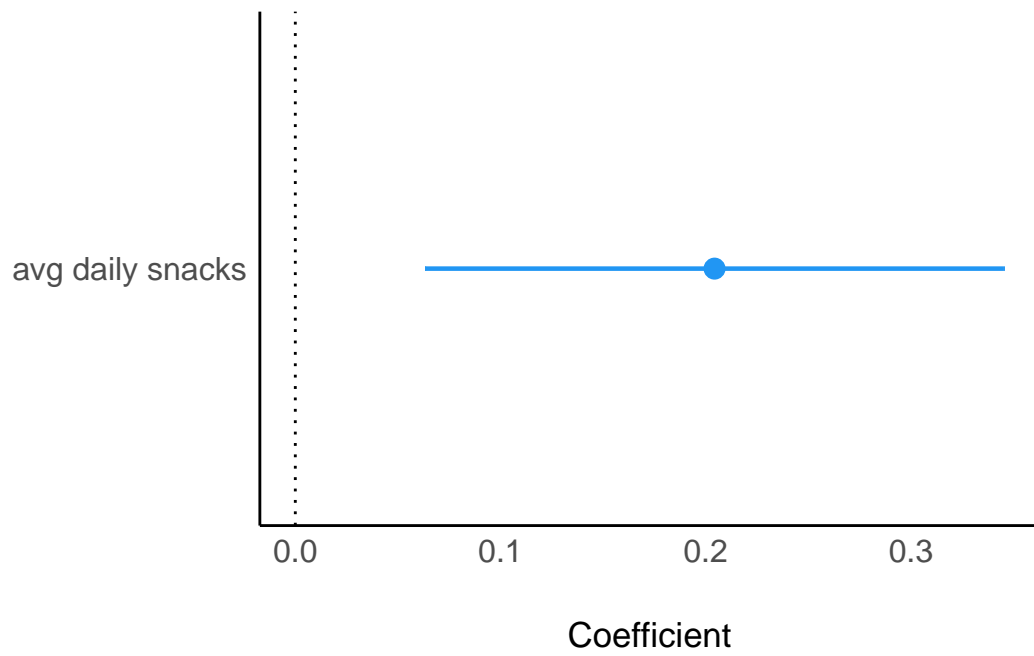
Parameter	Coefficient	SE	95% CI	t(14)	p
(Intercept)	3.55	0.14	[3.25, 3.86]	25.31	< .001
avg daily snacks	0.20	0.07	[0.06, 0.35]	3.11	0.008

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

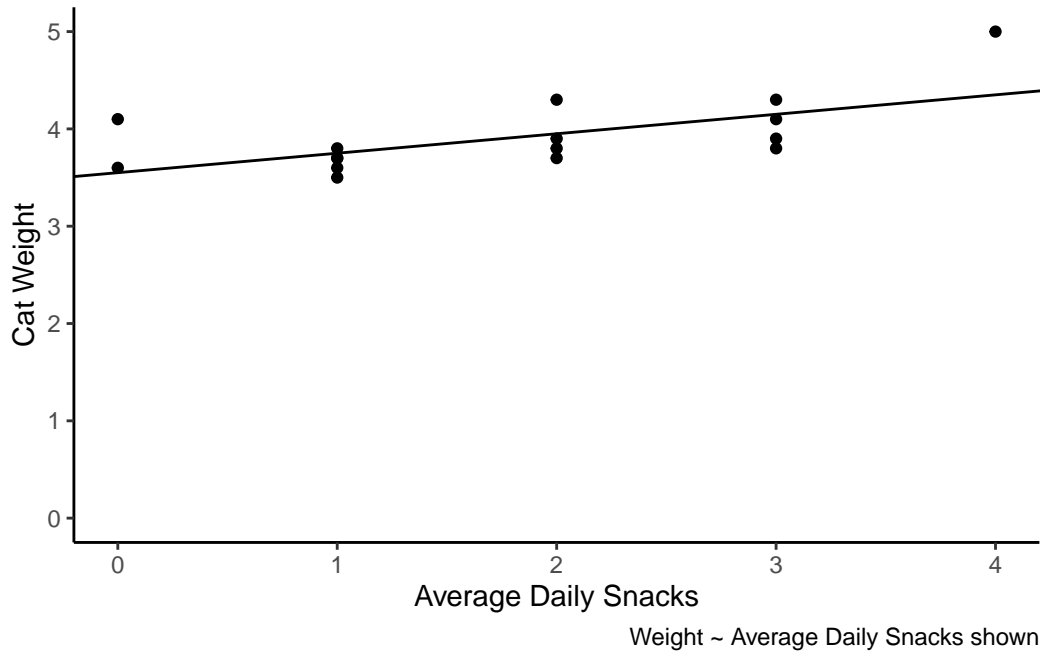




```
plot(model_parameters(model_fcat))
```



```
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)
```



