

Elements of Statistical Modeling for Experimental Biology

Copyright 2018 Jeffrey A. Walker

Draft: 2020-10-25

Contents

Preface	5
0.1 Math	8
0.2 R and programming	8
Part I: Getting Started	9
1 Getting Started – R Projects and R Markdown	11
1.1 R vs R Studio	12
1.2 Download and install R and R studio	12
1.3 Install R Markdown	12
1.4 Importing Packages	12
1.5 Create an R Studio Project for this textbook	14
Part II: An introduction to the analysis of experimental data with a linear model	19
2 Analyzing experimental data with a linear model	21
2.1 This text is about the estimation of treatment effects and the uncertainty in our estimates using linear models. This, raises the question, what is “an effect”?	21
Background physiology to the experiments in Figure 2 of “ASK1 inhibits browning of white adipose tissue in obesity”	29

Analyses for Figure 2 of “ASK1 inhibits browning of white adipose tissue in obesity”	33
2.2 Setup	33
2.3 Data source	34
2.4 control the color palette	34
2.5 useful functions	34
2.6 figure 2b – effect of ASK1 deletion on growth (body weight)	36
2.7 Figure 2c – Effect of ASK1 deletion on final body weight	38
2.8 Figure 2d – Effect of ASK1 KO on glucose tolerance (whole curve)	48
2.9 Figure 2e – Effect of ASK1 deletion on glucose tolerance (summary measure)	50
2.10 Figure 2f – Effect of ASK1 deletion on glucose infusion rate	55
2.11 Figure 2g – Effect of ASK1 deletion on tissue-specific glucose uptake	58
2.12 Figure 2h	63
2.13 Figure 2i – Effect of ASK1 deletion on liver TG	63
2.14 Figure 2j	67
Part III: R fundamentals	69
3 Data – Reading, Wrangling, and Writing	71
3.1 Learning from this chapter	73
3.2 Working in R	74
3.3 Data wrangling	83
3.4 Saving data	112
3.5 Exercises	113
4 Plotting Models	117
4.1 Pretty good plots show the model and the data	118
4.2 Some comments on plot components	121
4.3 Working in R	124
Part IV: Some Fundamentals of Statistical Modeling	143

CONTENTS	5
5 Variability and Uncertainty (Standard Deviations, Standard Errors, Confidence Intervals)	145
5.1 The sample standard deviation vs. the standard error of the mean	146
5.2 Using Google Sheets to generate fake data to explore the standard error	149
5.3 Using R to generate fake data to explore the standard error	150
5.4 Bootstrapped standard errors	154
5.5 Confidence Interval	157
6 P-values	161
6.1 A <i>p</i> -value is the probability of sampling a value as or more extreme than the test statistic if sampling from a null distribution	163
6.2 Pump your intuition – Creating a null distribution	165
6.3 A null distribution of <i>t</i> -values – the <i>t</i> distribution	167
6.4 P-values from the perspective of permutation	171
6.5 Parametric vs. non-parametric statistics	173
6.6 frequentist probability and the interpretation of p-values	173
6.7 Some major misconceptions of the <i>p</i> -value	178
6.8 What the <i>p</i> -value does not mean	183
6.9 Recommendations	184
6.10 Problems	185
7 Errors in inference	187
7.1 Classical NHST concepts of wrong	187
7.2 A non-Neyman-Pearson concept of power	193
Part V: Introduction to Linear Models	195
8 An introduction to linear models	197
8.1 Two specifications of a linear model	197
8.2 A linear model can be fit to data with continuous, discrete, or categorical <i>X</i> variables	201
8.3 Statistical models are used for prediction, explanation, and description	204

8.4 What do we call the X and Y variables?	205
8.5 Modeling strategy	206
8.6 Predictions from the model	207
8.7 Inference from the model	207
8.8 “linear model,”“regression model”, or “statistical model”?	211
9 Linear models with a single, continuous X	213
9.1 A linear model with a single, continuous X is classical “regression”	213
9.2 Working in R	232
9.3 Hidden code	246
9.4 Try it	253
9.5 Intuition pumps	254
10 Linear models with a single, categorical X	257
10.1 A linear model with a single, categorical X variable estimates the effects of the levels of X on the response.	257
10.2 Working in R	281
10.3 Issues in inference in models with a single, categorical X	292
10.4 Hidden Code	297
11 Model Checking	301
11.1 All statistical analyses should be followed by model checking	301
11.2 Linear model assumptions	302
11.3 Diagnostic plots use the residuals from the model fit	304
11.4 Using R	313
12 Model Fitting and Model Fit (OLS)	317
12.1 Least Squares Estimation and the Decomposition of Variance	317
12.2 OLS regression	318
12.3 How well does the model fit the data? R^2 and “variance explained”	319
13 Best practices – issues in inference	323
13.1 Multiple testing	323
13.2 difference in p is not different	331
13.3 Inference when data are not Normal	331

CONTENTS	7
Part VI: More than one X – Multivariable Models	345
14 Adding covariates to a linear model	347
14.1 Adding covariates can increases the precision of the effect of interest	347
14.2 Adding covariates can decrease prediction error in predictive models	352
14.3 Adding covariates can reduce bias due to confounding in explanatory models	352
14.4 Best practices 1: A pre-treatment measure of the response should be a covariate and not subtracted from the post-treatment measure (regression to the mean)	352
14.5 Best practices 2: Use a covariate instead of normalizing a response	359
15 Two (or more) Categorical X – Factorial designs	361
15.1 Factorial experiments	361
15.2 Reporting results	378
15.3 Working in R	378
15.4 Problems	388
16 ANOVA Tables	391
16.1 Summary of usage	391
16.2 Example: a one-way ANOVA using the vole data	392
16.3 Example: a two-way ANOVA using the urchin data	394
16.4 Unbalanced designs	404
16.5 Working in R	414
17 Predictive Models	417
17.1 Overfitting	417
17.2 Model building vs. Variable selection vs. Model selection	418
17.3 Shrinkage	418
Part VII – Expanding the Linear Model	419

18 Linear mixed models	421
18.1 Random effects	421
18.2 Random effects in statistical models	424
18.3 Linear mixed models are flexible	426
18.4 Blocking	427
18.5 Pseudoreplication	432
18.6 Mapping NHST to estimation: A paired t-test is a special case of a linear mixed model	436
18.7 Advanced topic – Linear mixed models shrink coefficients by par- tial pooling	437
18.8 Working in R	442
19 Generalized linear models I: Count data	447
19.1 The generalized linear model	448
19.2 Count data example – number of trematode worm larvae in eyes of threespine stickleback fish	450
19.3 Working in R	462
19.4 Problems	467
20 Linear models with heterogenous variance	469
20.1 gls	469
Part V: Expanding the Linear Model – Generalized Linear Models and Multilevel (Linear Mixed) Models	471
21 Plotting functions (#ggplotsci)	473
21.1 odd-even	475
21.2 estimate response and effects with emmeans	475
21.3 emm_table	476
21.4 pairs_table	476
21.5 gg_mean_error	478
21.6 gg_ancova	482
21.7 gg_mean_ci_ancova	485
21.8 gg_effects	489

CONTENTS	9
Appendix 1: Getting Started with R	493
21.9 Get your computer ready	493
21.10 Start learning R Studio	494
Appendix 2: Online Resources for Getting Started with Statistical Modeling in R	495
Appendix 3: Fake Data Simulations	497
21.11 Performance of Blocking relative to a linear model	497

Preface

More cynically, one could also well ask “Why has medicine not adopted frequentist inference, even though everyone presents P-values and hypothesis tests?” My answer is: Because frequentist inference, like Bayesian inference, is not taught. Instead everyone gets taught a misleading pseudo-frequentism: a set of rituals and misinterpretations caricaturing frequentist inference, leading to all kinds of misunderstandings. – Sander Greenland

We use statistics to learn from data with uncertainty. Traditional introductory textbooks in biostatistics implicitly or explicitly train students and researchers to “discover by p-value” using hypothesis tests (Chapter 6). Over the course of many chapters, the student learns to use a look-up table or flowchart to choose the correct “test” for the data at hand, compute a test statistic for their data, compute a p -value based on the test statistic, and compare the p -value to 0.05. Textbooks typically give very little guidance about what can be concluded if $p < 0.05$ or if $p > 0.05$, but many researchers conclude, incorrectly, they have “discovered” an effect if $p < 0.05$ but found “no effect” if $p > 0.05$.

This book is an introduction to the statistical analysis of data from biological experiments with a focus on the estimation of treatment effects and measures of the uncertainty of these estimates. Instead of a flowchart of “which statistical test”, this book emphasizes a **regression modeling** approach using linear models and extensions of linear models.

“What what? In my previous class I learned that regression was for data with a continuous independent variable and that t -tests and ANOVA were for data with categorical independent variables.” No! This misconception has roots in the history of regression vs. ANOVA and is reinforced by how introductory biostatistics textbooks, and their instructors, *choose* to teach statistics. In this class, you were probably taught to follow a flowchart strategy – something like

Compared to the flowchart strategy, the advantages of the regression modeling strategy include

1. A unified approach in place of a collection of seemingly unrelated tests. The unified approach is the *regression model*.

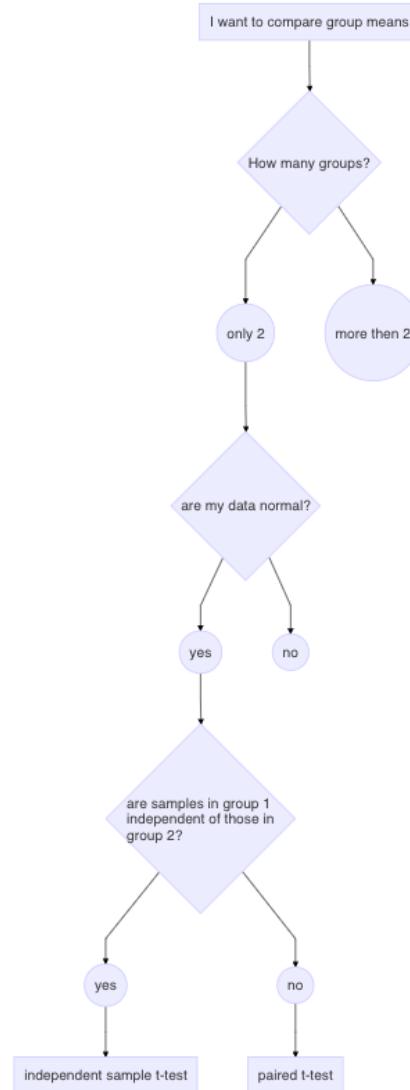


Figure 1: A small flow chart demo of the flowchart strategy of statistical analysis. This chart covers a very small subset of potential paths that could be built from an introductory biostatistics textbook.

It has long been appreciated that classical regression, *t*-tests, ANOVA, and other methods are all variations of the equation for a line $Y = mX + b$ using slightly different notation

$$Y = \beta_0 + \beta_1 X + \varepsilon \quad (1)$$

Chapter 1 explains the meaning of this notation but the point to make here is that because all regression models are variations of this equation, a modeling strategy of learning or doing statistics is more coherent than a flowchart strategy. Generalizations of this basic equation include general linear models, linear mixed models, generalized linear models, generalized additive models, causal graphical models, multivariate models, and machine learning. This book is not a comprehensive source for any of these methods but an introduction to the common foundations of all these methods.

2. Estimates of effects and uncertainty are, ultimately, *far* more useful than *p*-values. For example, to build useful models on the effects of an increasingly acidified ocean on coral growth, we want to estimate the *direction* and *magnitude* of the effects at different levels of acidification and how these estimates change under different conditions. We can compare the magnitude to a prediction of the magnitude from a mechanistic model of growth. We can use a magnitude and uncertainty to make predictions about the future of coral reefs, under different scenarios of ocean acidification. We can use the estimated effects and uncertainty to model the consequences of the effects of acidification on coral growth on fish production or carbon cycling.

By contrast, researchers learn little from a hypothesis test – that is, comparing *p* to 0.05. A *p*-value is a measure of compatibility between the data and the null hypothesis and, consequently, a pretty good – but imperfect – tool to dampen the frequency that we are fooled by randomness. Importantly, a *p*-value **is not a measure of the size of an effect**. At most, a small *p*-value gives a researcher some confidence in the existence and direction of an effect. But if we are investigating the effects of acidified ocean water on coral growth, it would be absurd to conclude from a *p*-value that pH does or does not affect growth. pH affects *everything* about cell biology.

p-values are neither necessary nor sufficient for good data analysis. Properly understood, a *p*-value is a useful tool in the data analysis toolkit. As stated above, the proper use of *p*-values dampens the frequency that we are fooled by randomness. Importantly, the estimation of effects and uncertainty and the computation of a *p*-value are not alternatives. Indeed, the *p*-value returned by many hypothesis tests are computed from the regression model used to estimate the effects. Throughout this text, statistical models are used to compute a *p*-value *in addition to* the estimates of effects and their uncertainty.

NHST Blues – The “discovery by p-value” strategy, or Null-Hypothesis Significance Testing (NHST), has been criticized by statisticians for many, many decades. Nevertheless, introductory biostatistics textbooks written by both biologists and statisticians continue to organize textbooks around a collection of hypothesis tests, with much less emphasis on estimation and uncertainty. The NHST strategy of learning or doing statistics is easy in that it requires little understanding of the statistical model underneath the tests and its assumptions, limitations, and behavior. The NHST strategy in combination with point-and-click software enables “mindless statistics”¹ and encourages the belief that statistics is a tool like a word processor is a tool, afterall, a rigorous analysis of one’s data requires little more than getting p-values and creating bar plots. Indeed, many PhD programs in the biosciences require no statistics coursework and the only training available to students is from the other graduate students and postdocs in the lab. As a consequence, the biological sciences literature is filled with error bars that imply data with negative values and p-values that have little relationship to the probability of the data under the null. More importantly for science, the reported statistics are often not doing for the study what the researchers and journal editors think they are doing.

0.1 Math

0.2 R and programming

¹Gegenrezer

Part I: Getting Started

Chapter 1

Getting Started – R Projects and R Markdown

A typical statistical modeling project will consist of:

1. importing data from Excel or text (.csv or .txt) files
2. cleaning data
3. initial exploratory plots
4. analysis
5. model checking
6. generating plots
7. generating tables
8. writing text to describe the project, the methods, the analysis, and the interpretation of the results (plots and tables)

The best practice for reproducible research is to have all of these steps in a single document and all of the files for this project in a single folder (directory), preferably on a cloud drive. Too many research projects are not reproducible because the data were cleaned in Excel, and then different parts of the data were separately imported into a GUI statistics software for analysis, and then output from the statistics software was transcribed to Excel to make a table. And other parts of the analysis are used to create a plot in some plotting software. And then the tables and plots are pasted into Microsoft Word to create a report. Any change at any step in this process will require the researcher to remember all the downstream parts that are dependent on the change and to re-do an analysis, or a table, or a plot, etc. etc.

R studio encourages best practices by creating a **project folder** that contains all project documents and implementing a version of markdown called R Markdown. An R Markdown document can explicitly link all parts of the workflow

so that changes in earlier steps automatically flow into the later steps. At the completion of a project, a researcher can choose “run all” from the menu and the data are read, cleaned, analyzed, plotted, tabulated, and put into a report with the text.

1.1 R vs R Studio

R is a programming language. It runs under the hood. You never see it. To use R, you need another piece of software that provides a **user interface**. The software we will use for this is R Studio. R Studio is a slick (very slick) **graphical user interface** (GUI) for developing R projects.

1.2 Download and install R and R studio

[Download R for your OS](#)

[Download R Studio Desktop](#)

If you need help installing R and R studio, here is Andy Field’s [Installing R and RStudio video tutorial](#))

1.3 Install R Markdown

In this text, we will write code to analyze data using R Markdown. R markdown is a version of Markdown. Markdown is tool for creating a document containing text (like Microsoft Word), images, tables, and code that can be output to the three modern output formats: html (web pages), pdf (reports and documents), and Microsoft Word (okay, this isn’t modern but it is widely used).

[Directions for installing R Markdown](#)

R Markdown can output pdf files. The mechanism for this is to first create a LaTeX (“la-tek”) file. LaTeX is an amazing tool for creating professional pdf documents. You do not need PDF output for this text, but I encourage you to download and install the [tinytex distribution](#), which was created especially for R Markdown in R Studio.

The [tinytex distribution](#) is here.

1.4 Importing Packages

The R scripts you write will include functions in packages that are not included in Base R. These packages need to be downloaded from an internet server to

your computer. You only need to do this once (although you have to redo it each time you update R). But, each time you start a new R session, you will need to load a package using the `library()` function. Now is a good time to import packages that we will use.

Open R Studio and choose the menu item “Tools” > “Install Packages”. In the “packages” input box, insert the names of packages to install the package. The names can be separated by spaces or commas, for example “`data.table`, `emmeans`, `ggplot2`”. Make sure that “install dependencies” is clicked before you click “Install”. Packages that we will use in this book are

1. Import and analysis packages

- `devtools` – we use this to install packages that are not on CRAN
- `here` – we use to read from and write to the correct folder
- `janitor` – we use the function `clean_names` from this package
- `readxl` – elegant importing from microsoft Excel spreadsheets
- `data.table` - improves functionality of data frames

2. analysis packages

- `nlme` – we use this for `gls` models
- `lme4` – we use this for linear mixed models
- `lmerTest` – we use this for inference with linear mixed models
- `glmmTMB` – we use this for generalized linear models
- `MASS` – we will use `glm.nb` from this package
- `afex` – we use this for classic ANOVA
- `emmeans` – we use this to compute modeled means and contrasts

3. graphing packages

- `ggplot2` – we use this for plotting
- `ggsci` – we use this for the color palettes
- `ggpubr` – we use this to make ggplots a bit easier
- `ggforce` – we use this for improved jitter plots
- `dabestr` – we use this to make several plot types
- `cowplot` – we use this to combine plots
- `ggpubfigs` – we use this for the color palettes

`ggpubfigs` is new and has not been uploaded to the R CRAN library. To install `ggpubfigs`, copy and paste this into the console:

```
devtools::install_github("JLSteenwyk/ggpubfigs")
```

Once these are installed, you don’t need to do this again although there will be additional packages that you might install. You simply need to use the `library()` function at the start of a markdown script.

1.5 Create an R Studio Project for this textbook

1. Create a project folder within the Documents folder (Mac OS) or My Documents folder (Windows OS). All files associated with this book will reside inside this folder. The name of the project folder should be something meaningful, such as “Applied_Biostatics” or the name of your class (for students in my Applied Biostatics class, this folder could be named “BIO_413”).
2. Within the project folder, create new folders named
 1. “Rmd” – this is where your R markdown files are stored
 2. “R” – this is where additional R script files are stored
 3. “data” – this is where data that we download from public archives are stored
 4. “output” – this is where you will store fake data generated in this class
 5. “images” – this is where image files are stored
3. Open R Studio and click the menu item File > New Project...
4. Choose “Existing Directory” and navigate to your project folder
5. Choose “Create Project”
6. Check that a “.Rproj” file is in your project folder

1.5.1 Create an R Markdown file for this Chapter

1. The top-left icon in R Studio is a little plus sign within a green circle. Click this and choose “R Markdown” from the pull-down menu.
2. Give the file a meaningful title like “Chapter 1 – Organization”
3. Delete all text below the first code chunk, starting with the header “## R Markdown”

1.5.1.1 Modify the yaml header

Replace “output: html_document” in the yaml header with the following in order to creat a table of content (toc) on the left side of the page and to enable code folding

```
output:
  html_document:
    toc: true
    toc_float: true
    code_folding: hide
```

1.5.1.2 Modify the “setup” chunk

The setup chunk should look something like this

```
knitr:::opts_chunk$set(echo = TRUE)

# wrangling packages
library(here)
library(janitor)
library(readxl)
library(data.table)

# analysis packages
library(MASS) # negative binomial and some other functions
library(nlme) # gls and some lmm

# graphing packages
library(ggsci) # color palettes
library(ggpubr) # publication quality plots
library(ggforce) # better jitter
library(cowplot) # combine plots
library(ggpubfigs) # color palettes
# ggpubfigs is not in CRAN. To install run the next line.
# devtools::install_github("JLSteenwyk/ggpubfigs")

# Okabe & Ito palette
ito_seven <- friendly_pal("ito_seven") # ggpubfigs
pal_okabe_ito <- ito_seven[c(6,5,3,7,1,2,4)] # order of Wilke

here <- here::here
data_path <- "data"
```

1.5.2 Create a “fake-data” chunk

- Let's play around with an R Markdown file. Create a new chunk and label it “fake-data”. Insert the following R script and then click the chunk's run button

```
set.seed(4)
n <- 10
fake_data <- data.table(
  treatment = rep(c("cn", "tr"), each = n),
  neutrophil_count_exp1 = rnegbin(n*2,
    mu = rep(c(10, 15), each = n),
```

```

            theta = 1),
neutrophil_count_exp2 = rnegbin(n*2,
                                 mu = rep(c(10, 20), each = n),
                                 theta = 1)
)
# View(fake_data)

```

This chunk creates fake neutrophil counts in two different experiments. The comment (#) sign before `View(fake_data)` “comments out” the line of code, so it is not run. View the data by highlighting `View(fake_data)` and choosing “Run selected line(s)” from the Run menu.