## A Bayesian Model

```
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:

```

```
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	
(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 ( $R^2 = 0.38$ , 95% CI [6.54e-06, 0.62], adj.  $R^2 = 0.15$ ). 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

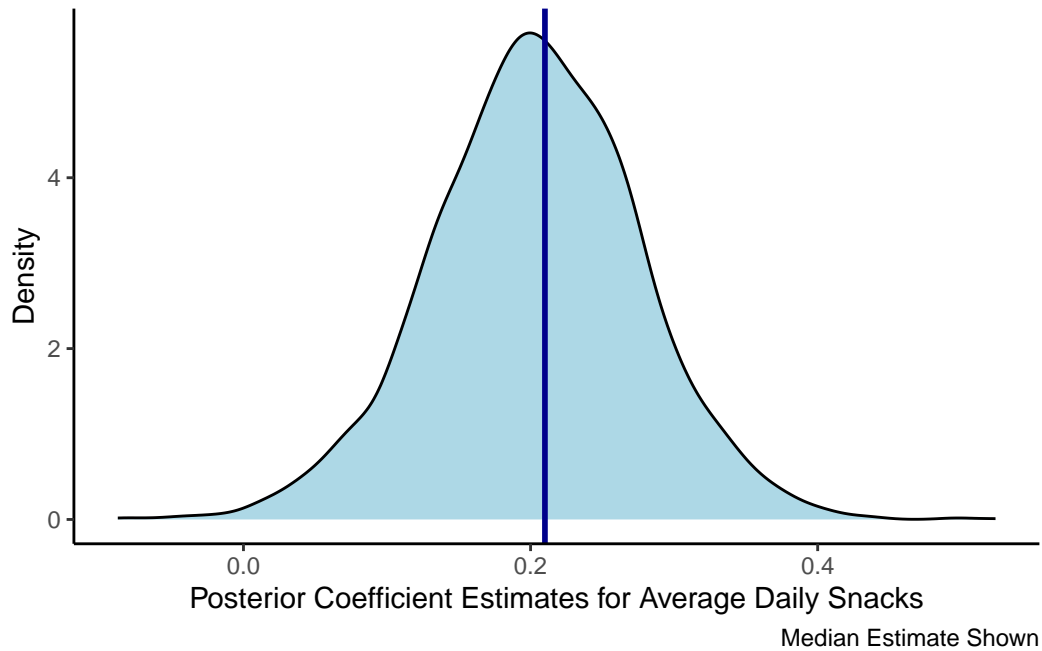


99.62% probability of being positive ( $> 0$ ), 99.30% of being significant ( $> 0.02$ ), and 90.10% of being large ( $> 0.11$ ). The estimation successfully converged ( $R_{\text{hat}} = 1.000$ ) and the indices are reliable ( $\text{ESS} = 2981$ )

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 ( $R^2 = 0.53$ ,  $F(2, 13) = 7.32$ ,  $p = 0.007$ , adj.  $R^2 = 0.46$ ). 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:

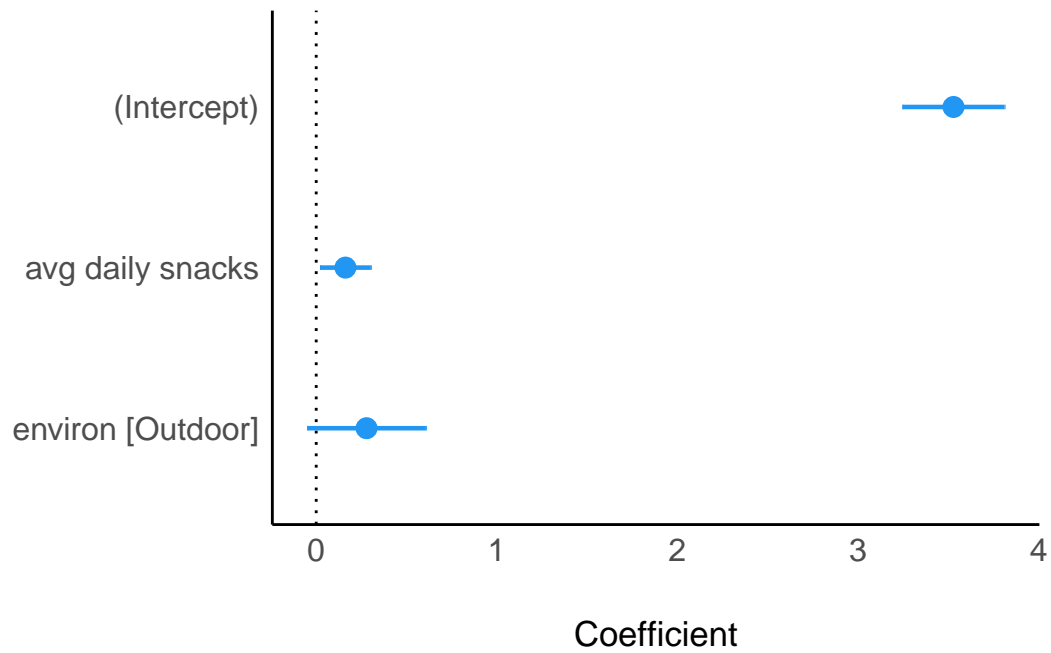
- The effect of avg daily snacks is statistically significant and positive (beta = 0.16, 95% CI [0.02, 0.30],  $t(13) = 2.48$ ,  $p = 0.027$ ; Std. beta = 0.51, 95% CI [0.07, 0.95])