1.5.3 Create a “plot” chunk

5. Create a new chunk and label it “plot”. Insert the following R script and then click the chunk’s run button

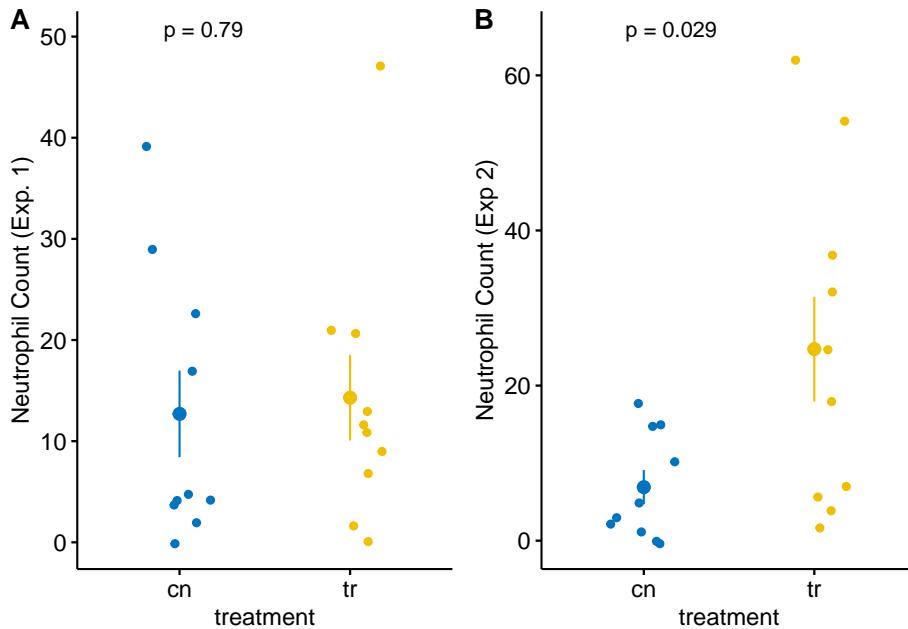
```

gg_1 <- ggstripchart(data = fake_data,
                      x = "treatment",
                      y = "neutrophil_count_exp1",
                      color = "treatment",
                      palette = "jco",
                      add = "mean_se",
                      legend = "none") +
  ylab("Neutrophil Count (Exp. 1)") +
  stat_compare_means(method = "t.test",
                     label.y = 50,
                     label = "p.format") +
  NULL

gg_2<- ggstripchart(data = fake_data,
                      x = "treatment",
                      y = "neutrophil_count_exp2",
                      color = "treatment",
                      palette = "jco",
                      add = "mean_se",
                      legend = "none") +
  ylab("Neutrophil Count (Exp 2)") +
  stat_compare_means(method = "t.test",
                     label.y = 65,
                     label = "p.format") +
  NULL

plot_grid(gg_1, gg_2, labels = "AUTO")

```



Each plot shows the mean count for each group, the standard error of the mean count, and the p -value from a t -test. This statistical analysis and plot are typical of those found in experimental biology journals. This text will teach alternatives that implement better practices.

1.5.4 Knit

6. Knit to an html file
7. Knit to a pdf file, if you've installed tinytex (or some other LaTeX distribution)
8. Knit to a word document

Part II: An introduction to the analysis of experimental data with a linear model

Chapter 2

Analyzing experimental data with a linear model

2.1 This text is about the estimation of treatment effects and the uncertainty in our estimates using linear models. This, raises the question, what is “an effect”?

This text has an unusual start – an example analysis. This example is a goal or target; it’s what you will be working towards as you learn from this text. The data for the analysis come from multiple experiments presented in Figure 2 in the article ASK1 inhibits browning of white adipose tissue in obesity. The analysis is in the last part of this chapter. The second part of this chapter is just enough biology to help you understand the biological importance of each experiment. The first part of this chapter uses one experiment from Figure 2 to outline what the statistical analysis of experimental data is all about. Much of this outline will be repeated in “An introduction to linear models” chapter.

The analysis in part 3 of this chapter is a set of experiments exploring the consequences of adipose-tissue specific deletion of the ASK1 signaling protein on multiple, adverse effects of a high-fat diet in mice, including weight gain, glucose intolerance, and increased liver triacylglycerol levels. Think of this as a template for organizing your own R Markdown documents. This document is a re-analysis of the experiments in Figure 2 in the article ASK1 inhibits browning of white adipose tissue in obesity, including generation of the publication-ready plots. I chose the data in Figure 2 of this paper because of the diversity of analyses and plot types. My analyses and plots differ slightly from those of the researchers because I implemented better practices – the stuff of this text.

The goal of the experiments is to measure the **effect** of the adipose-specific ASK1 deletion. To understand what I mean by “an effect”, and to understand how we can estimate an effect by **fitting a linear model** to data, let’s look more closely at the analysis for Figure 2i.

For Figure 2i, the researchers want to know if “knocking out” the ASK1 gene in the adipose tissue cells lowers the liver triglyceride (TG) level in mice fed a high-fat diet. That is, is $\bar{y}_{ASK1adipo} < \bar{y}_{ASK1F/F}$, where $\bar{y}_{ASK1adipo}$ is the mean liver TG level of the knockout (Δ is the del operator and refers to a deletion in genetics) mice and $\bar{y}_{ASK1F/F}$ is the mean liver TG level of the control mice. The difference in the means, $\bar{y}_{ASK1adipo} - \bar{y}_{ASK1F/F}$, is **the effect** (of ASK1 deletion on liver TG levels).

The measured means in each group are computed from a random **sample** of mice. If we only cared about the six mice in each group in this experiment, then we would not need to fit a linear model to the data to estimate the effect, we could simply compute each group mean and subtract the control mean from the knockout mean. But we care more about these dozen mice because we are trying to discover something general about ASK1 regulation of TG levels in mice, generally (and even in mammals, and especially humans, generally). To make this leap of **inference**, we use a model to claim that each sample mean is an **estimate** of the respective **population** mean. Given this model, we can compute the **standard error** of each mean and the **standard error of the difference in means**. A standard error is a measure of the **sampling variance of a statistic** and, therefore, a measure of the precision of the estimate. The standard error, then, is a measure of **uncertainty** in the estimate. Here is how to think about precision and uncertainty: if we were to repeat this experiment many, many times, we would generate a long list of mean TG levels for the control mice and a long list of mean TG levels for the knockout mice. The less variable the means in each list, the more precise. By using a model, we do not need to repeat this experiment many times to get a standard error.

The model we are going to fit to the Figure 2i data is

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i \quad (2.1)$$

$$\varepsilon_i \sim N(0, \sigma^2) \quad (2.2)$$

This is a model of how the Figure 2i data were generated. In this model, y_i is the liver TG level for some fictional (generated!) mouse (the i stands for the i th fictional mouse generated) and x_i is a variable that indicates the condition of the ask1 gene in fictional mouse i . For x_i , a value of 0 is given to mice with a functional ASK1 gene and a value of 1 is given to mice with a knocked out gene.

β_0 is the “true” mean of the TG level in mice fed a high-fat diet and with a functional ASK1 gene. By “true”, I mean the mean that would be computed if

2.1. THIS TEXT IS ABOUT THE ESTIMATION OF TREATMENT EFFECTS AND THE UNCERTAINTY IN OUR

we were to measure TG on an infinite number of these mice. The observed mean of the ASK1F/F group is an estimate of β_0 . The sum $\beta_0 + \beta_1$ is the true mean of the TG level in mice fed a high-fat diet but with a knocked out ASK1 gene. This means that β_1 is the true difference in the means, or the **true effect**. The observed difference in means between the ASK1 Δ adipo and ASK1F/F groups is an estimate of β_1 . This difference is the estimated effect.

Notice that the sum $\beta_0 + \beta_1 x_i$ equals the true mean of the infinite set of normal mice if $x_i = 0$ and equals the true mean of the infinite set of ASK1 knockout mice if $x_i = 1$. ε_i is the **error** for mouse i , which is the difference between the TG level for mouse i and the **expected** TG value for mouse i . The expected value for a mouse with a normal ASK1 gene is β_0 . The expected value for a mouse with a knocked out ASK1 gene is $\beta_0 + \beta_1$. The second line of the model simply states that ε_i is modeled as a random sample from a normal distribution with a mean of zero and a variance of σ^2 .

By **fitting a model to the data** we estimate the parameters β_0 , β_1 and σ . It is the estimation of σ that allows us to compute a measure of our uncertainty (a standard error) of our estimates of the means (β_0 and $\beta_0 + \beta_1$) and of the difference in the means (β_1).

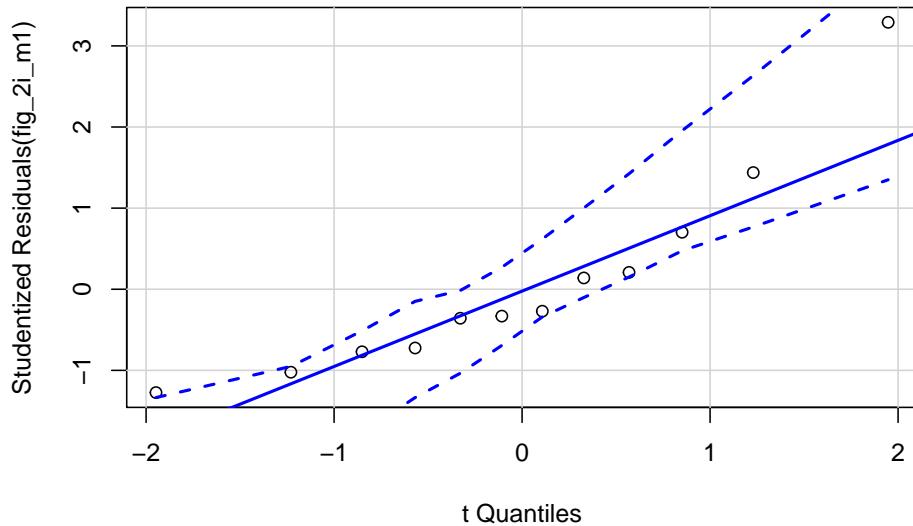
Let's fit this model to the Figure 2i data using R

```
fig_2i_m1 <- lm(liver_tg ~ treatment, data = fig_2i)
```

Robust inference from the model (generalizing from sample to population, including measures of the uncertainty of our estimates) requires that our data approximates the kind of data we'd expect from the data generating model specified above. All rigorous analysis should use specific **model checks** to evaluate this. First, the “normality check” – we use a **quantile-quantile (QQ) plot** to see if our data approximate what we'd see if we sampled from a normal distribution. This looks okay, in the sense that the observed data points (open circles) fall within the boundaries set by the dashed line. Inference is pretty robust to moderate departure from normal.

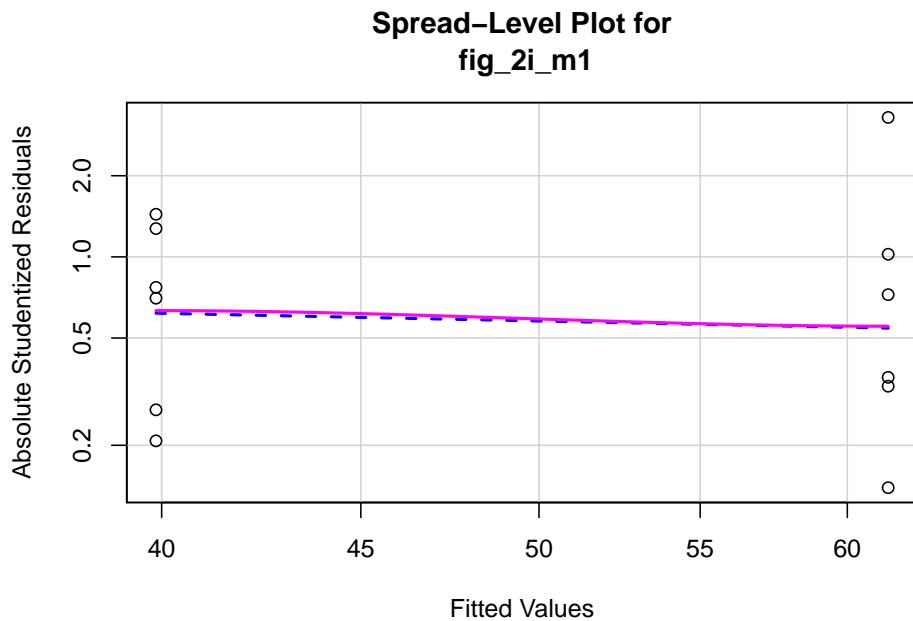
```
set.seed(1)
qqPlot(fig_2i_m1, id=FALSE)
```

30CHAPTER 2. ANALYZING EXPERIMENTAL DATA WITH A LINEAR MODEL



Second, the “homogeneity check” – we use a **spread level plot** to see if there is some pattern to the variance, for example if the spread of residuals is noticeably bigger in one group than another, or if the spread increases with the fitted value. This looks pretty good. Given these checks, lets move on and look at the table of model coefficients

```
spreadLevelPlot(fig_2i_m1, id=FALSE)
```



2.1. THIS TEXT IS ABOUT THE ESTIMATION OF TREATMENT EFFECTS AND THE UNCERTAINTY IN OUR

```
##  
## Suggested power transformation: 1.294553  
  
fig_2i_m1 <- lm(liver_tg ~ treatment, data = fig_2i)  
fig_2i_m1_coef <- cbind(coef(summary(fig_2i_m1)),  
                           confint(fig_2i_m1))  
knitr::kable(fig_2i_m1_coef, digits = c(1, 2, 1, 3, 1, 1))
```

Estimate
Std. Error
t value
Pr(> t)
2.5 %
97.5 %
(Intercept)
61.5
4.98
12.3
0.000
50.4
72.6
treatmentASK1Δadipo
-21.6
7.05
-3.1
0.012
-37.3
-5.9

The two values in the column “Estimate” are the estimates of β_0 and β_1 . The top value (61.5) is the mean of the control mice (the units are $\mu\text{mol/g}$). The mean of the knockout mice is the sum of the two values (39.9 $\mu\text{mol/g}$). And the effect of ASK1 deletion on TG levels is simply the second value (-21.6 $\mu\text{mol/g}$). The standard error of the effect is 7.05 $\mu\text{mol/g}$. We can use the standard error to compute a *t*-value (-3.1, in the column “t value”). A t-value is a **test statistic**. The probability (“p value”) of the significance test is 0.012. This if the

probability of sampling a t -value as large or larger than the observed t -value, if we were to sample from a null distribution of t -values (a distribution of sampled t values if the true value of β_1 was 0). We can also use the standard error to compute a **95% confidence interval of the effect**. The lower bound of this interval is -37.3 $\mu\text{mol/g}$ and the upper bound is -5.9 $\mu\text{mol/g}$. A confidence interval is another way of communicating uncertainty, and the way advocated in this text. In a 95% confidence interval, 95% of similarly constructed intervals (from hypothetical sampling of six mice from the ASK1 normal population and six mice from the ASK1 knockout population) will contain the true mean. Another way to think about a confidence interval is, it is the range of true differences that are compatible with the data, where compatible means “not rejected” by a t -test (a t -test between the estimated effect and any number inside the interval would return a p -value greater than 0.05).

Here is how we might report this result in a paper:

Mean TG level in ASK1 Δ adipo mice on a high-fat diet was 21.6 $\mu\text{mol/g}$ less than that in ASK1F/F mice on a high-fat diet (95% CI: -37.3, -5.9, $p = 0.012$).

An a plot for the paper:

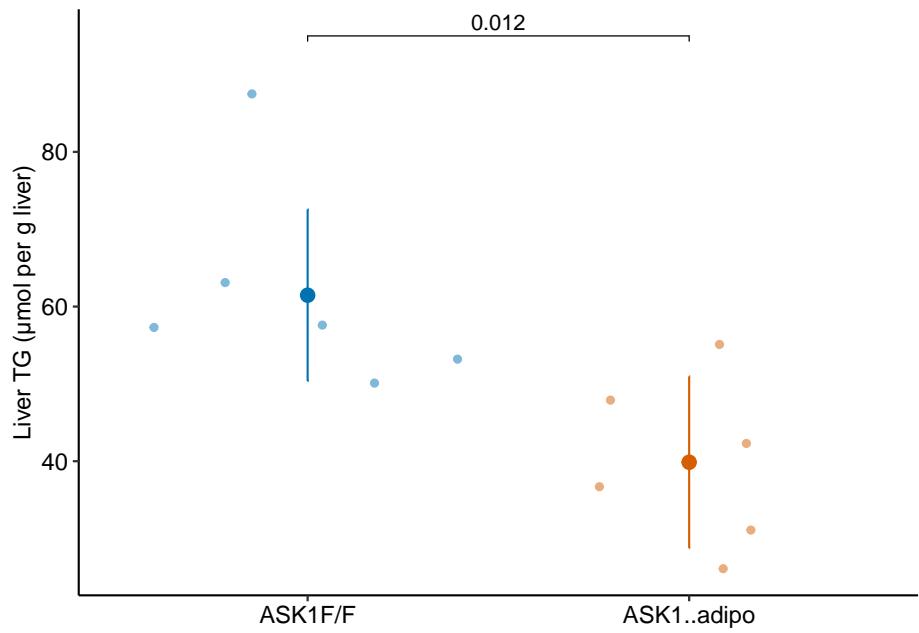
```
fig_2i_m1_emm <- emmeans(fig_2i_m1, specs = "treatment")
fig_2i_m1_pairs <- contrast(fig_2i_m1_emm,
                             method = "revpairwise") %>%
  summary(infer = TRUE)
fig_2i_m1_emm_dt <- summary(fig_2i_m1_emm) %>%
  data.table
fig_2i_m1_pairs_dt <- data.table(fig_2i_m1_pairs)
fig_2i_m1_pairs_dt[, p.pretty := pvalString(p.value)]
fig_2i_m1_pairs_dt[, group1 := 1]
fig_2i_m1_pairs_dt[, group2 := 2]

fig_2i_gg <- ggplot(data = fig_2i,
                     aes(x = treatment,
                         y = liver_tg,
                         color = treatment)) +
  # points
  geom_sina(alpha = 0.5) +
  # plot means and CI
  geom_errorbar(data = fig_2i_m1_emm_dt,
                aes(y = emmean,
                    ymin = lower.CL,
                    ymax = upper.CL,
                    color = treatment),
```

2.1. THIS TEXT IS ABOUT THE ESTIMATION OF TREATMENT EFFECTS AND THE UNCERTAINTY IN OUR

```
width = 0
) +  
  
geom_point(data = fig_2i_m1_emm_dt,
            aes(y = emmean,
                color = treatment),
            size = 3
) +  
  
# plot p-values (y positions are adjusted by eye)
stat_pvalue_manual(fig_2i_m1_pairs_dt,
                    label = "p.pretty",
                    y.position=c(95),
                    tip.length = 0.01) +  
  
# aesthetics
ylab("Liver TG ( $\mu$ mol per g liver)") +
scale_color_manual(values=pal_okabe_ito[5:6],
                    name = NULL) +
theme_pubr() +
theme(legend.position="none") +
theme(axis.title.x=element_blank()) +  
  
NULL  
  
fig_2i_gg
```

34 CHAPTER 2. ANALYZING EXPERIMENTAL DATA WITH A LINEAR MODEL



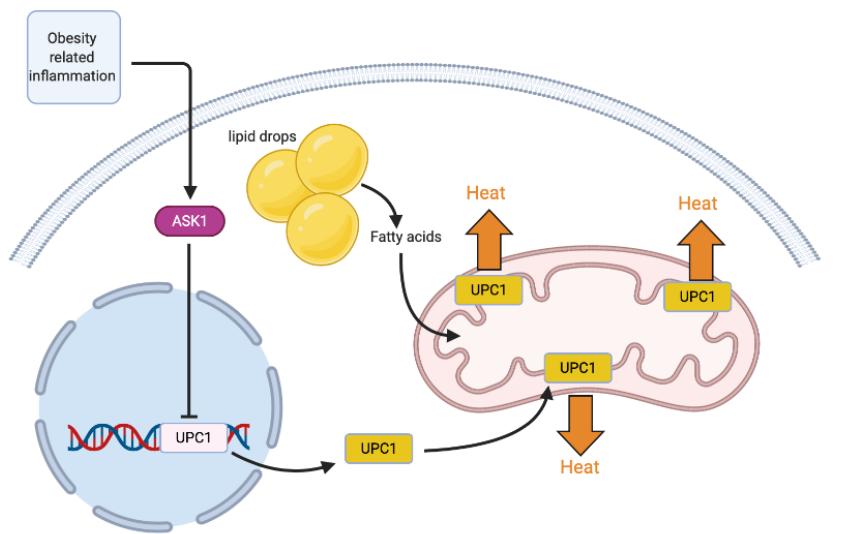
Background physiology to the experiments in Figure 2 of “ASK1 inhibits browning of white adipose tissue in obesity”

A little background on the subject of the article: white adipose tissue (WAT) is composed of adipose (fat) cells that function as energy storage cells. The energy is in the form of the fatty acids in the triacylglycerols, which form large lipid drops in the cell. The stored fatty acids are released from the WAT when other organs need energy. Mammalian brown adipose tissue (BAT) is composed of adipose cells that burn the stored fat to generate heat. This is enabled by the expression of the protein *uncoupling receptor 1* (UCP1) in the mitochondria. UCP1 uncouples the proton gradient across the inner mitochondrial membrane from ATP synthesis.