- The effect of environ [Outdoor] is statistically non-significant and positive (beta = 0.28, 95% CI [-0.05, 0.61],  $t(13)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

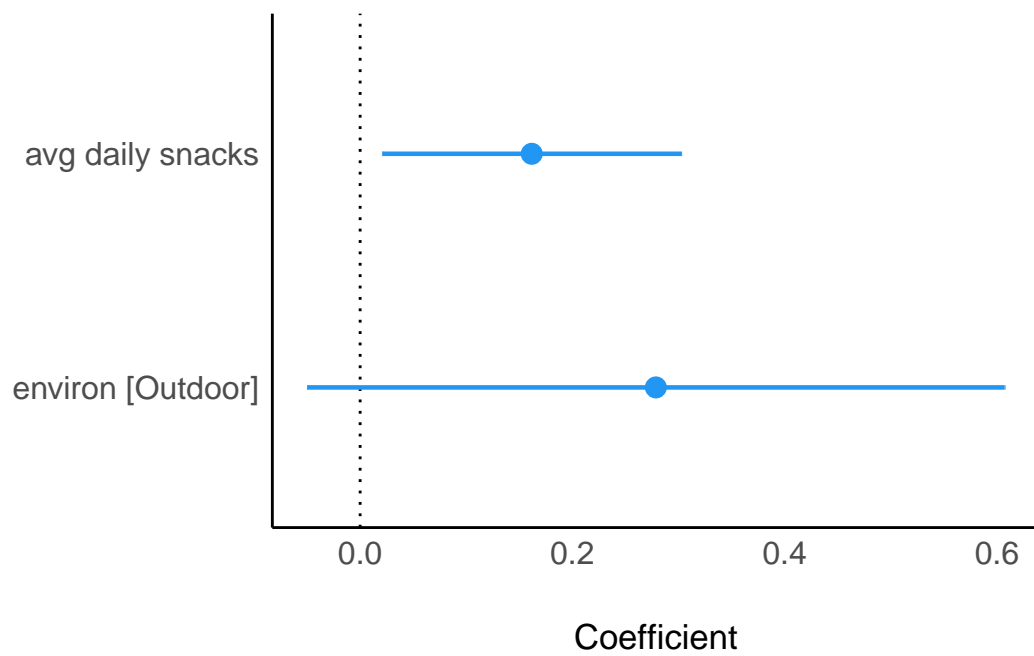
```
parameters(model_fcat2)
```

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

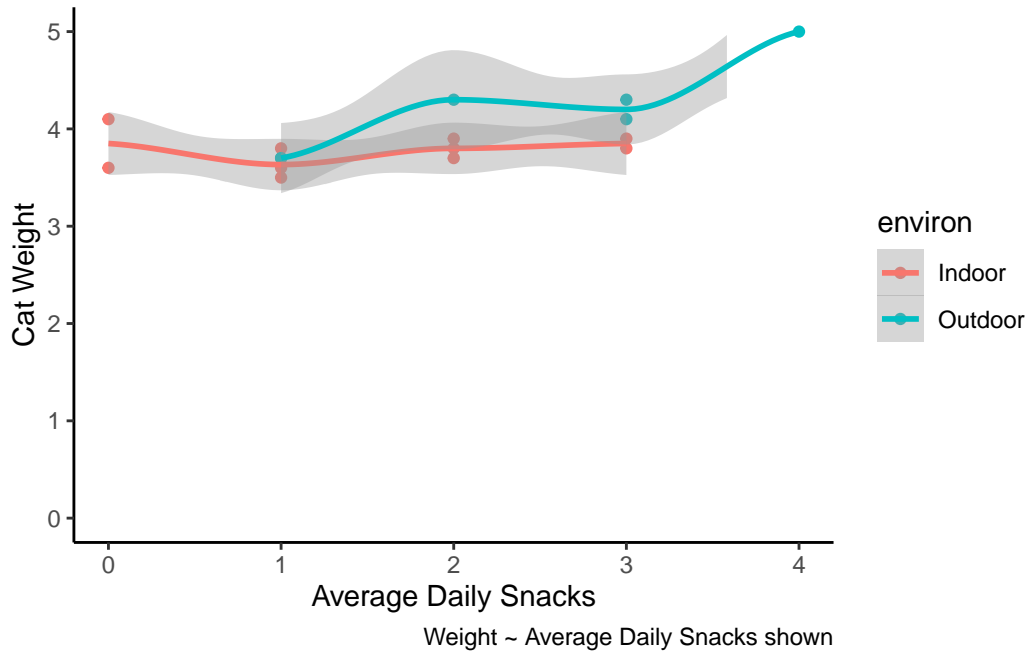
```
plot(model_parameters(model_fcat2), show_intercept = TRUE)
```



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



```
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

```
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
avg_daily_snacks	0.16	[ 0.02, 0.30]	98.90%	[-0.04, 0.04]	1.55%	1.001
environOutdoor	0.28	[-0.04, 0.62]	95.65%	[-0.04, 0.04]	4.18%	1.000

```
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:

```

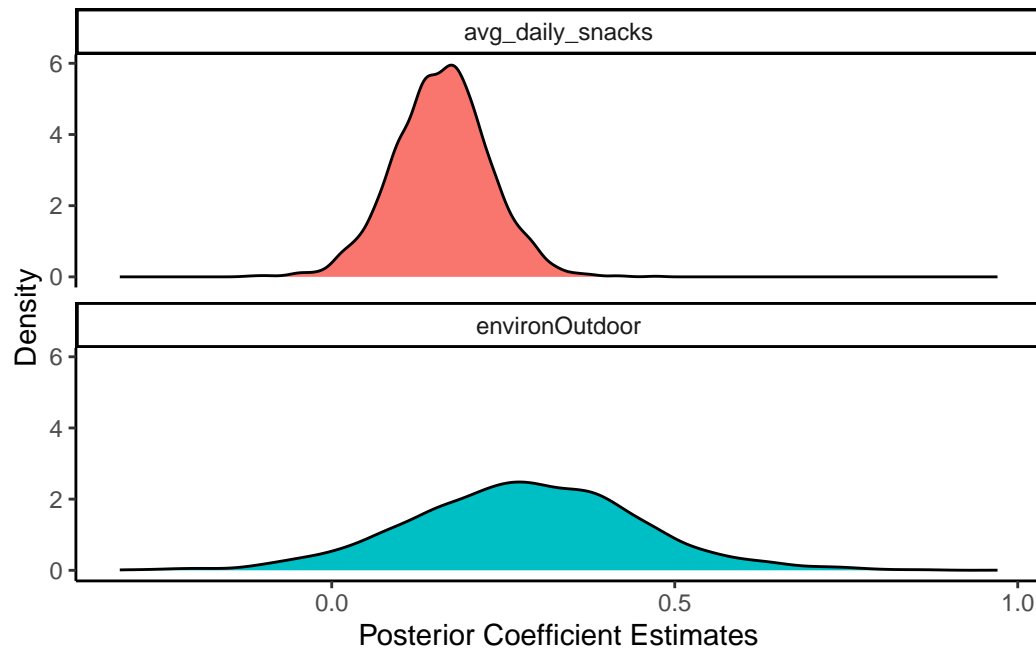
We fitted a Bayesian linear model (estimated using MCMC sampling with 4 chains of 2000 iterations and a warmup of 1000) to predict weight with avg\_daily\_snacks and environ (formula:  $\text{weight} \sim \text{avg\_daily\_snacks} + \text{environ}$ ). Priors over parameters were all set as normal (mean = 0.00, SD = 0.80; mean = 0.00, SD = 1.87) distributions. The model's explanatory power is substantial ( $R^2 = 0.50$ , 95% CI [0.17, 0.74], adj.  $R^2 = 0.26$ ). 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).

```
posterior2 <- get_parameters(model_bcat2)

posterior2 |>
  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")
```

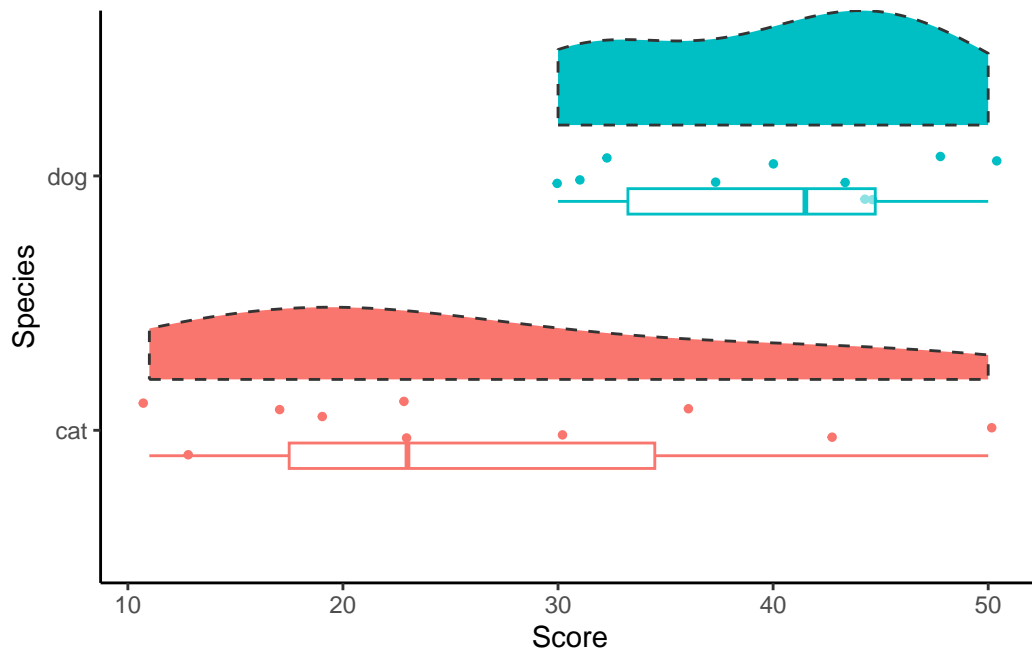


# Lecture: Calculating variance

## Why does it matter?