In response to adverse health consequences of obesity, including metabolic syndrome, researchers are investigating various ways to increase BAT, or stimulate BAT activity, or transform WAT cells into more BAT-like cells, by turning up expression of UCP1. The regulation of UCP1 in WAT is a potential drug target for obesity.

The researchers of the ASK1 study investigated the effects of an intracellular signaling protein (ASK1) on the browning of white adipose tissue. Previous research had suggested that 1) inflammation stimulates ASK1 activity and 2) increased ASK1 activity inhibits UCP1 expression (Figure 2.1). The experiments in Figure 2 of the ASK1 study follow this up and explore the question, if ASK1 is knocked out in the WAT cells, will this reverse the adverse effects of a high-fat diet, including weight gain, glucose intolerance, and liver triacylglycerol levels?

For the experiments in Figure 2, the researchers created mice in which the ASK1



Created in BioRender.com

Figure 2.1: Inflammation to obesity stimulates ASK1 activity. ASK1 activity inhibits UCP1 expression.

2.1. THIS TEXT IS ABOUT THE ESTIMATION OF TREATMENT EFFECTS AND THE UNCERTAINTY IN OUT

gene was inhibited from being expressed (or “knocked out”) in the white adipose tissue cells. The *ask1* treatment has two levels: “ASK1 Δ adipo”, which are the adipocyte-specific ASK1 knockout (KO) mice, and “ASK1F/F”, which are the controls. For some of the experiments, the researchers split the mice in each *ask1* treatment level and assigned these to either a Chow or a High Fat Diet (HFD). This experimental design is two-crossed factors, each with two levels, which I call a 2×2 factorial design in this text.

- Some of the plots are coded directly in this document. Others use functions from the chapter “Plotting functions”. But, to use these in an R Markdown document, these functions have to be saved in a “R Script” file. This script file then needs to be read at the start of the R Markdown document. I named the script file “ggplotsci.R” and placed it in a folder called “R” at the level of the project (directly within the project folder).
- This example was written with the Bookdown style sheet (because its part of this book), which doesn’t have one nice features of creating R Markdown documents for reports and manuscripts – code folding. In an R Markdown document with code folding, a user can toggle between showing and hiding code. The html output with code folding is here.

Analyses for Figure 2 of “ASK1 inhibits browning of white adipose tissue in obesity”

2.2 Setup

Some plots in this document require the file “ggplotsci.R” within the “R” folder, within the project folder.

```
knitr::opts_chunk$set(echo = TRUE, warning=FALSE, message=FALSE)

# wrangling packages
library(here)
library(janitor)
library(readxl)
library(data.table)
library(stringr)

# analysis packages
library(emmeans)
library(car) # qqplot, spreadlevel
library(DHARMa)

# graphing packages
library(ggsci)
library(ggpubr)
library(ggforce)
library(cowplot)
library(lazyWeave) #pvalstring
```

```
here <- here::here
data_path <- "data"

ggplotsci_path <- here("R", "ggplotsci.R")
source(ggplotsci_path)
```

2.3 Data source

Data source: ASK1 inhibits browning of white adipose tissue in obesity

This chunk assigns the path to the Excel data file for all panels of Figure 2. The data for each panel are in a single sheet in the Excel file named “Source Date_Figure 2”.

```
data_folder <- "ASK1 inhibits browning of white adipose tissue in obesity"
file_name <- "41467_2020_15483_MOESM4_ESM.xlsx"
file_path <- here(data_path, data_folder, file_name)

fig_2_sheet <- "Source Date_Figure 2"
```

2.4 control the color palette

```
fig_2_palette <- pal_okabe_ito[5:6]
# fig_2_palette <- pal_okabe_ito
#fig_2_palette <- pal_nature_mod
```

2.5 useful functions

A function to import longitudinal data from Fig 2

```
# function to read in parts of 2b
import_fig_2_part <- function(range_2){
  fig_2_part <- read_excel(file_path,
                           sheet = fig_2_sheet,
                           range = range_2,
                           col_names = TRUE) %>%
    data.table()
  group <- colnames(fig_2_part)[1]
```

```

setnames(fig_2_part, old = group, new = "treatment")
fig_2_part[, treatment := as.character(treatment)] # this was read as logical
fig_2_part[, treatment := group] # assign treatment group
fig_2_part[, mouse_id := paste(group, .I)]
return(fig_2_part)
}

# script to compute various area under the curves (AUC) using trapezoidal method
# le Floch's "incremental" auc subtracts the baseline value from all points.
# This can create some elements with negative area if post-baseline values are less
# than baseline value.
# Some researchers "correct" this by setting any(y - ybar < 0 to zero. Don't do this.

auc <- function(x, y, method="auc", average = FALSE){
  # method = "auc", auc computed using trapezoidal calc
  # method = "iauc" is an incremental AUC of Le Floch
  # method = "pos_iauc" is a "positive" incremental AUC of Le Floch but not Wolever
  # method = "post_0_auc" is AUC of post-time0 values
  # if average then divide area by duration
  if(method=="iauc"){y <- y - y[1]}
  if(method=="pos_iauc"){y[y < 0] <- 0}
  if(method=="post_0_auc"){
    x <- x[-1]
    y <- y[-1]
  }
  n <- length(x)
  area <- 0
  for(i in 2:n){
    area <- area + (x[i] - x[i-1])*(y[i-1] + y[i])
  }
  value <- area/2
  if(average == TRUE){
    value <- value/(x[length(x)] - x[1])
  }
  return(value)
}

```

2.6 figure 2b – effect of ASK1 deletion on growth (body weight)

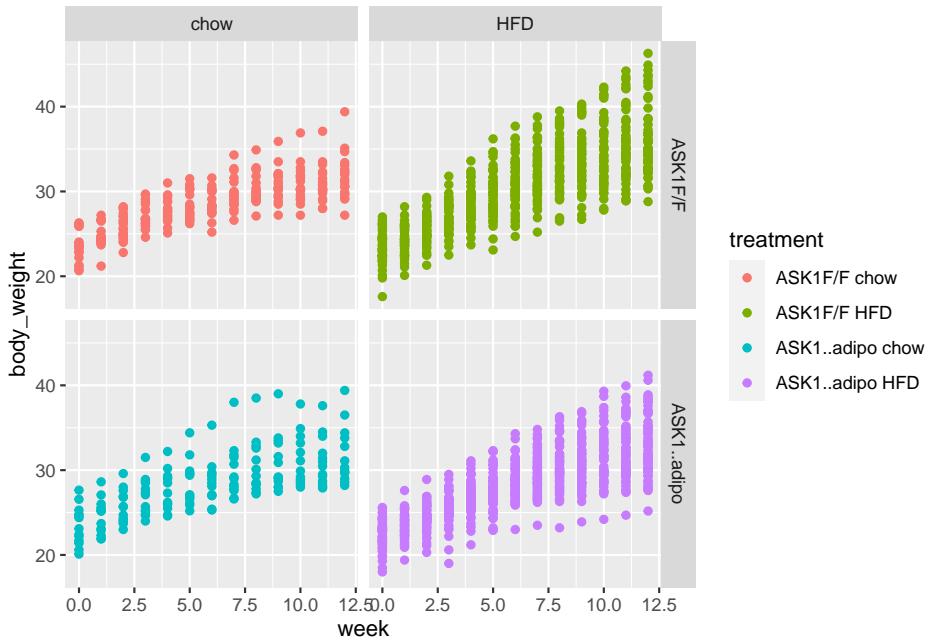
2.6.1 figure 2b – import

```
range_list <- c("A21:N41", "A43:N56", "A58:N110", "A112:N170")
fig_2b_wide <- data.table(NULL)
for(range_i in range_list){
  part <- import_fig_2_part(range_i)
  fig_2b_wide <- rbind(fig_2b_wide,
                        part)
}
fig_2b <- melt(fig_2b_wide,
               id.vars = c("treatment", "mouse_id"),
               variable.name = "week",
               value.name = "body_weight")
fig_2b[, week := as.numeric(as.character(week))]
fig_2b[, c("ask1", "diet") := tstrsplit(treatment, " ", fixed=TRUE)]
fig_2b[, week_f := factor(week)]
```

2.6.2 figure 2b – exploratory plots

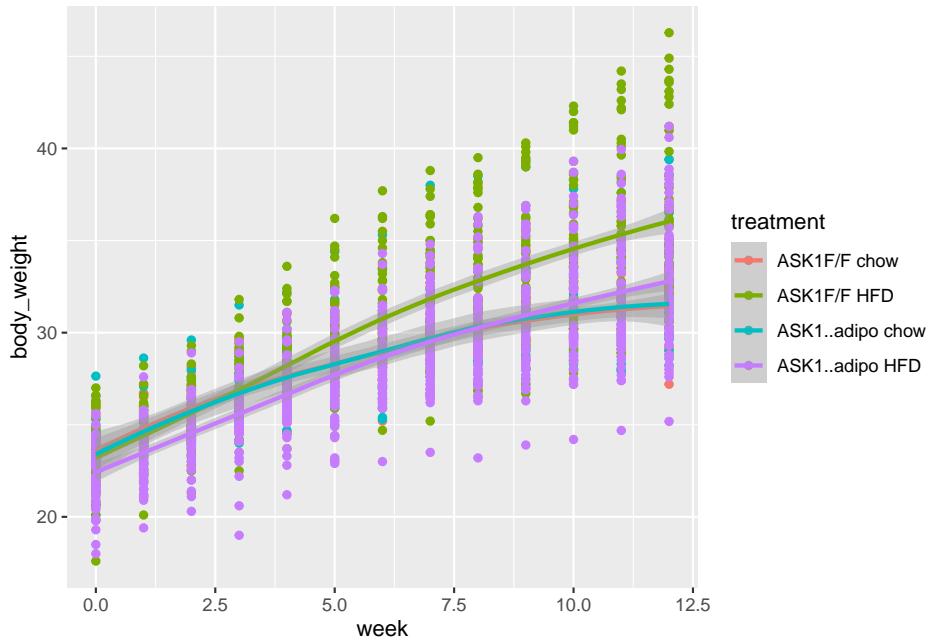
```
qplot(x = week,
       y = body_weight,
       data = fig_2b,
       color = treatment) +
facet_grid(ask1~diet)
```

2.6. FIGURE 2B – EFFECT OF ASK1 DELETION ON GROWTH (BODY WEIGHT)43



1. no obvious outliers
2. reduced growth rate at bigger size

```
qplot(x = week,
      y = body_weight,
      data = fig_2b,
      color = treatment) +
  geom_smooth()
```



1. loess smooth. Growth in ASK1F/F + HFD clearly greater than other three treatment combinations.

2.7 Figure 2c – Effect of ASK1 deletion on final body weight

2.7.1 Figure 2c – import

```
range_2c <- "A173:BD176"
y_cols <- c("ASK1F/F chow", "ASK1Δadipo chow", "ASK1F/F HFD", "ASK1Δadipo HFD")
fig_2c_import <- read_excel(file_path,
                           sheet = fig_2_sheet,
                           range = range_2c,
                           col_names = FALSE) %>%
  transpose(make.names=1) %>%
  data.table() %>%
  melt(measure.vars = y_cols,
       variable.name = "treatment",
       value.name = "body_weight_gain") %>%
  na.omit()
fig_2c_import[, mouse_id := paste(treatment, .I, sep = "_"),
              by = treatment]
```

2.7.2 Figure 2c – check own computation of weight change v imported value

Note that three cases are missing from fig_2c import that are in fig_2b

```
# change colnames to char
fig_2c <- copy(fig_2b_wide)
weeks <- unique(fig_2b[, week])
setnames(fig_2c,
         old = colnames(fig_2c),
         new = c("treatment", paste0("week_", weeks), "mouse_id"))
fig_2c[, weight_gain := week_12 - week_0]
fig_2c <- fig_2c[, .SD, .SDcols = c("treatment",
                                      "week_0",
                                      "week_12",
                                      "weight_gain")]
fig_2c[, mouse_id := paste(treatment, .I, sep = "_"),
       by = treatment]

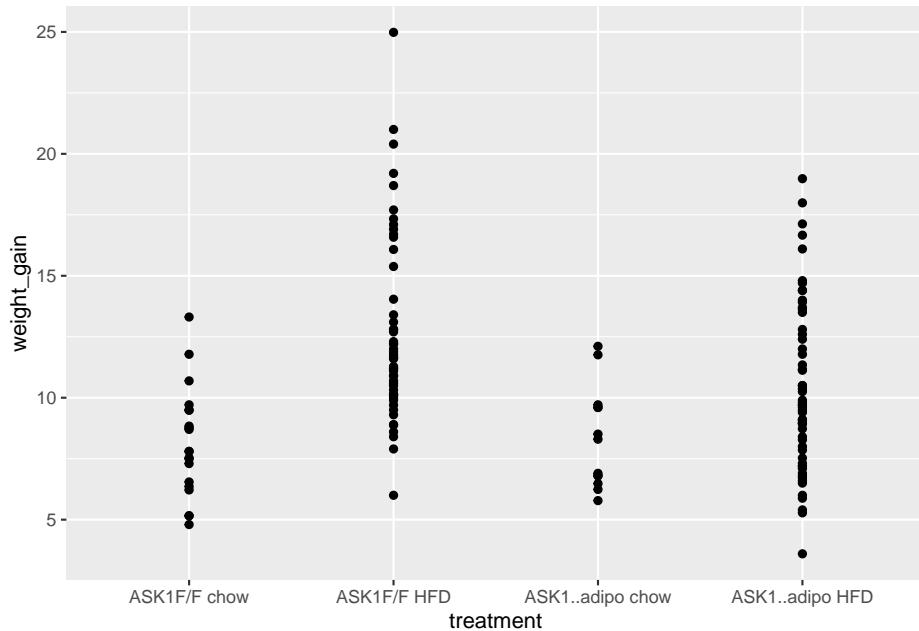
fig_2c[, c("ask1", "diet") := tstrsplit(treatment, " ", fixed=TRUE)]

fig_2c_check <- merge(fig_2c,
                      fig_2c_import,
                      by = c("mouse_id"),
                      all = TRUE)

#View(fig_2c_check)
```

2.7.3 Figure 2c – exploratory plots

```
qplot(x = treatment,
      y = weight_gain,
      data = fig_2c)
```



- no obvious outliers
- variance increases with mean, as expected from growth, suggests a multiplicative model. But start with simple lm.

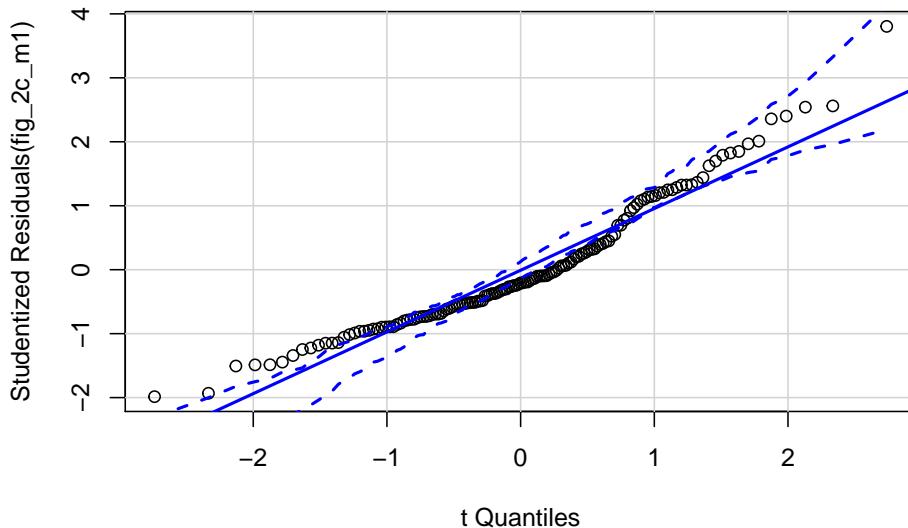
2.7.4 Figure 2c – fit the model: m1 (lm)

```
fig_2c_m1 <- lm(weight_gain ~ week_0 + ask1*diet, data = fig_2c)
```

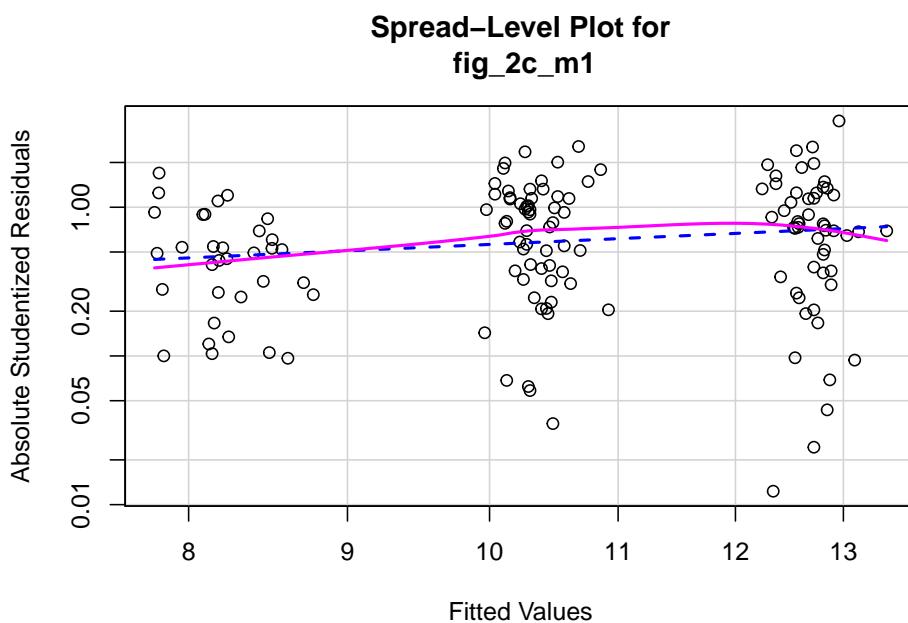
2.7.5 Figure 2c – check the model: m1

```
# check normality assumption
set.seed(1)
qqPlot(fig_2c_m1, id=FALSE)
```

2.7. FIGURE 2C – EFFECT OF ASK1 DELETION ON FINAL BODY WEIGHT 47



```
spreadLevelPlot(fig_2c_m1, id=FALSE)
```



```
##  
## Suggested power transformation: 0.06419448
```

- QQ indicates possible right skew but especially left side is squashed toward mean

- spread-level indicates variance increases with mean

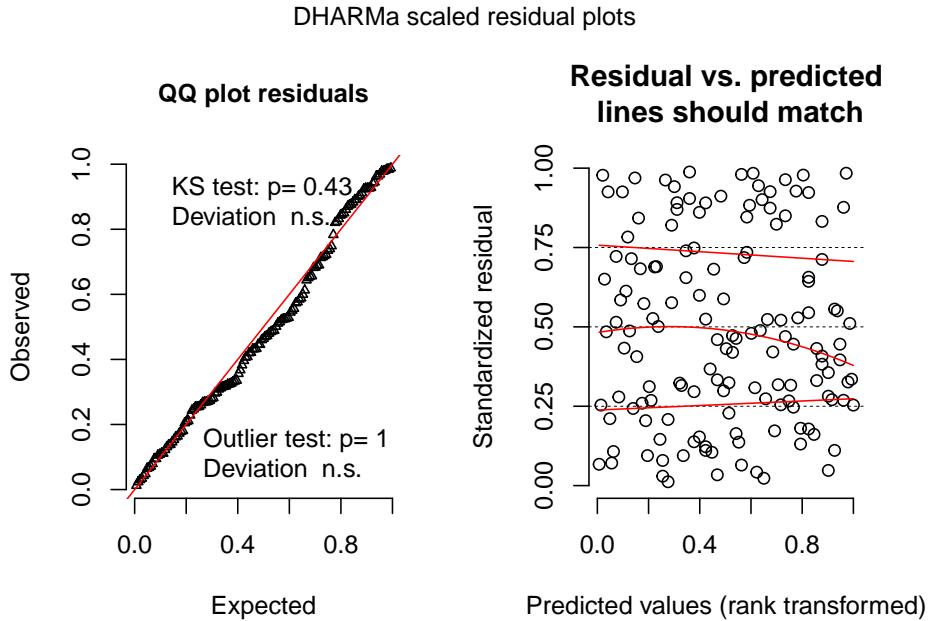
For p-value, this may not be too severe but for intervals, best to account for this. Try gamma with log link (which makes biological sense for growth)