```
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))

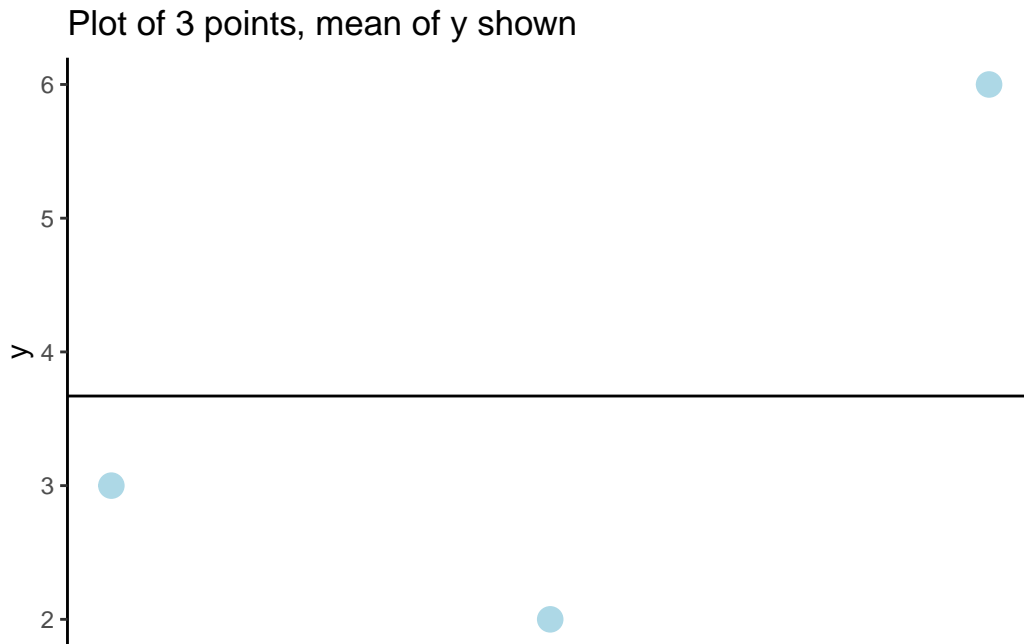
vardat |>
  pivot_longer(cols = c(cat, dog),
               names_to = "Species",
               values_to = "Score") |>
  ggplot(aes(x = Species)) +
  geom_point(aes(y = Score, colour = Species), position = position_jitter(width = .13), si
see::geom_violinhalf(aes(y = Score, fill = Species), linetype = "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))

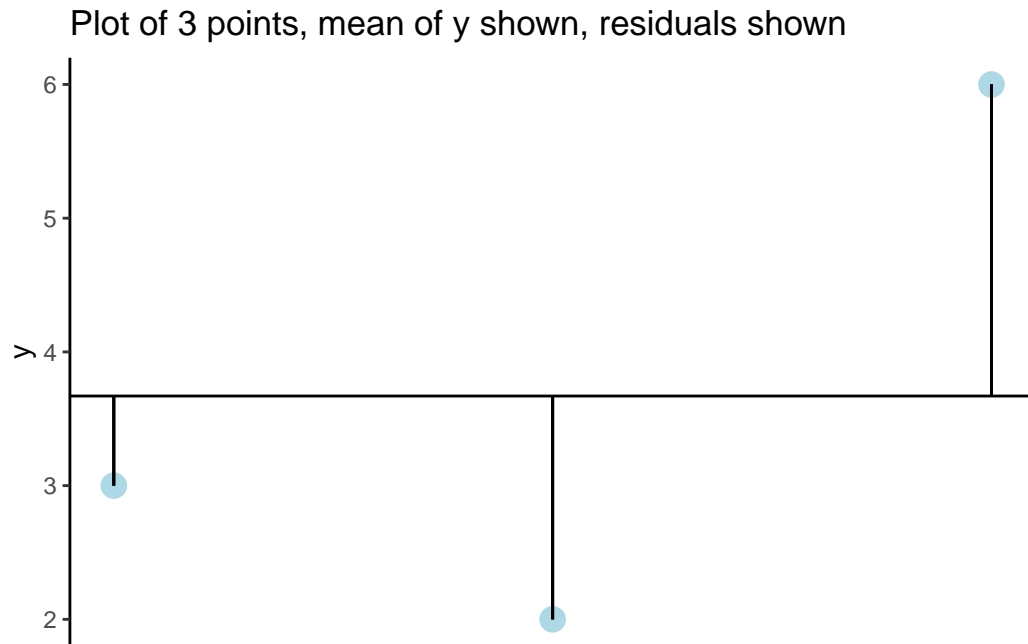
resid |>
  ggplot(aes(x, y)) +
  geom_point(size = 4, colour = "lightblue") +
  theme_classic() +
  geom_hline(yintercept = 3.67) +
  labs(title = "Plot of 3 points, mean of y shown") +
  theme(axis.text.x=element_blank(),
        axis.ticks.x=element_blank(),
        axis.title.x = element_blank())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



## 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**

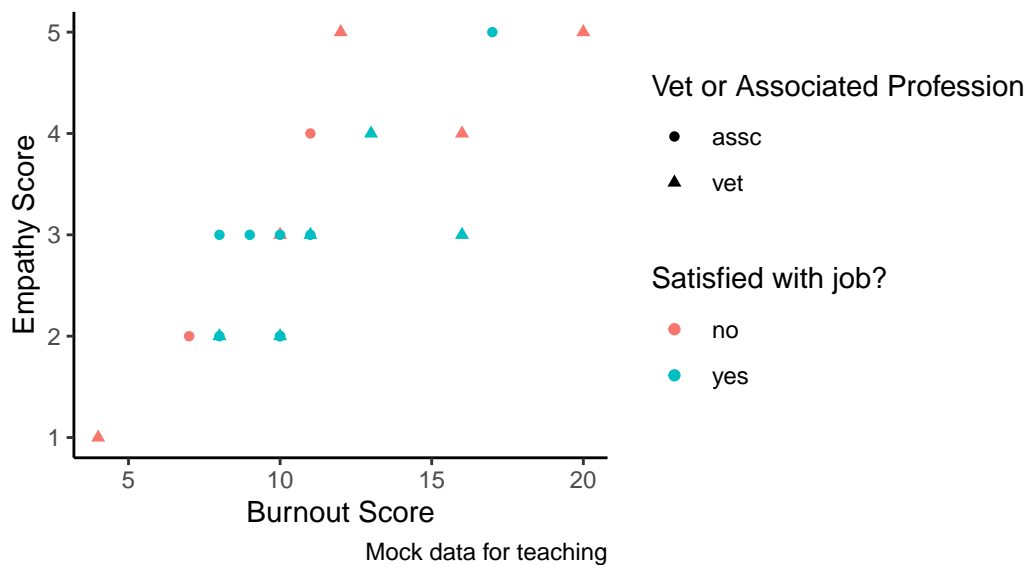
# Lecture: Effect Sizes and Covariance

## Mock Data and visualisation

```
job_dat <- tibble(job = c("vet", "vet", "vet", "vet", "vet", "vet", "vet", "vet", "vet", "vet", "v  
      "assc", "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?")
```