2.7.6 Figure 2c – fit the model: m2 (gamma glm)

```
fig_2c_m2 <- glm(weight_gain ~ week_0 + ask1*diet,
                    family = Gamma(link = "log"),
                    data = fig_2c)
```

2.7.7 Figure 2c – check the model, m2

```
set.seed(1)
fig_2c_m2_sim <- simulateResiduals(fig_2c_m2, n=250)
plot(fig_2c_m2_sim)
```



- well behaved QQ and spread-level

2.7.8 Figure 2c – inference from the model

```
coef_table <- cbind(coef(summary(fig_2c_m2)),
                      exp(confint(fig_2c_m2))) %>%
  data.table(keep.rownames = TRUE)
coef_table[, Estimate:=exp(Estimate)]
knitr::kable(coef_table, digits = c(0,2,2,2,4,2,2))
```

rn	
Estimate	
Std. Error	
t value	
Pr(> t)	
2.5 %	
97.5 %	
(Intercept)	10.87
	0.35
	6.83
	0.0000
	5.53
	21.50
week_0	0.99
	0.01
	-0.84
	0.3997
	0.96
	1.02
ask1ASK1Δadipo	1.03
	0.11
	0.24

50CHAPTER 2. ANALYZING EXPERIMENTAL DATA WITH A LINEAR MODEL

```
0.8138
0.83
1.28
dietHFD
1.56
0.08
5.45
0.0000
1.33
1.82
ask1ASK1Δadipo:dietHFD
0.79
0.13
-1.93
0.0551
0.61
1.00
```

```
fig_2c_m2_emm <- emmeans(fig_2c_m2, specs = c("diet", "ask1"),
                           type = "response")
fig_2c_m2_pairs <- contrast(fig_2c_m2_emm,
                             method = "revpairwise",
                             simple = "each",
                             combine = TRUE,
                             adjust = "none") %>%
  summary(infer = TRUE)
```

fig_2c_m2_emm

	diet	ask1	response	SE	df	asymp.LCL	asymp.UCL
##	chow	ASK1F/F	8.19	0.568	Inf	7.15	9.39
##	HFD	ASK1F/F	12.75	0.548	Inf	11.72	13.87
##	chow	ASK1Δadipo	8.41	0.721	Inf	7.11	9.95
##	HFD	ASK1Δadipo	10.28	0.424	Inf	9.48	11.14
##	Confidence level used: 0.95						
##	Intervals are back-transformed from the log scale						

2.7. FIGURE 2C – EFFECT OF ASK1 DELETION ON FINAL BODY WEIGHT 51

```
fig_2c_m2_pairs
```

```
##   ask1      diet contrast      ratio      SE  df asymp.LCL asymp.UCL
##   ASK1F/F    .    HFD / chow      1.556  0.1263 Inf     1.328     1.825
##   ASK1Δadipo .    HFD / chow      1.222  0.1168 Inf     1.013     1.474
##   .          chow ASK1Δadipo / ASK1F/F 1.026  0.1127 Inf     0.828     1.273
##   .          HFD  ASK1Δadipo / ASK1F/F 0.806  0.0485 Inf     0.716     0.907
##   z.ratio p.value
##   5.453 <.0001
##   2.097 0.0360
##   0.236 0.8134
##   -3.587 0.0003
##
## Confidence level used: 0.95
## Intervals are back-transformed from the log scale
## Tests are performed on the log scale
```

- within ASK1 Cn, HFD mean is 1.6X chow mean
- within ASK1 KO, HFD mean is 1.2X chow mean
- within chow, ASK1 KO mean is 1.0X ASK1 Cn mean
- within HFD, ASK1 KO mean is 0.86X ASK1 Cn mean

```
# same as interaction effect in coefficient table
contrast(fig_2c_m2_emm, interaction = "pairwise", by = NULL) %>%
  summary(infer = TRUE)
```

```
##   diet_pairwise ask1_pairwise      ratio      SE  df asymp.LCL asymp.UCL
##   chow / HFD    ASK1F/F / ASK1Δadipo 0.785  0.0982 Inf     0.614     1
##   z.ratio p.value
##   -1.934 0.0531
##
## Confidence level used: 0.95
## Intervals are back-transformed from the log scale
## Tests are performed on the log scale
```

- the reduction in weight gain in the ASK1 KO mice compared to ASK1 CN is 0.785X. Notice that p > 0.05.

2.7.9 Figure 2c – plot the model

```

fig_2c_m2_emm_dt <- summary(fig_2c_m2_emm) %>%
  data.table
fig_2c_m2_pairs_dt <- data.table(fig_2c_m2_pairs)
fig_2c_m2_pairs_dt[, p.pretty := pvalString(p.value)]

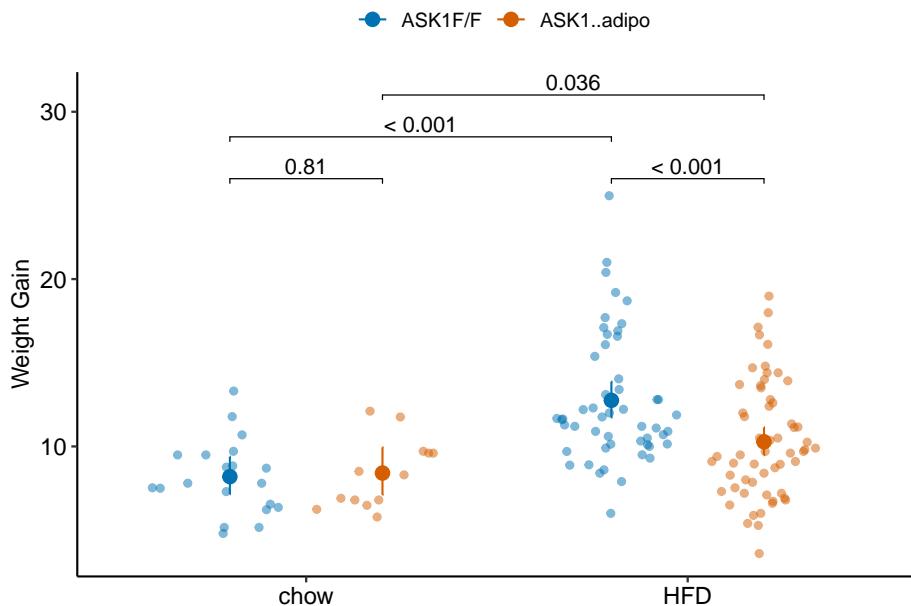
dodge_width <- 0.8 # separation between groups
# get x positions of brackets for p-values
# requires looking at table and mentally figuring out
# Chow is at x = 1 and HFD is at x = 2
fig_2c_m2_pairs_dt[, group1 := c(1-dodge_width/4,
                                1+dodge_width/4,
                                1-dodge_width/4,
                                2-dodge_width/4)]
fig_2c_m2_pairs_dt[, group2 := c(2-dodge_width/4,
                                2+dodge_width/4,
                                1+dodge_width/4,
                                2+dodge_width/4)]

pd <- position_dodge(width = dodge_width)
fig_2c_gg <- ggplot(data = fig_2c,
                      aes(x = diet,
                           y = weight_gain,
                           color = ask1)) +
  # points
  geom_sina(alpha = 0.5,
            position = pd) +
  # plot means and CI
  geom_errorbar(data = fig_2c_m2_emm_dt,
                aes(y = response,
                    ymin = asympt.LCL,
                    ymax = asympt.UCL,
                    color = ask1),
                width = 0,
                position = pd
  ) +
  geom_point(data = fig_2c_m2_emm_dt,
             aes(y = response,
                 color = ask1),
             size = 3,
             position = pd
  ) +

```

2.7. FIGURE 2C – EFFECT OF ASK1 DELETION ON FINAL BODY WEIGHT53

```
# plot p-values (y positions are adjusted by eye)
stat_pvalue_manual(fig_2c_m2_pairs_dt,
                    label = "p.pretty",
                    y.position=c(28.5, 31, 26, 26),
                    tip.length = 0.01) +
  # aesthetics
  ylab("Weight Gain") +
  scale_color_manual(values=fig_2_palette,
                     name = NULL) +
  theme_pubr() +
  theme(legend.position="top") +
  theme(axis.title.x=element_blank()) +
  NULL
fig_2c_gg
```



2.7.10 Figure 2c – report

Results could be reported using either:

(This is inconsistent with plot, if using this, the plot should reverse what factor is on the x-axis and what factor is the grouping (color) variable) Mean weight gain in ASK1F/F mice on HFD was 1.56 (95% CI: 1.33, 1.82, $p < 0.0001$) times

that of ASK1F/F mice on chow while mean weight gain in ASK1 Δ adipo mice on HFD was only 1.22 (95% CI: 1.01, 1.47, $p = 0.036$) times that of ASK1 Δ adipo mice on chow. This reduction in weight gain in ASK1 Δ adipo mice compared to ASK1F/F control mice was 0.79 times (95% CI; 0.61, 1.00, $p = 0.0531$).

(This is consistent with the plot in that its comparing difference in the grouping factor within each level of the factor on the x-axis) Mean weight gain in ASK1 Δ adipo mice on chow was trivially larger (1.03 times) than that in ASK1F/F mice on chow (95% CI: 0.83, 1.27, $p = 0.81$) while mean weight gain in ASK1 Δ adipo mice on HFD was smaller (0.81 times) than that in ASK1F/F control mice on HFD (95% CI: 0.72 , 0.91, $p = 0.0003$). This reduction in weight gain in ASK1 Δ adipo mice compared to ASK1 Δ adipo mice is 0.79 times (95% CI; 0.61, 1.00, $p = 0.0531$).

note to research team. The big difference in p -values between weight difference on chow and weight difference on HFD might lead one to believe there is a “difference in this difference”. Using a p -value = effect strategy, this is not supported.

2.8 Figure 2d – Effect of ASK1 KO on glucose tolerance (whole curve)

2.8.1 Figure 2d – Import

```
range_list <- c("A179:H189", "A191:H199", "A201:H214", "A216:H230")
fig_2d_wide <- data.table(NULL)
for(range_i in range_list){
  part <- import_fig_2_part(range_i)
  fig_2d_wide <- rbind(fig_2d_wide,
                        part)
}
fig_2d_wide[, c("ask1", "diet") := tstrsplit(treatment, " ", fixed=TRUE)]

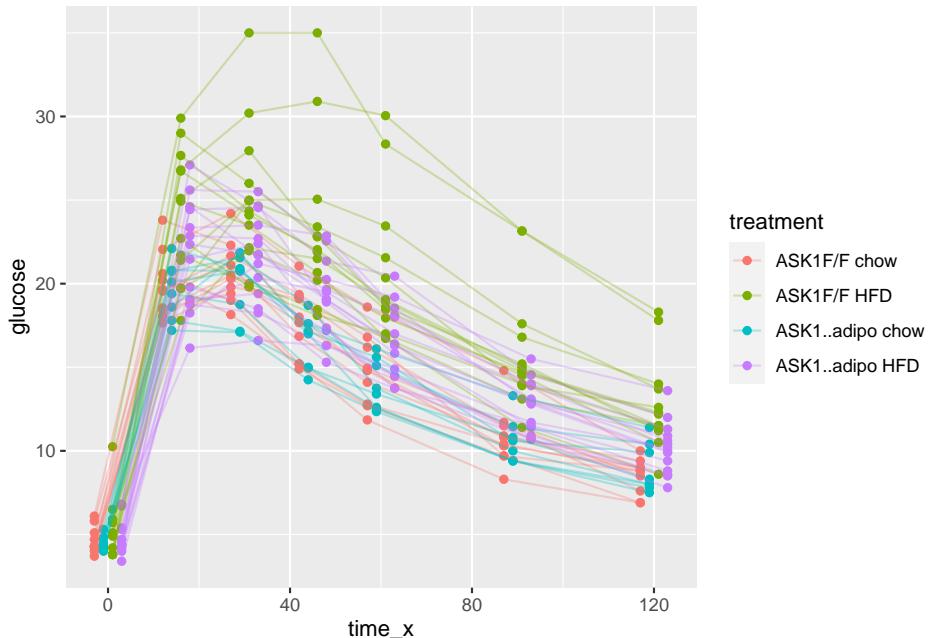
# melt
fig_2d <- melt(fig_2d_wide,
                id.vars = c("treatment",
                           "ask1",
                           "diet",
                           "mouse_id"),
                variable.name = "time",
                value.name = "glucose")
fig_2d[, time := as.numeric(as.character(time))]
```

2.8. FIGURE 2D – EFFECT OF ASK1 KO ON GLUCOSE TOLERANCE (WHOLE CURVE)55

```
# for plot only (not analysis!)
shift <- 2
fig_2d[treatment == "ASK1F/F chow", time_x := time - shift*1.5]
fig_2d[treatment == "ASK1Δadipo chow", time_x := time - shift*.5]
fig_2d[treatment == "ASK1F/F HFD", time_x := time + shift*.5]
fig_2d[treatment == "ASK1Δadipo HFD", time_x := time + shift*1.5]
```

2.8.2 Figure 2d – exploratory plots

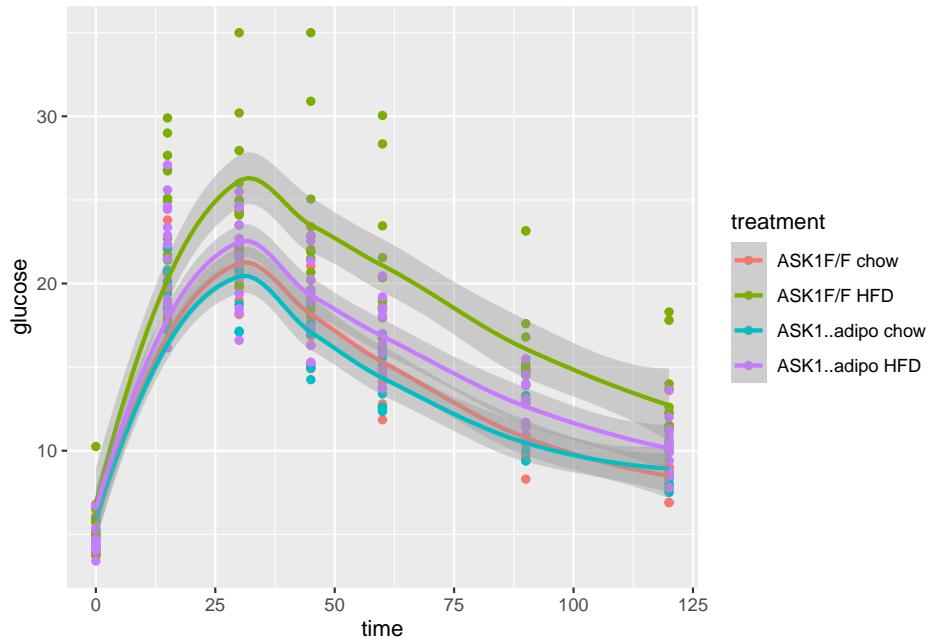
```
qplot(x = time_x, y = glucose, color = treatment, data = fig_2d) +
  geom_line(aes(group = mouse_id), alpha = 0.3)
```



* no obvious unphysical outliers but two mice w/ high values in “F/F HFD”
 * similar at time zero (initial effect is trivial)

use AUC conditional on time 0 glucose

```
qplot(x = time,
      y = glucose,
      data = fig_2d,
      color = treatment) +
  geom_smooth()
```



2.8.3 Figure 2d – fit the model

2.8.4 Figure 2d – check the model

2.8.5 Figure 2d – inference

2.8.6 Figure 2d – plot the model

2.9 Figure 2e – Effect of ASK1 deletion on glucose tolerance (summary measure)

The researchers did create a table to import but this analysis uses the mean post-baseline glucose amount as the response instead of the area under the curve of over the full 120 minutes. This mean is computed as the post-baseline area under the curve divided by the duration of time of the post-baseline measures (105 minutes). This analysis will use fig_2d_wide since there is only one a single Y variable per mouse.

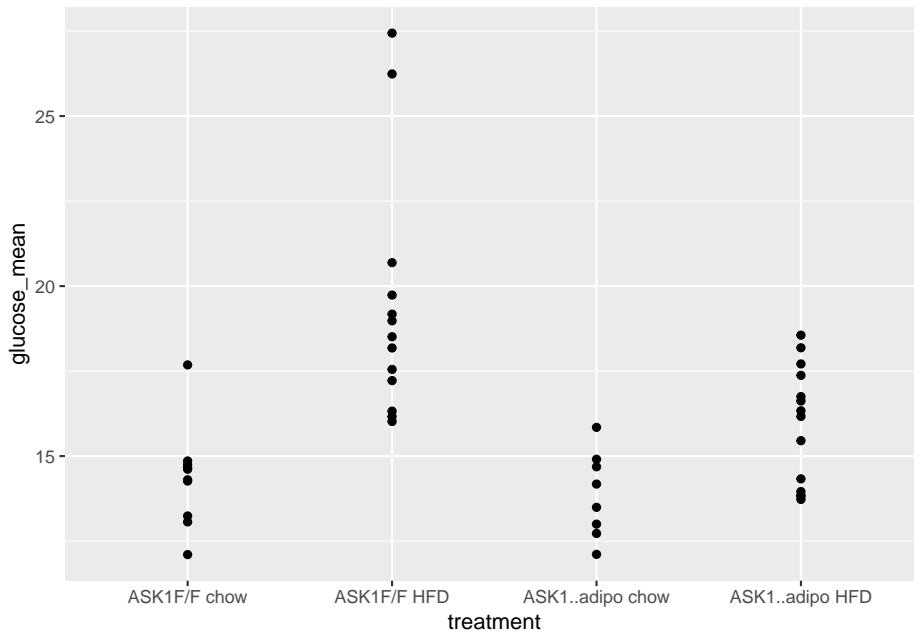
2.9.1 Figure 2e – message the data

```
# AUC of post-baseline values
# do this after melt as we don't need this in long format)
fig_2e <- fig_2d_wide
fig_2e[, glucose_0 := get("0")]

times <- c(0, 15, 30, 45, 60, 90, 120)
time_cols <- as.character(times)
Y <- fig_2e[, .SD, .SDcols = time_cols]
fig_2e[, glucose_mean := apply(Y, 1, auc,
                                x=times,
                                method = "post_0_auc",
                                average = TRUE)]
```

2.9.2 Figure 2e – exploratory plots

```
qplot(x = treatment, y = glucose_mean, data = fig_2e)
```



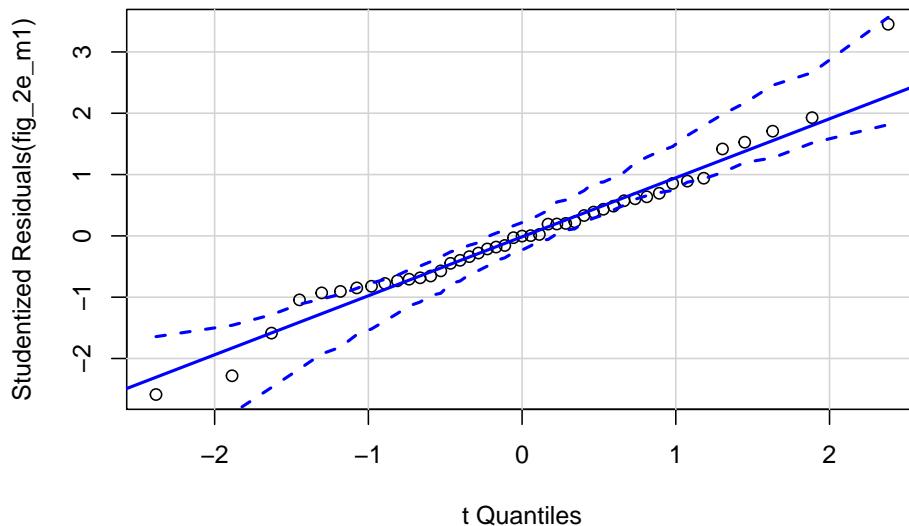
58 CHAPTER 2. ANALYZING EXPERIMENTAL DATA WITH A LINEAR MODEL

2.9.3 Figure 2e – fit the model

```
fig_2e_m1 <- lm(glucose_mean ~ glucose_0 + ask1*diet, data = fig_2e)
```

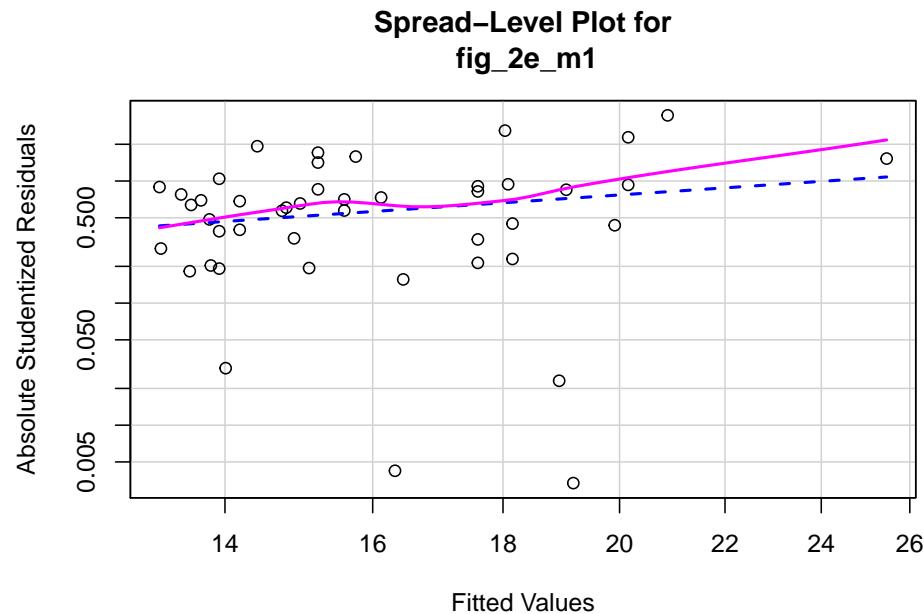
2.9.4 Figure 2e – check the model

```
# check normality assumption
set.seed(1)
qqPlot(fig_2e_m1, id=FALSE)
```



```
spreadLevelPlot(fig_2e_m1, id=FALSE)
```

2.9. FIGURE 2E – EFFECT OF ASK1 DELETION ON GLUCOSE TOLERANCE (SUMMARY MEASURE)59



```
##  
## Suggested power transformation: -0.4035073
```

2.9.5 Figure 2e – inference from the model

```
fig_2e_m1_coef <- coef(summary(fig_2e_m1))  
fig_2e_m1_coef
```

	Estimate	Std. Error	t value	Pr(> t)
## (Intercept)	8.6835026	1.2556730	6.9154169	2.460045e-08
## glucose_0	1.2194720	0.2390919	5.1004329	8.592596e-06
## ask1ASK1Δadipo	-0.3488511	0.8759124	-0.3982716	6.925479e-01
## diethHFD	4.2782121	0.7908612	5.4095613	3.185930e-06
## ask1ASK1Δadipo:diethHFD	-2.7503448	1.1288320	-2.4364518	1.937783e-02

```
fig_2e_m1_emm <- emmeans(fig_2e_m1, specs = c("diet", "ask1"))  
fig_2e_m1_pairs <- contrast(fig_2e_m1_emm,  
                           method = "revpairwise",  
                           simple = "each",  
                           combine = TRUE,  
                           adjust = "none") %>%  
summary(infer = TRUE)
```

60CHAPTER 2. ANALYZING EXPERIMENTAL DATA WITH A LINEAR MODEL

```
fig_2e_m1_emm
```

```
## diet ask1      emmean    SE df lower.CL upper.CL
## chow ASK1F/F     14.7 0.587 40     13.5     15.9
## HFD  ASK1F/F     18.9 0.520 40     17.9     20.0
## chow ASK1Δadipo   14.3 0.659 40     13.0     15.7
## HFD  ASK1Δadipo   15.9 0.493 40     14.9     16.8
##
## Confidence level used: 0.95
```

```
fig_2e_m1_pairs
```

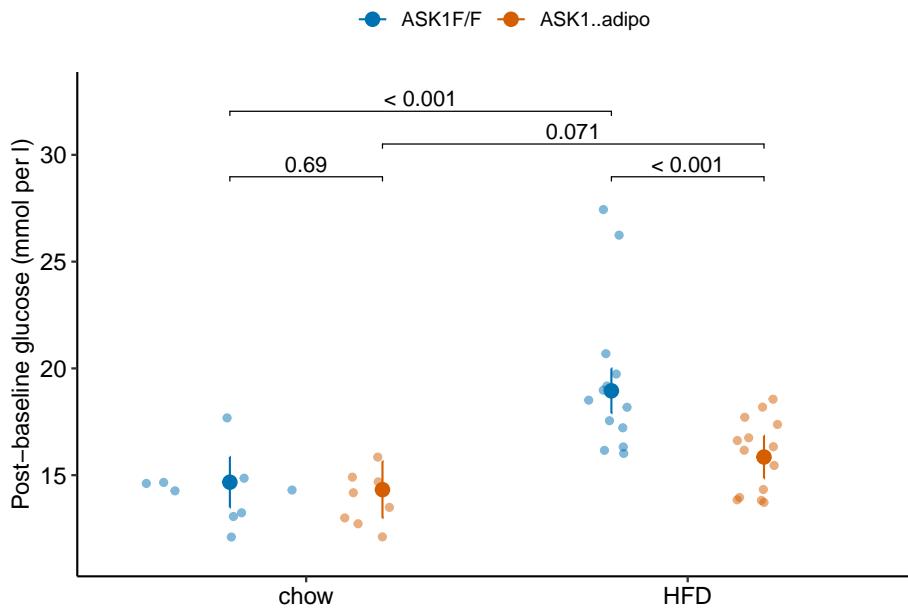
```
## ask1      diet contrast            estimate    SE df lower.CL upper.CL
## ASK1F/F   .    HFD - chow          4.278 0.791 40    2.680    5.88
## ASK1Δadipo .    HFD - chow          1.528 0.824 40   -0.138    3.19
## .          chow ASK1Δadipo - ASK1F/F -0.349 0.876 40   -2.119    1.42
## .          HFD  ASK1Δadipo - ASK1F/F -3.099 0.715 40   -4.544   -1.65
##
## t.ratio p.value
## 5.410 <.0001
## 1.853 0.0712
## -0.398 0.6925
## -4.335 0.0001
##
## Confidence level used: 0.95
```

2.9.6 Figure 2e – plot the model

```
fig_2e_gg <- gg_mean_error(data = fig_2e,
                           fit = fig_2e_m1,
                           fit_emm = fig_2e_m1_emm,
                           fit_pairs = fig_2e_m1_pairs,
                           x_col = "diet",
                           y_col = "glucose_mean",
                           g_col = "ask1",
                           wrap_col = NULL,
                           x_label = "none",
                           y_label = "Post-baseline glucose (mmol per l)",
                           g_label = "",
                           dots = "sina",
                           dodge_width = 0.8,
                           adjust = 0.5,
                           p_show = c(3, 4, 2, 1),
```

2.10. FIGURE 2F – EFFECT OF ASK1 DELETION ON GLUCOSE INFUSION RATE61

```
p_pos = c(1,1,2,3) +
  scale_color_manual(values=fig_2_palette)
fig_2e_gg
```



2.10 Figure 2f – Effect of ASK1 deletion on glucose infusion rate

2.10.1 Figure 2f – import

```
range_2f <- "A239:I240"
treatment_levels <- c("ASK1F/F", "ASK1Δadipo")
fig_2f <- read_excel(file_path,
                      sheet = fig_2_sheet,
                      range = range_2f,
                      col_names = FALSE) %>%
  transpose(make.names=1) %>%
  data.table() %>%
  melt(measure.vars = treatment_levels,
       variable.name = "treatment",
       value.name = "glucose_infusion_rate") %>%
  na.omit()
```

```
fig_2f[, treatment := factor(treatment, treatment_levels)]
```

2.10.2 Figure 2f – exploratory plots

2.10.3 Figure 2f – fit the model

```
fig_2f_m1 <- lm(glucose_infusion_rate ~ treatment, data = fig_2f)
```

2.10.4 Figure 2f – check the model

2.10.5 Figure 2f – inference

```
fig_2f_m1_coef <- summary(fig_2f_m1) %>%
  coef()
fig_2f_m1_emm <- emmeans(fig_2f_m1, specs = "treatment")
fig_2f_m1_pairs <- contrast(fig_2f_m1_emm,
  method = "revpairwise") %>%
  summary(infer = TRUE)
```

```
fig_2f_m1_pairs
```

```
##   contrast           estimate    SE df lower.CL upper.CL t.ratio p.value
##   ASK1Δadipo - ASK1F/F     18.9 6.3 12     5.18    32.6 3.000  0.0111
##
##   Confidence level used: 0.95
```

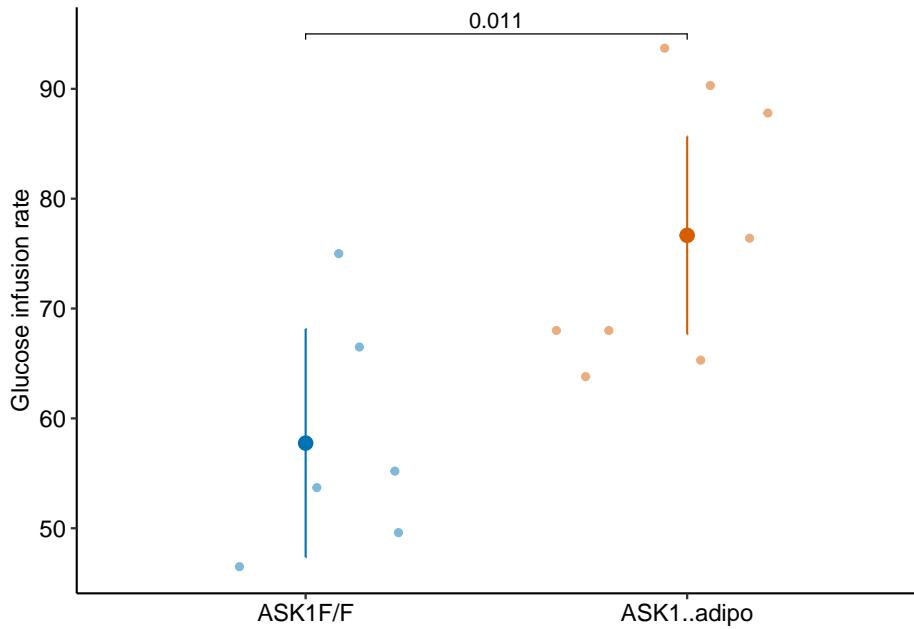
2.10.6 Figure 2f – plot the model

```
fig_2f_m1_emm_dt <- summary(fig_2f_m1_emm) %>%
  data.table
fig_2f_m1_pairs_dt <- data.table(fig_2f_m1_pairs)
fig_2f_m1_pairs_dt[, p.pretty := pvalString(p.value)]
fig_2f_m1_pairs_dt[, group1 := 1]
fig_2f_m1_pairs_dt[, group2 := 2]

fig_2f_gg <- ggplot(data = fig_2f,
```

2.10. FIGURE 2F – EFFECT OF ASK1 DELETION ON GLUCOSE INFUSION RATE63

```
    aes(x = treatment,
        y = glucose_infusion_rate,
        color = treatment)) +  
  
  # points
  geom_sina(alpha = 0.5) +  
  
  # plot means and CI
  geom_errorbar(data = fig_2f_m1_emm_dt,
                aes(y = emmean,
                    ymin = lower.CL,
                    ymax = upper.CL,
                    color = treatment),
                width = 0
  ) +  
  
  geom_point(data = fig_2f_m1_emm_dt,
              aes(y = emmean,
                  color = treatment),
              size = 3
  ) +  
  
  # plot p-values (y positions are adjusted by eye)
  stat_pvalue_manual(fig_2f_m1_pairs_dt,
                     label = "p.pretty",
                     y.position=c(95),
                     tip.length = 0.01) +  
  
  # aesthetics
  ylab("Glucose infusion rate") +
  scale_color_manual(values=fig_2_palette,
                     name = NULL) +
  theme_pubr() +
  theme(legend.position="none") +
  theme(axis.title.x=element_blank()) +
  NULL  
  
fig_2f_gg
```



2.11 Figure 2g – Effect of ASK1 deletion on tissue-specific glucose uptake

2.11.1 Figure 2g – import

```

range_list <- c("A244:G247", "A250:H253")
# import ASK1F/F
fig_2g_1 <- read_excel(file_path,
                        sheet = fig_2_sheet,
                        range = "A244:G247",
                        col_names = FALSE) %>%
  transpose(make.names=1) %>%
  data.table()
fig_2g_1[, treatment := "ASK1F/F"]

# import ASK1Δadipo
fig_2g_2 <- read_excel(file_path,
                        sheet = fig_2_sheet,
                        range = "A250:H253",
                        col_names = FALSE) %>%
  transpose(make.names=1) %>%
  data.table()

```

2.11. FIGURE 2G – EFFECT OF ASK1 DELETION ON TISSUE-SPECIFIC GLUCOSE UPTAKE65

```
fig_2g_2[, treatment := "ASK1Δadipo"]  
  
# combine  
fig_2g <- rbind(fig_2g_1, fig_2g_2)
```

2.11.2 Figure 2g – exploratory plots

2.11.3 Figure 2g – fit the model

```
# a more sophisticated would be a mixed model to dampen noise  
fig_2g_m1_ingWAT <- lm(ingWAT ~ treatment, data = fig_2g)  
fig_2g_m1_epiWAT <- lm(epiWAT ~ treatment, data = fig_2g)  
fig_2g_m1_Muscle <- lm(Muscle ~ treatment, data = fig_2g)  
fig_2g_m1_BAT <- lm(BAT ~ treatment, data = fig_2g)
```

2.11.4 Figure 2g – check the model

2.11.5 Figure 2g – inference

```
fig_2g_infer <- function(m1){  
  m1_emm <- emmeans(m1, specs = "treatment")  
  m1_pairs <- contrast(m1_emm,  
                        method = "revpairwise") %>%  
    summary(infer = TRUE)  
  return(list(emm = m1_emm,  
             pairs = m1_pairs))  
}  
  
fig_2g_m1_emm_dt <- data.table(NULL)  
fig_2g_m1_pairs_dt <- data.table(NULL)  
m1_list <- list(fig_2g_m1_ingWAT,  
                  fig_2g_m1_epiWAT,  
                  fig_2g_m1_Muscle,  
                  fig_2g_m1_BAT)  
y_cols <- c("ingWAT", "epiWAT", "Muscle", "BAT")  
for(i in 1:length(y_cols)){  
  m1_infer <- fig_2g_infer(m1_list[[i]])  
  
  m1_emm_dt <- summary(m1_infer$emm) %>%  
    data.table
```

66CHAPTER 2. ANALYZING EXPERIMENTAL DATA WITH A LINEAR MODEL

```

fig_2g_m1_emm_dt <- rbind(fig_2g_m1_emm_dt,
                           data.table(tissue = y_cols[i], m1_emm_dt))
m1_pairs_dt <- m1_infer$pairs %>%
  data.table
fig_2g_m1_pairs_dt <- rbind(fig_2g_m1_pairs_dt,
                               data.table(tissue = y_cols[i], m1_pairs_dt))
}

fig_2g_m1_pairs_dt

##   tissue      contrast estimate      SE df    lower.CL
## 1: ingWAT ASK1Δadipo - ASK1F/F  3.595000 1.468289 10  0.32344725
## 2: epiWAT ASK1Δadipo - ASK1F/F  1.390238 0.669957 11 -0.08432738
## 3: Muscle ASK1Δadipo - ASK1F/F  2.694048 5.675468 11 -9.79757382
## 4: BAT ASK1Δadipo - ASK1F/F 33.855000 28.715230  7 -34.04572935
##   upper.CL t.ratio p.value
## 1: 6.866553 2.4484273 0.03435010
## 2: 2.864804 2.0751153 0.06222096
## 3: 15.185669 0.4746829 0.64429728
## 4: 101.755729 1.1789911 0.27691810

```

2.11.6 Figure 2g – plot the model

```

# melt fig_2g

fig_2g_long <- melt(fig_2g,
                     id.vars = "treatment",
                     variable.name = "tissue",
                     value.name = "glucose_uptake")

# change name of ASK1Δadipo label
fig_2g_long[treatment == "ASK1Δadipo", treatment := "ASK1-/-adipo"]
fig_2g_m1_emm_dt[treatment == "ASK1Δadipo", treatment := "ASK1-/-adipo"]

fig_2g_m1_pairs_dt[ , p.pretty := pvalString(p.value)]
fig_2g_m1_pairs_dt[, group1 := 1]
fig_2g_m1_pairs_dt[, group2 := 2]

fig_2g_plot <- function(tissue_i,
                         y_lab = FALSE, # title y-axis?
                         g_lab = FALSE # add group label?
                         ){
  y_max <- max(fig_2g_long[tissue == tissue_i, glucose_uptake], na.rm=TRUE)

```

2.11. FIGURE 2G – EFFECT OF ASK1 DELETION ON TISSUE-SPECIFIC GLUCOSE UPTAKE67

```
y_min <- min(fig_2g_long[tissue == tissue_i, glucose_uptake], na.rm=TRUE)
y_pos <- y_max + (y_max-y_min)*.05

gg <- ggplot(data = fig_2g_long[tissue == tissue_i],
             aes(x = treatment,
                 y = glucose_uptake,
                 color = treatment)) +
  
  # points
  geom_sina(alpha = 0.5) +
  
  # plot means and CI
  geom_errorbar(data = fig_2g_m1_emm_dt[tissue == tissue_i],
                aes(y = emmean,
                    ymin = lower.CL,
                    ymax = upper.CL,
                    color = treatment),
                width = 0
  ) +
  
  geom_point(data = fig_2g_m1_emm_dt[tissue == tissue_i],
             aes(y = emmean,
                 color = treatment),
             size = 3
  ) +
  
  # plot p-values (y positions are adjusted by eye)
  stat_pvalue_manual(fig_2g_m1_pairs_dt[tissue == tissue_i],
                     label = "p.pretty",
                     y.position=c(y_pos),
                     tip.length = 0.01) +
  
  # aesthetics
  ylab("Glucose Uptake") +
  scale_color_manual(values=fig_2_palette,
                     name = NULL) +
  ggtitle(tissue_i) +
  
  theme_pubr() +
  theme(legend.position="top") +
  theme(axis.title.x = element_blank(),
        axis.text.x = element_blank(),
        axis.ticks.x = element_blank()) +
  
NULL
```

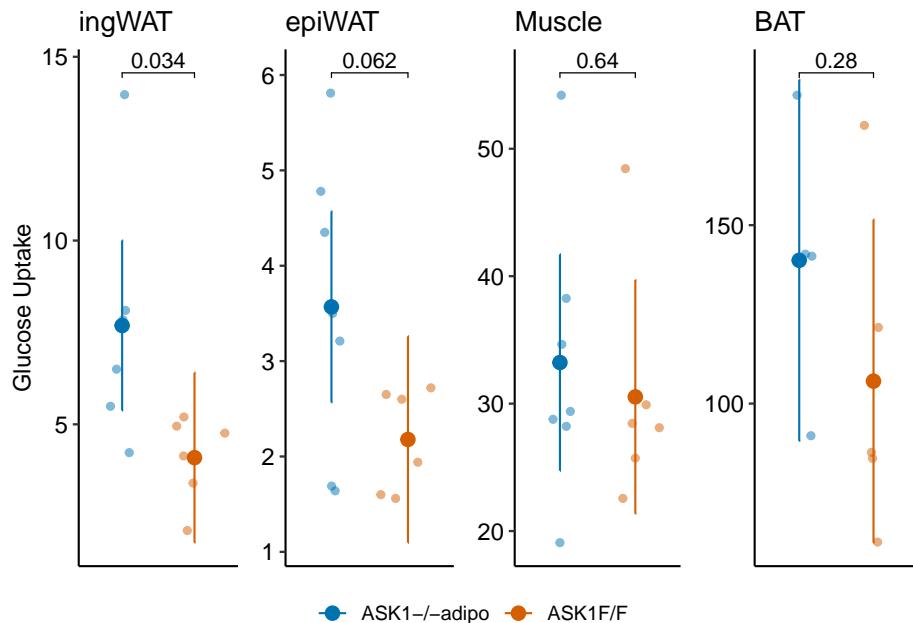
```

if(y_lab == FALSE){
  gg <- gg + theme(axis.title.y = element_blank())
}
if(g_lab == FALSE){
  gg <- gg + theme(legend.position="none")
}

return(gg)
}

y_cols <- c("ingWAT", "epiWAT", "Muscle", "BAT")
legend <- get_legend(fig_2g_plot("ingWAT", y_lab = TRUE, g_lab = TRUE))
gg1 <- fig_2g_plot("ingWAT", y_lab = TRUE, )
gg2 <- fig_2g_plot("epiWAT")
gg3 <- fig_2g_plot("Muscle")
gg4 <- fig_2g_plot("BAT")
top_gg <- plot_grid(gg1, gg2, gg3, gg4,
                      nrow=1,
                      rel_widths = c(1.15, 1, 1.05, 1.1)) # by eye
plot_grid(top_gg, legend,
          nrow=2,
          rel_heights = c(1, 0.1))