Burnout and empathy scores for vets and associated professions:  
Job Satisfaction shown



## Calculate 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
```

## Calculate Hedge's g

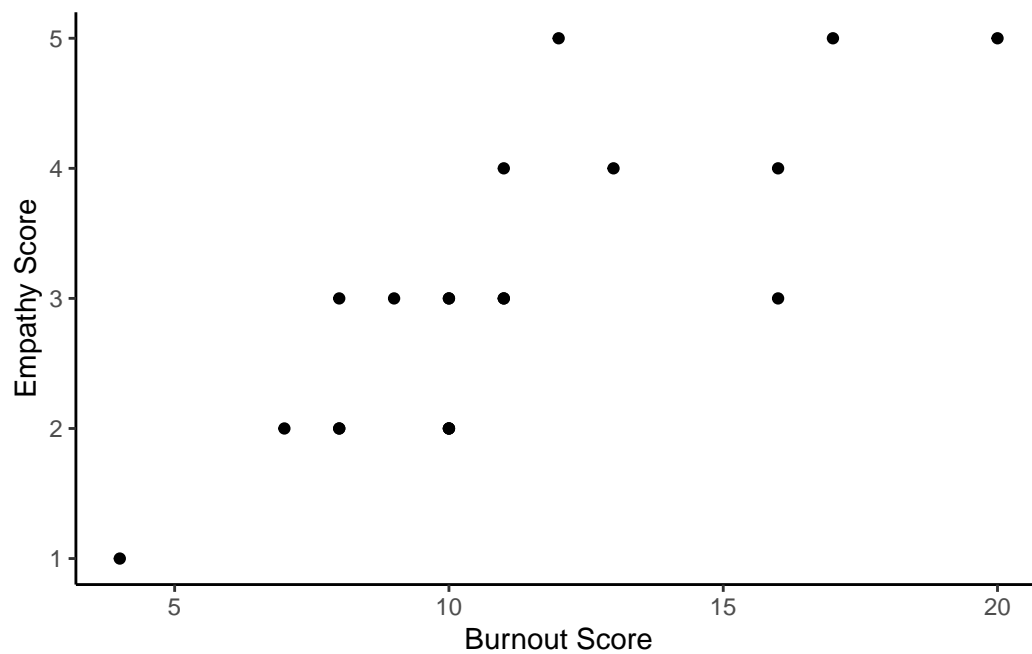
```
cohen.d(d = job_dat$burnout, f = job_dat$job, hedges.correction = TRUE)
```

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_tbl <- xtabs(~job_dat$burnout + job_dat$satisfaction)
ftable(job_tbl)
```



	job_dat\$satisfaction	
job_dat\$burnout	no	yes
4	1	0
7	1	0
8	0	3
9	0	1
10	1	4
11	1	2
12	1	0
13	0	1
16	1	1
17	0	1
20	1	0

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

Pearson's Chi-squared test

```
data:  ftable(job_tbl)
X-squared = 11.355, df = 10, p-value = 0.3305
```

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

```
[1] 0.7535022
```

## 6 Week 5: Sources of Data

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

## **7 Week 6: Analysing Qualitative Data**

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

## 8 Week 7: Analysing Quantitative Data

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

## **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 Bueros Novik, and R Wolfe. 2018. “Joint Longitudinal and Time-to-Event Models via Stan.” [https://github.com/stan-dev/stancon\\_talks/](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>.