```



2.12 Figure 2h

2.13 Figure 2i – Effect of ASK1 deletion on liver TG

```

range_2i <- "A265:G266"
treatment_levels <- c("ASK1F/F", "ASK1Δadipo")
fig_2i <- read_excel(file_path,
                      sheet = fig_2_sheet,
                      range = range_2i,
                      col_names = FALSE) %>%
  transpose(make.names=1) %>%
  data.table() %>%
  melt(measure.vars = treatment_levels,
       variable.name = "treatment",
       value.name = "liver_tg") %>%
  na.omit()

fig_2i[, treatment := factor(treatment, treatment_levels)]
# View(fig_2i)

```

2.13.1 Figure 2i – fit the model

```
fig_2i_m1 <- lm(liver_tg ~ treatment, data = fig_2i)
```

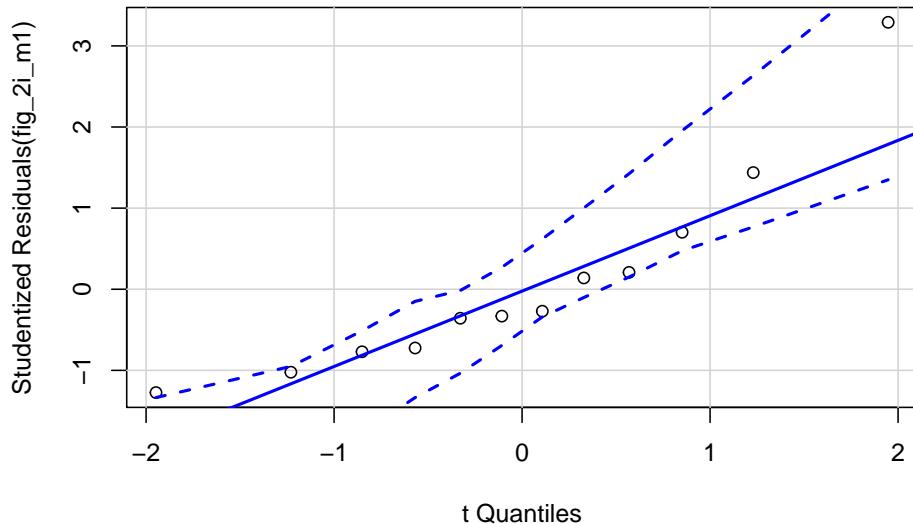
2.13.2 Figure 2i – check the model

```

set.seed(1)
qqPlot(fig_2i_m1, id=FALSE)

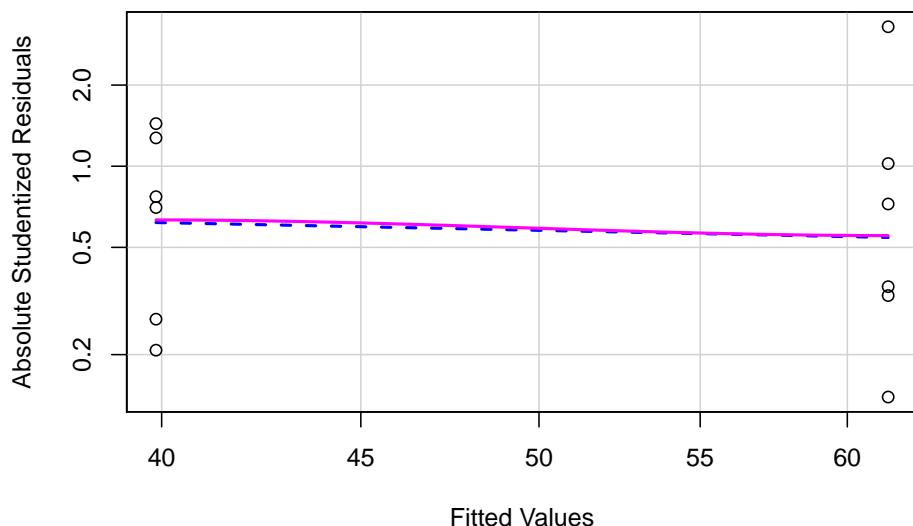
```

70CHAPTER 2. ANALYZING EXPERIMENTAL DATA WITH A LINEAR MODEL



```
spreadLevelPlot(fig_2i_m1, id=FALSE)
```

**Spread-Level Plot for
fig_2i_m1**



```
##  
## Suggested power transformation: 1.294553
```

- The QQ plot looks okay, in the sense that the observed data points (open circles) fall within the boundaries set by the dashed line.

- spread level looks pretty good
- Fit normal model

2.13.3 Figure 2i – inference

```

fig_2i_m1 <- lm(liver_tg ~ treatment, data = fig_2i)
fig_2i_m1_coef <- cbind(coef(summary(fig_2i_m1)),
                         confint(fig_2i_m1))
fig_2i_m1_emm <- emmeans(fig_2i_m1, specs = "treatment")
fig_2i_m1_pairs <- contrast(fig_2i_m1_emm,
                             method = "revpairwise") %>%
  summary(infer = TRUE)

fig_2i_m1_pairs

## contrast           estimate    SE df lower.CL upper.CL t.ratio p.value
## ASK1Δadipo - ASK1F/F     -21.6 7.05 10     -37.3      -5.9 -3.066 0.0119
##
## Confidence level used: 0.95

```

2.13.4 Figure 2i – plot the model

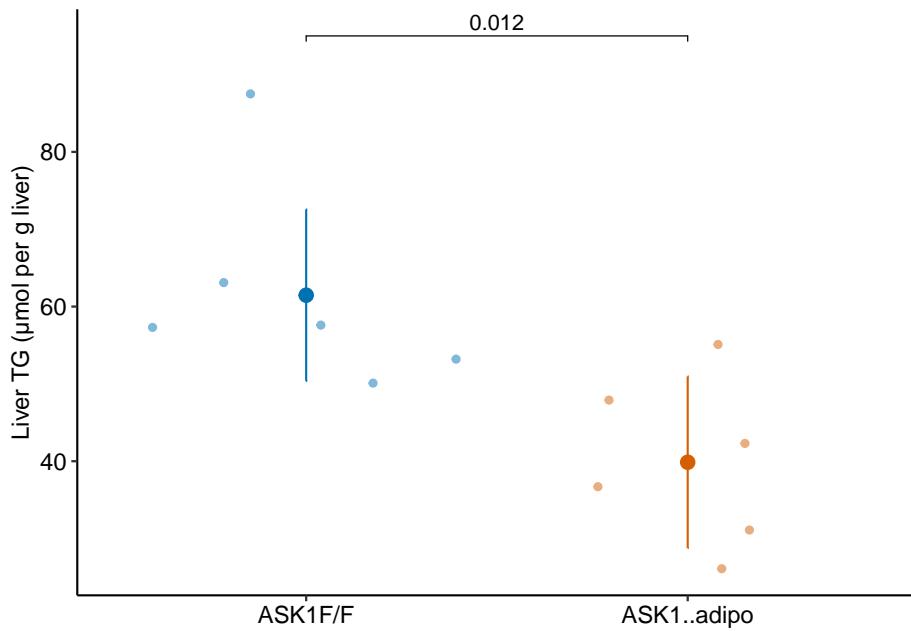
```

fig_2i_m1_emm_dt <- summary(fig_2i_m1_emm) %>%
  data.table
fig_2i_m1_pairs_dt <- data.table(fig_2i_m1_pairs)
fig_2i_m1_pairs_dt[, p.pretty := pvalString(p.value)]
fig_2i_m1_pairs_dt[, group1 := 1]
fig_2i_m1_pairs_dt[, group2 := 2]

fig_2i_gg <- ggplot(data = fig_2i,
                      aes(x = treatment,
                           y = liver_tg,
                           color = treatment)) +
  # points
  geom_sina(alpha = 0.5) +
  # plot means and CI
  geom_errorbar(data = fig_2i_m1_emm_dt,
                aes(y = emmean,

```

```
        ymin = lower.CL,
        ymax = upper.CL,
        color = treatment),
        width = 0
    ) +  
  
    geom_point(data = fig_2i_m1_emm_dt,
                aes(y = emmean,
                    color = treatment),
                size = 3
    ) +  
  
    # plot p-values (y positions are adjusted by eye)
    stat_pvalue_manual(fig_2i_m1_pairs_dt,
                        label = "p.pretty",
                        y.position=c(95),
                        tip.length = 0.01) +  
  
    # aesthetics
    ylab("Liver TG (µmol per g liver)") +
    scale_color_manual(values=fig_2_palette,
                       name = NULL) +
    theme_pubr() +
    theme(legend.position="none") +
    theme(axis.title.x=element_blank()) +
    NULL  
  
fig_2i_gg
```



2.13.5 Figure 2i – report the model

Mean TG level in ASK1 Δ adipo mice on a high-fat diet was 21.6 $\mu\text{mol/g}$ less than that in ASK1F/F mice on a high-fat diet (95% CI: -37.3, -5.9, $p = 0.012$) (Figure xxx).

2.14 Figure 2j

Part III: R fundamentals

Chapter 3

Data – Reading, Wrangling, and Writing

Importing data into R can be a struggle for new R users and, unfortunately, most online “how to import” sources give easy but superficial methods that don’t follow best practices for increasing reproducibility or do not allow flexible organization of files within a project.

(TL;DR – use `here()` from the `here` package)

1. `df <- read.table(file="clipboard")` imports data copied to the clipboard, from an Excel/Sheets file or from an open text file. For this to be semi-reproducible, a comment specifying the filename, worksheet and range that was copied is necessary. More problematic (catastrophically so for reproducibility), is, how does a researcher know that they highlighted and copied the correct range in the Excel sheet?
2. `df <- read.csv(file.choose())` opens the familiar “open file” dialog box, which lets the user navigate to the file of choice. For this to be semi-reproducible, a comment specifying the filename to import is necessary. The catastrophic problem (for reproducibility) is, how does a researcher know which file was actually opened during an R session? The researcher might think they opened “`walker_maine_bee_data_clean_androscoggin.csv`” but mistakenly opened “`walker_maine_bee_data_clean_arostook.csv`”.
3. `df <- read.table(file="my_data.txt")` and `df <- read_excel(file="my_data.xlsx")` are reproducible because the filename is explicitly specified. But, this method requires that “`my_data`” is physically located in the same folder as the file containing the R script (the `.Rmd` file in our case) and this violates the best practice of clean project organization with different folders for the different kinds of files (data, R scripts, images, manuscript text, etc.).

4. R Studio has an elegant import tool in the environment pane that opens a custom dialog box that allows the researcher to navigate to the file, and to specify what part of the file to import, such as the specific sheet and range for an Excel file. This has the same reproducibility issues as #1 and #2 but R Studio includes the equivalent script, which adds all relevant information for reproducibility. One then simply copies and pastes this script into a code chunk and voila! The next time the script is run, the data can be imported from the script without using menus and dialog boxes. Except that..the script does not seem to take into account that the **working directory** of an R Markdown file is not the project folder but the folder containing the R Markdown file and so this two-step method fails. More personally, I'd prefer to run a chunk that quickly opens the data file instead of re-navigating through my file system and re-specifying the sheet and range every time I re-start the project in a new R session.

There are at least three solutions to the issues raised above, all requiring some understanding of **file paths** and directory structure in an operating system. A file such as “my_data.xlsx” has an **absolute** file path, which is the full address of the file (the filename is something like your house street number). The absolute file path of “my_data.xlsx” might be “/Users/jwalker/Documents/applied-biostatistics/data/my_data.xlsx”. A **relative** file path is the file path from the **working directory**. In an R Studio project, the working directory is the project directory, which is the directory containing the .Rproj file. This will be the working directory of the console. *Importantly*, the working directory of an R Markdown code chunk is the folder containing the saved R Markdown file. An R Studio Notebook is an R Markdown file so the working directory of a notebook code chunk is the folder containing the saved notebook file. If an R Markdown file is located within the *rmd* folder, which is located within the project folder, then the relative file path to “my_file.xlsx” is “../data/my_file.xlsx”. The “..” tells the file OS to move “up” into the parent directory (which is the project folder) and the “data” tells the file OS to move “down” into the data folder. These are put together into a single address using “/”. The beauty of relative paths is that they remain the same – and so do not break one’s script – if the project folder, and all of its contents including the *data* folder and the *rmd* folder, is moved to another location on the hard drive (say into a new “Research” folder). By contrast, the absolute file path changes, which breaks any old script.

The three solutions are

1. Create a **relative path** to the file using something like `file_path <- "../data/my_data.xlsx"`. This should *always* work but it fails on some computers. For example, if the project folder is on a Windows OS (but not Mac OS) desktop, the assigned relative address doesn’t seem to look in the folder containing the file.

2. Create a setup chunk that reroutes the working directory to the project folder using the script

```
# use this in a chuck called "setup" to force the working directory to be
# at the level of the project file.
knitr::opts_knit$set(root.dir = rprojroot::find_rstudio_root_file())
```

For this to work, the chunk has to be named “setup”, that is, the text inside the curly brackets at the top of the chunk should be “r setup”. Then, with this chunk, the relative file path is `file_path <- "../data/my_data.xlsx"` if “my_data.xlsx” is immediately inside the data folder which is immediately inside the project folder. This should work on any machine, and should work even if a project folder is moved.

3. Use the function `here()`. The most robust solution seems to be using the function `here()` from the `here` package. The function works something like this

```
data_folder <- "data" # path to data that are imported
file_name <- "my_data.xlsx"
file_path <- here(data_folder, file_name) # paste together parts of the address
my_file <- read_excel(file = file_path)
```

`here()` creates an absolute path, but one that is created on the fly, and will change (or should change) correctly if the project folder is moved on the same machine, to a cloud drive, or to another machine altogether.

3.1 Learning from this chapter

It will be easiest to learn from this chapter by starting with a clean R Markdown file for the chapter. Create a new R Markdown file and save it to the “Rmd” folder of your “applied biostats” project.

Important: import/export scripts will not work properly until the file is saved! Get in the habit of creating the file, saving it immediately, and saving it often.

1. Keep the first chunk that includes the script `knitr::opts_chunk$set(echo = TRUE)`
2. Delete all text below this chunk, starting with the header “## R Markdown”
3. Copy and paste this chunk into your setup chunk

```

knitr::opts_chunk$set(echo = TRUE)

# import and wrangling packages
library(here) # here() creates the absolute path to the file
library(janitor) # clean_names to clean col labels of imported data
library(readxl) # import excel
library(data.table) # use data.table for wrangling

# analysis packages
library(emmeans) # get estimated marginal means and CIs, used for plot

# plotting packages
library(ggplot2) # ggplot environment
library(ggpubr) # publication ready plots

here <- here::here # make sure ``here`` uses `here::here`

# relative paths to project folders
data_folder <- "data" # path to data that are imported
output_folder <- "output" # path to data that are saved
image_folder <- "images"

```

Note on the script above 1. Be kind to the future you by loading only the packages necessary for the code in the R Markdown file that you are working on. If your default is to load everything, the future you will be confused why something was installed. 2. Be kind to the future you by commenting on why a package is loaded; usually this is a specific function from the package 3. `here <- here::here` is my favorite script ever. What is it doing? One can read this as “assign the function `here` from the `here` package to the object `here`” (this is reading the script right to left). Why do this? It turns out the multiple packages define a function called “`here`”. If any of these packages are loaded after the `here` package, then `here` from `here` won’t work – it will be replaced by `here` from the more recently loaded package. To make sure that `here` uses the function from the `here` package, I simply reassign `here` from the `here` package to the object “`here`” *after* loading in all packages.

3.2 Working in R

3.2.1 Importing data

3.2.1.1 Excel file

The Excel dataset is from an experiment on the growth response of zebra finch chicks to an incubation call that presumably signals “hot environment” to the

embryos (Mariette, M.M. and Buchanan, K.L., 2016. Prenatal acoustic communication programs offspring for high posthatching temperatures in a songbird. *Science*, 353(6301), pp.812-814). The source file is from the Dryad Repository here:

file name: “allDatasetsMarietteBuchanan2016.xls”

source: <https://datadryad.org/stash/dataset/doi:10.5061/dryad.v8969>

Steps

1. Copy the title of the article, which is “Prenatal acoustic communication programs offspring for high post-hatching temperatures in a songbird”
2. Create a new folder within the “data” folder. Name the folder the title of the paper by pasting the name from the clipboard. This is the “data from” folder, since it contains the data from the publication with the title of the folder.
3. Download the .xls file into this folder

A .xls file is an old (pre 2007) Microsoft Excel file type. It is a binary file and can only be opened into a readable format with software that knows how to translate the proprietary format. The more modern Excel file type is .xlsx, which contains within it multiple xml components. An xml file is a text file, and so contains readable content, but the content is xml code to display something. In general, I am a big advocate of archiving stuff as text files (manuscripts, data, scripts, blog posts) because these will *always* be readable by future software. Microsoft Excel is not likely to die anytime soon and software that can read .xls and especially .xlsx files (again, .xlsx files are text files) is even less likely to disappear but we can feel even more confident if data are archived as text files. That said, a single microsoft excel file with multiple sheets is an efficient method for distributing data and the readxl package provides excellent tools for reading different sheets of a single .xls or .xlsx file.

The code below uses the function `read_excel()` from the package `readxl`. More about the amazing power of this package is the tidyverse page and chapter 11 in the *R for Data Science* book.

```
folder <- "Prenatal acoustic communication programs offspring for high post-hatching temperatures"
fn <- "allDatasetsMarietteBuchanan2016.xls"
file_path <- here(data_folder, folder, fn)

chick <- read_excel(file_path,
                     sheet = "nestlingMass")
chick <- clean_names(chick) # clean column names
chick <- data.table(chick) # convert to data.table

# View(chick)
```

This text will consistently uses this protocol for storing and retrieving downloaded files. The final line of the chunk is commented out. I do this so that it does not run when the R Markdown sheet is knitted. But I can highlight “View(chick)” with the cursor and “Run selected line(s)” from the “Run” menu and get a new tab with a spreadsheet like view of the imported data.

The first three lines in the script above creates the directory path to the file. This path includes three variables

1. `data_folder` – assigned to “data” in the setup chunk. “data” is a folder within the project folder that contains (or will contain) all datasets for this text. The data come from many different published papers, so the data for each publication gets its own folder within “data”.
2. `folder` – the name of the “data from” folder within “data” containing the data files. In this text, these folder names will always be the name of the published paper.
3. `filename` – the name of the file to read. There may be multiple data files within the publication’s data folder.

These are all put together into the absolute path using the function `here()` from the `here` package. Take a look at the value of `file_path` to confirm.

The next three lines (starting with `chick <-`)

1. `read_excel` imports the data and assign this to a data.frame named `chick`
2. `clean_names` cleans the column names of `chick`
3. `data.table` converts `chick` to a data.table. A data.table is a data.frame with magical properties.

In steps 2 and 3, the functions take the data.frame and process it in some way and then assigned the processed data.frame to an object that has the same name (`chick`). This script can be made slightly more “elegant” using the “pipe” operator `%>%`.

```
chick <- read_excel(file_path,
                     sheet = "nestlingMass") %>% # import
                     clean_names() %>% # clean the column names
                     data.table() # convert to data.table
```

This is a single line of code containing three separate operations all piped together. A way to think about this is

1. The `read_excel` function imports the data from the file located at `file_path` and assigns this data to `chick`. The pipe operator then sends this to

2. the `clean_names` function, which cleans the column names of “chick”. The pipe operator then sends this to
3. the `data.table` function, which convert the data.frame to a data.table.

A 3rd way to do this is with nested functions.

```
chick <- data.table(clean_names(read_excel(file_path,
                                         sheet = "nestlingMass")))) # convert to data.table
```

Some R users think the piped code is more readable then the three separate functions. Maybe. I think we all agree that the nested functions are the least readable. I use pipes, but I think its worth using the three separate functions to learn what each is doing.

Let's back up to understand these steps, and especially the `clean_names` step. In your chunk with three separate lines of code, click on the first line, the one containing `read_excel`, so that the cursor is somewhere on this line. Then click on the “run” menu in the top right of the R Studio panel and choose the first item “Run selected line(s)”. The file will be re-imported.

Look at the column names (or column headers in Excel lingo) of the imported data using `names` or `colnames` (yes, there are elebenty million ways to do anything in R) (`names` is very general in that it can be used to return the names of the parts of any list, while `colnames` is specific to matrix-like objects). Type this into the console, not into your R Markdown chunk:

```
names(chick)
```

```
## [1] "chick_id"                  "brood_id"
## [3] "brood_composition"         "sex"
## [5] "rank_in_nest"              "playback_treatment"
## [7] "nest_temperature_above_ambient" "max_daily_temp_hatch_day"
## [9] "mean_max_temp_hatch_to_day2"   "mean_max_temp_hatch_to_day10"
## [11] "mean_max_temp_hatch_to_day13"  "hatching_mass"
## [13] "day1_mass"                  "day2_mass"
## [15] "day10_mass"                 "day13_mass"
## [17] "day13_tarsus"
```

In general, it is bad practice to include spaces, parentheses, and special characters such as -, \$ or ^, in the column names of a data frame because these increase handling costs later on. The best practice is to replace a blank with an underscore, for example `rank_in_nest`. Some coders separate words with a period (`rank.in.nest`). Others mash words together into a single word like this `rankinnest` but this should generally be avoided because the result can be hard to read. Finally, some coders use Caps to separate words like this `RankInNest`.

This is easier to read than simple concatenation but the underscore is the easiest to read.

The `clean_names` from the `janitor` package is a beautiful function to clean the column names of a data frame including replacing spaces with an underscore and stripping parentheses. The default clean includes changing any uppercase letter to lower case. Many coders like to work with all lowercase variable names to avoid having to hit the shift key. I am one of these.

Worst Practices – resist the temptation to change the column names in the data file, which reduces reproducibility. Leave original data files original. Always increase reproducibility!

colleague blues – Most researchers live in an Excel world and save data in a way that is efficient for computing stuff in Excel but not efficient for statistical analysis using R or other statistical computing software packages (with the exception of Graphpad Prism). Analyzing data will be much less frustrating if the data are saved in a format that facilitates analysis. Best practices for creating data files

1. https://www.youtube.com/watch?time_continue=309&v=Ry2xjTBtNFE
 - An excellent video introduction to best practices for organizing data in a spreadsheet that will subsequently be analyzed by statistics software.
2. Broman, K. W., & Woo, K. H. (2017). Data organization in spreadsheets (No. e3183v1). <https://doi.org/10.7287/peerj.preprints.3183v1> – An excellent review of best practices for organizing data in a spreadsheet.

3.2.1.1.1 The `read_excel` function

`read_excel` is a beautifully flexible function because Excel. Data can be in different sheets and there can be different datasets within a single sheet. And, researchers tend to use Excel like a blackboard in that an Excel sheet often contains calculations such as means, standard deviations and t-tests. When using `read_excel` it is important to send the function enough information to read the correct data. For the chick data, if we simply used

```
chick <- read_excel(file_path) %>%  
  clean_names() %>%  
  data.table()
```

without specifying the sheet, `read_excel` defaults to reading the first sheet (“OccurrenceIncubationCall”), which is not what we wanted. We can specify the exact range to import using the `range =` argument

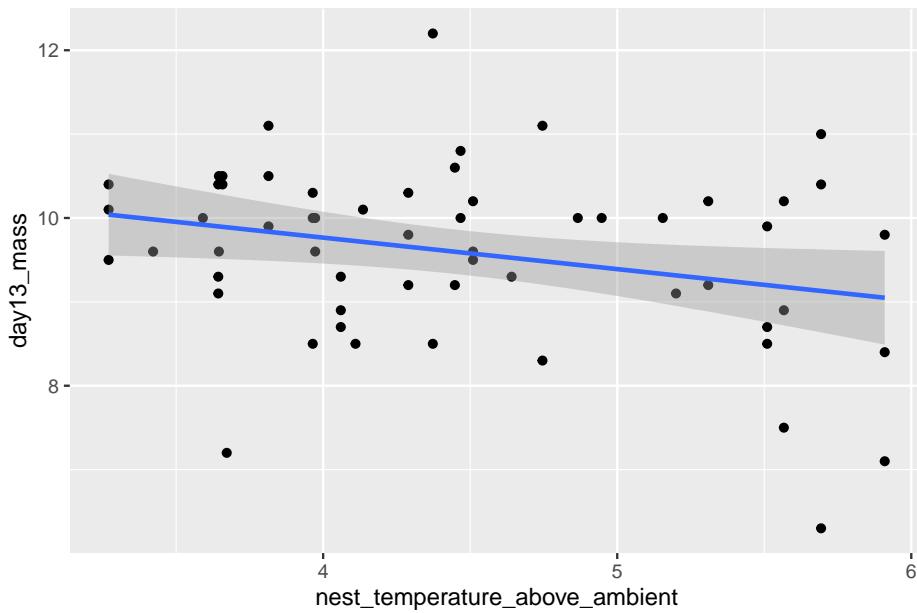
```
chick <- read_excel(file_path,
                     sheet = "nestlingMass",
                     range = "A1:Q131") %>% # import
                     clean_names() %>% # clean the column names
                     data.table() # convert to data.table
```

This isn't necessary for these data because the "nestlingMass" sheet contains only a matrix of data and not extraneous information and the `read_excel` function is smart enough to figure this out. For many of the data sets in wet bench experimental biology, the range argument will be crucial because multiple datasets are archived on a single sheet.

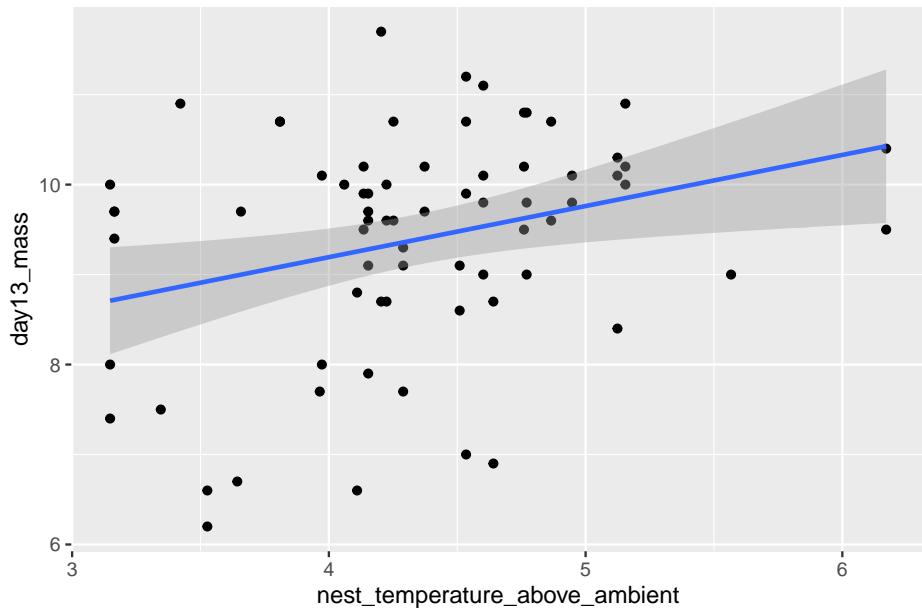
3.2.1.1.2 Explore with plots

Just for fun, let's plot the data and reproduce something close to Fig. 2A and B. We are using the `qplot` function, which is from the `ggplot2` package. `qplots` are quick plots – something you want to do to quickly look at data but don't want to turn into a publication quality plot.

```
qplot(x = nest_temperature_above_ambient,
      y = day13_mass,
      data = chick[playback_treatment == "treat"]) +
      geom_smooth(method = "lm")
```



```
qplot(x = nest_temperature_above_ambient,
      y = day13_mass,
      data = chick[playback_treatment == "cont"]) +
  geom_smooth(method = "lm")
```



Notes on the code to make these plots

1. the names of the columns to use as the x and y axes were not embedded into quotes (e.g. “nest_temperature_above_ambient”). Sometimes a column name has to be in quotes and sometimes not. In general, if the column name is sent to the function as a list (even if its a list with a single item), the names need to be in quotes. Regardless, remember this when you are debugging.
2. The first plot includes only the subset of data in which the value of `playback_treatment` is “treat”. Similarly, the second plot includes only the subset of data in which the value of `playback_treatment` is “cont”. I have sent a subset of the data to the plot function. There are elebenty million ways to subset data in R, I have done it the “data.table way”.
3. each argument in the `qplot` function is on a separate line (created by adding a return after the comma) and the `geom_smooth` function is on a new line. This just makes the function more readable then not doing this. What do you think?

```
qplot(x = nest_temperature_above_ambient, y = day13_mass, data = chick[playback_treatment == "tre
```

4. I have included the name of each argument. This isn't necessary but it makes the function more readable *and* avoids potential bugs. The arguments without the argument names looks like this

```
qplot(nest_temperature_above_ambient, day13_mass, data = chick[playback_treatment == "treat"]) +
```

3.2.1.2 Text file

The example dataset comes from an experiment on the effect of neonicotinoid pesticides on bumble bee colony growth.

file name: “Whitehorn, O’Connor, Wackers, Goulson (2012) Data from ‘Neonicotinoid pesticide reduces bumblebee colony growth and queen production’.csv.csv”

Yes the name of the file has both single quotes in the file name and “.csv” as part of the file name so that, including the extension, the end of the name is “.csv.csv”. This is not a good file name

source: <https://datadryad.org//resource/doi:10.5061/dryad.1805c973>

Steps

1. Copy the title of the paper title, which is “Neonicotinoid pesticide reduces bumblebee colony growth and queen production”
2. Create a new folder within “data”. Name the folder the title of the paper by pasting from the clipboard. This is the “data from” folder, since it contains the data from the publication with the title of the folder.
3. Download the .csv file into this folder

A .csv file is a text file that is comma-delimited, which means that the entries of a row are separated by commas. A text file is readable by any text editor software and most other kinds of software. Datasets that are stored as text files are typically saved as either .csv (where the entries of a row are separated by commas) or .txt (where the entries are separated by tabs). The base R way to read a .csv file is using `read.csv`. The `read.table` function is more versatile, as the delimiter can be specified. The function `fread()` from the `data.table` package is fast, smart, and flexible. It is smart in the sense that it guesses what the delimiter is. Unfortunately, because of spaces in the column labels for this file, `fread` guesses incorrectly (another reason why spaces in column labels should be avoided). To overcome this, the statement below specifies that the file contains a “header” (a line containing column labels)

```

folder <- "Neonicotinoid pesticide reduces bumblebee colony growth and queen production"
filename <- "Whitehorn, O'Connor, Wackers, Goulson (2012) Data from 'Neonicotinoid pest"
file_path <- here(data_folder, folder, filename)
bee <- fread(file_path, header=TRUE) %>%
  clean_names()

```

Here, as with the import of the Excel file, the first three lines create the directory path to the file. There is no need to pipe `bee` to `data.table()` because `fread` automatically imports the data as a `data.table`. It does not need to be converted.

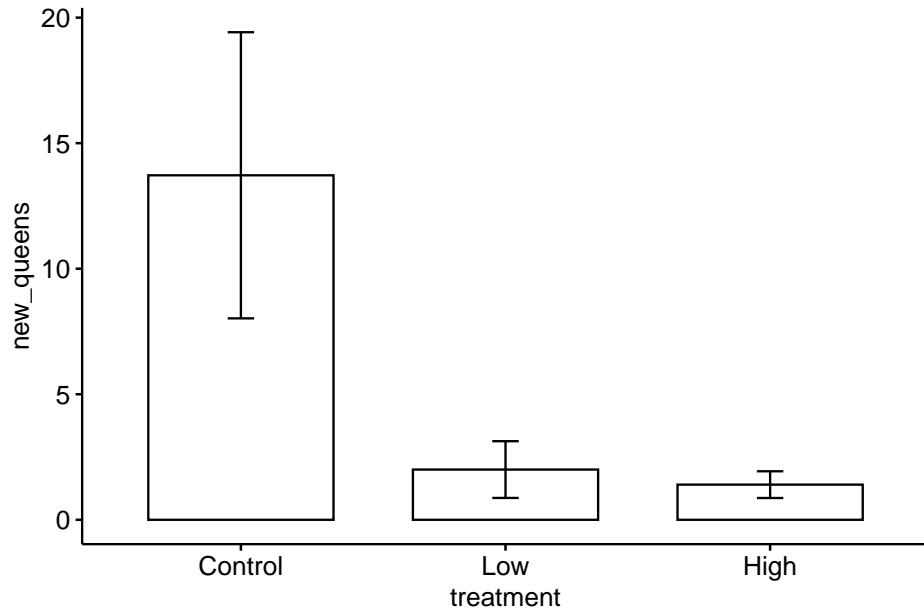
Here is a reproduction of Fig 2 from the journal article using the `ggbarplot` function from the `ggpubr` package.

```

bee[, treatment := factor(treatment, c("Control", "Low", "High"))] # reorder factor levels

ggbarplot(data=bee,
           x="treatment",
           y="new_queens",
           add = "mean_se")

```



3.2.1.3 Troubleshooting file import

If you get an error that starts with “Error: path does not exist:” then R is not “seeing” your specified file given the path you’ve given it.

1. Make sure you loaded the package `here` in a “setup” chunk and that you have run the chunk
2. Make sure you have assigned `data_folder <- "data"` in the setup chunk and have run the chunk.
3. Make sure your “data” folder is *one level* inside your project folder. “one level” means it is not buried deeper inside other folders within the project folder.
4. Make sure your “data from” folder (the folder with the title of the publication) is one level inside your “data” folder
5. Make sure your data file is one level inside the correct “data from” folder.
6. **Bug alert** Make sure you have the name of the “data from ...” folder correct in your script. Do not type the name of the folder. Instead, go to the finder and highlight the folder containing the data file, copy the name, return to the R markdown script, type `folder <- ""` and then paste the clipboard (the name of the folder) in between the quote marks.
7. **Bug alert** Make sure the file name is correct in the script. As with the folder name, I go to the finder and copy the file name and paste it in place. In Windows use `ctrl-a` instead of `ctrl-c` to copy the full filename including the extension.

More generally, Humans are very good at understanding misspelled and OdDLy capitalized words but the R language (or any computer language) is very literal. R is **case sensitive** (some programming languages are not). “Prenatal acoustic communication”, “Prenatal Acoustic Communication”, and “prenatal acoustic communication” are all different values. Spelling AND capitalization have to be perfect, not simply close. Spelling includes *spaces*. A frequent bug is a file name typed as “Prenatal acoustic communication” when the actual name is “Prenatal acoustic communication”. Can you spot the bug? The original (what we need to copy) has two spaces between “acoustic” and “communication” while the incorrect copy has only one.

Spelling bugs are avoided by simply copying and pasting names of folders, names of files, column names of data frames, and level names of factors, which leads to a general rule of R scripting...

3.2.1.4 Rule number one in R scripting {# rule1}

Always copy and paste any text that will be inserted into quotes

Do not try to type it out. You have been warned.

3.3 Data wrangling

Data archived in Excel spreadsheets, at least in wet-bench experimental biology projects, are generally not in a format this is readily analyzed in R, or any

statistical software other than perhaps Graphpad Prism. Use these examples as templates for how to import and wrangle Excel-archived data in your project.

3.3.1 Reshaping data – Wide to long

3.3.1.1 Wide to long – Adipsin data

Source: Adipsin preserves beta cells in diabetic mice and associates with protection from type 2 diabetes in humans

Public source – the Adipsin paper is behind a paywall. A public source of the paper from NIH is available.

[Link to source data](#)

	A	B
1		
2	Glucose uptake	
3	db/db-GFP	db/db-Adipsin
4	19.2	23.4
5	22.3	23.6
6	23.8	22.9
7	20.2	21.4
8	19.4	22
9	22.2	
10		
11		

Fig. 1k of the Adipsin paper presents a bar plot of the glucose uptake in response to control (GFP) or adipsin treatment. A screenshot of the Excel-archived data is shown above. The data are in **wide format**. In wide-format, the values of a single variable (here, this is glucose uptake level) are given in separate columns for each treatment level (group). The values for the GFP group are in Column A and the values for the Adipsin group are in Column B. Wide format is efficient for computations in a spreadsheet, such as computing means and standard deviations of columns of data, and for plotting.

For most statistical analyses of experimental data in R (and most statistics software), all values of a single variable should be in a single column. This

is called **long format**. I've manually rearranged the data from the archived spread sheet into long format by stacking each group's values into a single column, shown in the screen capture below. All values of glucose uptake are in a single column. In long format, there needs to be a way to identify which values belong to which group and this is achieved here with column "treatment". In addition to the treatment column.

	A	B
1	treatment	glucose_uptake
2	db/db-GFP	19.2
3	db/db-GFP	22.3
4	db/db-GFP	23.8
5	db/db-GFP	20.2
6	db/db-GFP	19.4
7	db/db-GFP	22.2
8	db/db-Adipsin	23.4
9	db/db-Adipsin	23.6
10	db/db-Adipsin	22.9
11	db/db-Adipsin	21.4
12	db/db-Adipsin	22

The difference between wide and long also reflects how we think about statistical analysis. When we do a *t*-test to compare the means of glucose uptake between GFP and Adipsin groups, we might think we have two things: the set of glucose uptake values for the GFP group and the set of values for the Adipsin group. When we fit a linear model, we also have two things, the variable *treatment* containing treatment level assignment and the variable *glucose_uptake* containing the glucose uptake values. In wide format, there is nothing to suggest that *treatment* is a variable.

There are many functions to tidy data from wide to long. `melt` from the `data.table` package is especially useful. It is `data.table`'s version of `melt` from the `reshape2` package.

The major arguments of `data.table::melt` are

```
melt(data, id.vars, measure.vars, variable.name, value.name)
```

`melt` takes the data in the columns listed in `measure.vars` and stacks these into a single column named `value.name`. The names of the columns in `measure.vars` are the values of the elements in a new column named

`variable.name`. The elements of any column in `id.vars` are repeated p times, where p is the number of columns that were stacked.

Let's melt the three different response variables of the `adipsin` data and `merge` them into a single `data.table`. There are several ways to combine data sets including `merge` and `cbind`. We'll compare these later.

```
file_folder <- "Adipsin preserves beta cells in diabetic mice and associates with prot
fn <- "41591_2019_610_MOESM3_ESM.xlsx"
file_path <- here(data_folder, file_folder, fn)

treatment_levels <- c("db/db-GFP", "db/db-Adipsin")

# as separate line
fig_1k_wide <- read_excel(file_path,
                           sheet = "Figure 1k",
                           range = "A3:B9")
fig_1k_wide <- data.table(fig_1k_wide)
fig_1k <- melt(fig_1k_wide,
               measure.vars = treatment_levels,
               variable.name = "treatment",
               value.name = "glucose_uptake")

# or piped -- which do you prefer?
fig_1k <- read_excel(file_path,
                      sheet = "Figure 1k",
                      range = "A3:B9") %>%
  data.table() %>%
  melt(measure.vars = treatment_levels,
       variable.name = "treatment",
       value.name = "glucose_uptake")

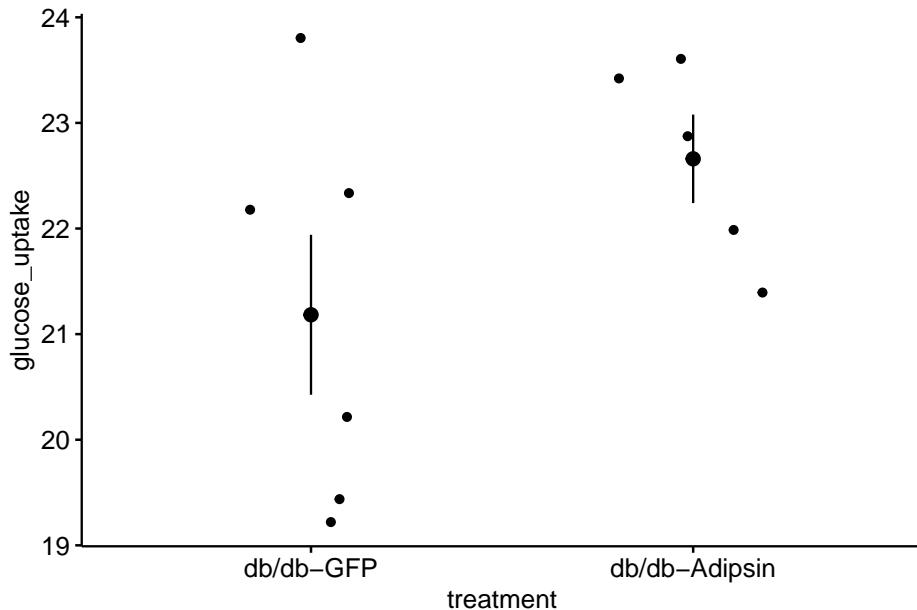
# View(fig_1k) # highlight without the comment sign and "run selected lines()" to view
```

A pretty-good-plot using the `ggpubr` package

```
# put warning=FALSE into the chunk header to suppress the warning

gg <- ggstripchart(x = "treatment",
                    y = "glucose_uptake",
                    add = "mean_se",
                    data = fig_1k)

gg
```



3.3.1.2 Wide to long – Enteric nervous system data

Source: Rolig, A. S., Mittge, E. K., Ganz, J., Troll, J. V., Melancon, E., Wiles, T. J., ... Guillemain, K. (2017). The enteric nervous system promotes intestinal health by constraining microbiota composition. PLOS Biology, 15(2), e2000689.

Source data

Let's import and reshape the data for figure 2d. Look at the excel file and the data in Fig. 2d. There is a single treatment with four levels, but the authors have organized the data in each level in separate columns and used the column header as the level name.

F	G	H	I
Figure 2D-microbiota is sufficient			
GF donor	WT donor	sox10-donor	iapMO donor
4	4	3	4
0	5	5	2
0	8	9	2
0	0	11	2
0	4	3	1
3	1	5	0
1	5	8	2
0	4	23	0
4	3	12	5
1	4	17	5
0	1	6	
	0	6	
	4	12	
	3	12	
	3	0	
	0	1	
	0	14	
	0	0	
	2	2	
		2	
		3	
		8	

Let's melt the data from wide to long by stacking the four columns into a single column "neutrophil_count" and adding a treatment column identifying the group.

```
folder <- "The enteric nervous system promotes intestinal health by constraining micro
filename <- "journal.pbio.2000689.s008.xlsx"
file_path <- here(data_folder, folder, filename)

# figure 2D data
sheet_i <- "Figure 2"
range_i <- "F2:I24"
fig_2d_wide <- read_excel(file_path, sheet=sheet_i, range=range_i) %>%
  clean_names() %>%
  data.table()

# change column names by replacing without "_donor" in each name
# these new column names will become the levels of the treatment factor
new_colnames <- c("gf", "wt", "sox10", "iap_mo")
setnames(fig_2d_wide, old=colnames(fig_2d_wide), new=new_colnames)
```

```

# wide to long
fig_2d <- melt(fig_2d_wide,
               measure.vars=colnames(fig_2d_wide),
               variable.name="treatment",
               value.name="neutrophil_count")

# omit empty rows
fig_2d <- na.omit(fig_2d)

# re-order factors
fig_2d[, treatment := factor(treatment,
                             levels = c("wt", "gf", "sox10", "iap_mo"))]

# View(fig_2d)

```

To learn (instead of just copy and modify), it's best to do this in steps and not run the whole chunk. At each step, look at the result using `View`. The script above includes three extra wrangling steps.

1. Changing column names in `fig_2d_wide`. The column names in wide format will become the treatment level names of the *treatment* factor after reshaping. It will be easier down the road if these names are shorter and the `"_donor"` in each name is redundant. The `setnames` function renames the column names.
2. For these data, the number of measures within the different treatments differs and, as a consequence, there are multiple cells with `NA` which indicates a missing value. `View(fig_2d_wide)` (this can be typed in the console) to see this. After reshaping to long format (`fig_2d`), the rows with missing values become empty rows – there is no useful information in them (View this). To see this, re-run the lines of the chunk up to the line `"# omit empty rows"`. The `na.omit` function deletes any row with missing values. Here, this deletes these information-less rows. Be very careful with `na.omit`. You do not want to delete rows of data that contain information you want.
3. For both analysis and plots, we want to compare values to the control level, which is named “`wt`” for the `fig_2d` data. That is, we want “`wt`” to be the *reference* level. To achieve this, the levels of the factor *treatment* need to be re-ordered using the `levels` argument. (note, I typically do not add “`=`”, but simply pass the list of levels)

3.3.1.3 Wide to long – bee data

The example above is pretty easy, because the all columns in the original data frame are melted (stacked). Here is an example in which only a subset of columns

are stacked. In addition, only a subset of the remaining columns are retained in the long format data frame. The data are from Panel A of supplement Fig. 8 (<https://journals.plos.org/plosbiology/article/file?type=supplementary&id=info:doi/10.1371/journal.pbio.2003467.s019>) from

Source: Kešnerová, L., Mars, R.A., Ellegaard, K.M., Troilo, M., Sauer, U. and Engel, P., 2017. Disentangling metabolic functions of bacteria in the honey bee gut. PLoS biology, 15(12), p.e2003467.

Source data

```
folder <- "Data from Disentangling metabolic functions of bacteria in the honey bee gut"
filename <- "journal.pbio.2003467.s001.xlsx"

# figure 2D data
sheet_i <- "S8 Fig"
range_i <- "A2:H12"
file_path <- here(data_folder, folder, filename)
fig_s8a_wide <- read_excel(file_path,
                           sheet=sheet_i,
                           range=range_i) %>%
  clean_names() %>%
  data.table()

# wide to long
stack_cols <- paste0("replicate", 1:5)
fig_s8a <- melt(fig_s8a_wide,
                 id.vars = c("media", "time_h"),
                 measure.vars = stack_cols,
                 variable.name = "Replicate",
                 value.name = "OD600") # measure of absorbance at 600nm
```

3.3.1.4 Wide to long – stacking multiple sets of columns

This example comes from my lab, where a student measured sprint speed in each fish three times prior to treatment and three times following treatment. The wide format data looked something like this

```
set.seed(1)
fd_wide <- data.table(fish_ID=paste0("fish",1:4),
                      treatment=rep(c("cn", "tr"), each=2),
                      length=rnorm(4, 12, 2),
                      pre_1=rnorm(4, 50, 5),
                      pre_2=rnorm(4, 50, 5),
                      pre_3=rnorm(4, 50, 5),
                      post_1=rnorm(4, 50, 5),
```

```
    post_2=rnorm(4, 50, 5),
    post_3=rnorm(4, 50, 5)
)
knitr::kable(fd_wide, digits=1)
```

fish_ID

treatment

length

pre_1

pre_2

pre_3

post_1

post_2

post_3

fish1

cn

10.7

51.6

52.9

46.9

49.9

54.6

53.1

fish2

cn

12.4

45.9

48.5

38.9

54.7

53.9

49.7

```
fish3
```

```
tr
```

```
10.3
```

```
52.4
```

```
57.6
```

```
55.6
```

```
54.1
```

```
50.4
```

```
49.2
```

```
fish4
```

```
tr
```

```
15.2
```

```
53.7
```

```
51.9
```

```
49.8
```

```
53.0
```

```
40.1
```

```
42.6
```

To analyze the response (post-treatment sprint) adjusted for pre-treatment sprint, the three pre-treatment sprint measures need to be stacked into a single column and the three post-treatment measures need to be stacked into a single column. This is easy using `melt` from the `data.table` package.

```
pre_cols <- paste("pre", 1:3, sep="_")
post_cols <- paste("post", 1:3, sep="_")
fd <- melt(fd_wide,
  id.vars=c("fish_ID", "treatment", "length"),
  measure.vars=list(pre_cols, post_cols),
  variable.name="Order",
  value.name=c("sprint_pre", "sprint_post"))
knitr::kable(fd, digits=1)
```

```
fish_ID
```

```
treatment
```

```
length
```

Order

sprint_pre

sprint_post

fish1

cn

10.7

1

51.6

49.9

fish2

cn

12.4

1

45.9

54.7

fish3

tr

10.3

1

52.4

54.1

fish4

tr

15.2

1

53.7

53.0

fish1

cn

10.7

2

52.9

54.6

fish2

cn

12.4

2

48.5

53.9

fish3

tr

10.3

2

57.6

50.4

fish4

tr

15.2

2

51.9

40.1

fish1

cn

10.7

3

46.9

53.1

fish2

cn

12.4

3

38.9

49.7

fish3

tr

10.3

3

55.6

49.2

fish4

tr

15.2

3

49.8

42.6

3.3.2 Reshaping data – Transpose (turning the columns into rows)

3.3.2.1 Transpose – PI3K inhibitors data

Source: Suppression of insulin feedback enhances the efficacy of PI3K inhibitors

Source data

Figure 3A of this publication is a plot of blood glucose level taken on the same individual mice from four treatment groups over six time periods. Data on a single variable such as blood glucose, taken on the same individual at multiple time points, are known as **longitudinal data** but are often mistakenly called **repeated measures data**. There are mulitple ways to analyze longitudinal data, some goood, some less good. There are two reasonable ways to archive longitudinal data for analysis in R. The Excel-archived data for Figure 3A is neither. A screen capture of two of the four treatment groups is shown below.

	A	B	C	D	E	F	G	H	I	J	K
1	Time (Min.)			Normal Chow				Ketogenic Diet			
2		161	173	163	160	162	161	174	165		
3	30	497	488	481	546	491	337	232	388	389	348
4	80	490	511	501	543	482	348	279	410	394	443
5	90	471	527	504	491	527	368	264	435	466	391
6	120	504	556	426	472	510	284	314	384	330	389
7	180	440	380	474	390	457	323	203	381	273	359

In the archived data the individual mice are in columns. The measure at each time point is in rows. And the treatment group is in blocks. Typical data for analysis in R should have the individual mice in rows and each variable in columns (an exception in experimental biology is omics data, such as RNA

expression levels. Many packages with functions to analyze these data have the genes on each row and the individual on each column). The Figure 3A data are turned on its side. We need to **transpose** the data, or rotate the matrix 90 degrees (make the columns rows and the rows columns) to turn the data into wide format. From this we can create a new data.table with the data in long format.

```
folder <- "Suppression of insulin feedback enhances the efficacy of PI3K inhibitors"
filename <- "41586_2018_343_MOESM6_ESM.xlsx"
file_path <- here(data_folder, folder, filename)
pi3k_side <- read_excel(file_path,
                        sheet = "Figure 3A (Blood Glucose)",
                        range = "A2:U7",
                        col_names = FALSE) %>%
  data.table()

# give columns names as the treatment of each mouse
# verify n=5 per group
treatment_levels <- c("Chow", "Ketogenic", "Metformin", "SGLT2i")
colnames(pi3k_side) <- c("time",
                         rep(treatment_levels, each = 5))

# transpose
# keep colnames in "side" as values of treatment col in "wide"
# make values of "time" in "side" the colnames in "wide"
pi3k_wide <- transpose(pi3k_side,
                        keep.names = "treatment",
                        make.names = "time")

# make a baseline column
pi3k_wide[, glucose_0 := get("0")]

# make-up a mouse id for each mouse
pi3k_wide[, id := paste(treatment, 1:N, sep = "_"), by = treatment]

# make treatment a factor with "chow" as reference
pi3k_wide[, treatment := factor(treatment, treatment_levels)]

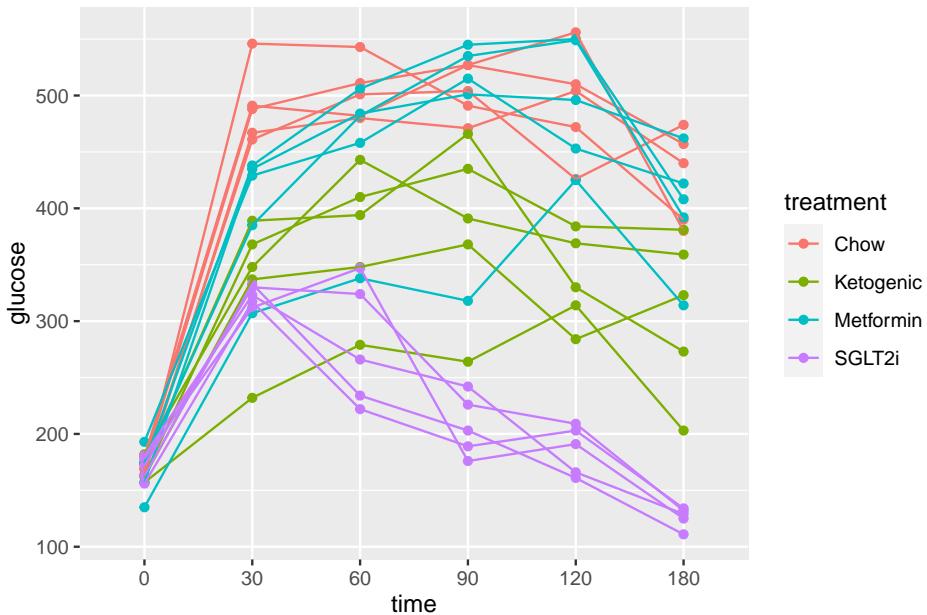
# make a long version
pi3k_long <- melt(pi3k_wide,
                   id.vars = c("treatment", "id", "glucose_0"),
                   variable.name = "time",
                   value.name = "glucose")
```

Notes

1. Read the comments on the usage of the `keep.names` and `make.names` arguments of `transpose`. These are powerful.
2. `pi3k_wide` has column names that are times (in minutes). This presents wrangling problems (column names shouldn't be numbers. Here it is useful to create the long format `data.table` with a time column of numbers). For example, the code above creates copies the column "0" into a new column "glucose_0" using `glucose_0 := get("0")`. Had the code been `glucose_0 := "0"`, all values would be the character "0". Had the code been `glucose_0 := 0`, all values would be the number 0. `get` looks for the column with the name of whatever is inside the parentheses.

Let's do a quick plot to examine the data

```
qplot(x = time,
      y = glucose,
      data = pi3k_long,
      color = treatment) +
  geom_line(aes(group = id))
```



3.3.3 Combining data

Source Bak, A.M., Vendelbo, M.H., Christensen, B., Viggers, R., Bibby, B.M., Rungby, J., Jørgensen, J.O.L., Møller, N. and Jessen, N., 2018. Prolonged

fasting-induced metabolic signatures in human skeletal muscle of lean and obese men. PloS one, 13(9), p.e0200817.

Source data

The data are from a randomized **crossover** design where 18 men (9 lean and 9 obese) were measured for multiple metabolic markers at two times: 1) in a post-absorptive state after 12 hours overnight fast, and 2) in a prolonged fasting state after 72 hours of fasting. In addition, at each time point, metabolic markers were measured prior to and after an insulin infusion. Here, we want to reproduce values in Table 2, which are measures of mean blood insulin and metabolite levels after 12 hours and 72 hours fasting in both the lean and obese groups.

A difficulty for the analyst is that the response data are in the “Table 2” sheet but the variable containing the assignment to “lean” or “obese” group is in the “Table 1” sheet. To analyze these response, the two datasets need to be combined into a single data frame. The important consideration when combining data is that *like is matched with like*. For the fasting dataset, “like” is the subject id, and we have some data for each subject id in Table 1 and other data for the same subject ids in Table 2. This means that we essentially want to glue the columns of table 2 to the columns of table 1 in a way that insures that the correct data for each subject id is on the same row. This is a bit more complicated for these data because Table 1 contains 18 data rows, one for each subject id and Table 2 contains 36 data rows, 2 for each subject id, because each subject has data measured at 12 hours and at 72 hours.

3.3.4 Subsetting data

It is common to see researchers create multiple subsets of data for further processing. This practice should be discouraged because the same variables will be in multiple data frames and it can be hard to keep track of any processing of variables in the different datasets. Instead, subset the data at the level of analysis.

There are many ways to subset data in R. Experienced users tend to divide up into those using base R, those using the tidyverse packages, or those using data.table. Learn one well. This book uses data.table. Before outlining usage in data.table, let’s back up a bit and review different *indexing* systems.

- In Excel, rows are specified (or “indexed”) by numbers and columns by letters. Every cell has an address, for example C2 is the cell in the 2nd row and 3rd column. Notice that in Excel, the column part of the address comes before the row part.
- In statistics, it is extremely common to use a system where x_{ij} is the value of the element in the i th row and j th column of the matrix \mathbf{X} . Notice that in this notation, the row index (i) comes before the column index (j).

- In programming languages, including R, it is extremely common to use a system where `my_data[i, j]` is the value of the element in the i th row and j th column of the matrix-like object named “`my_data`” (such as a data frame in R).
- `data.table` explicitly refers to the row index and column index as i and j .

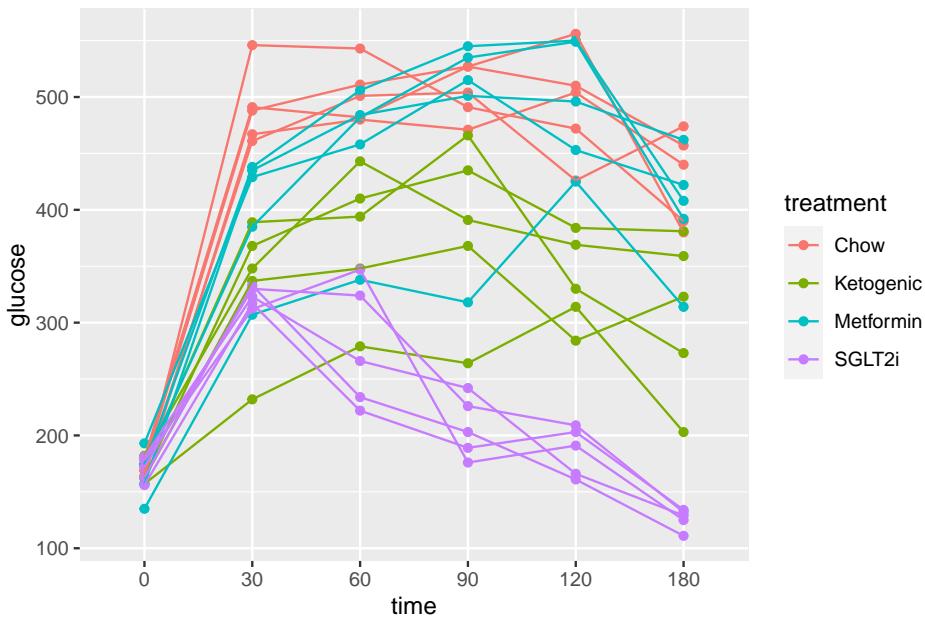
3.3.4.1 Specifying a subset of rows (“observations” or “cases”)

A subset of rows is specified using either a list of row numbers or

In a `data.table`, a subset of rows is specified using either a list of row numbers or a combination of comparison operators (`==`, `!=`, `>`, `<`, `>=`, `<=`, `%in%`) and Boolean logic operators (`&`, `|`, `!` – these are “and”, “or”, “not”) as i .

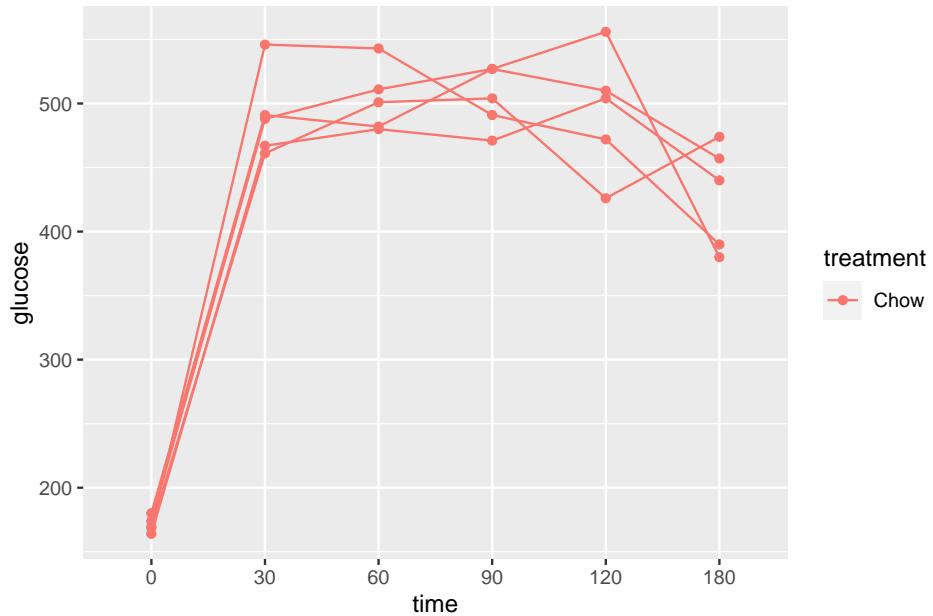
Let’s use the `pi3k_long` data from above to explore this. First, the plot of plasma glucose for all individuals in each treatment group across all time points.

```
qplot(x = time,
      y = glucose,
      data = pi3k_long,
      color = treatment) +
  geom_line(aes(group = id))
```



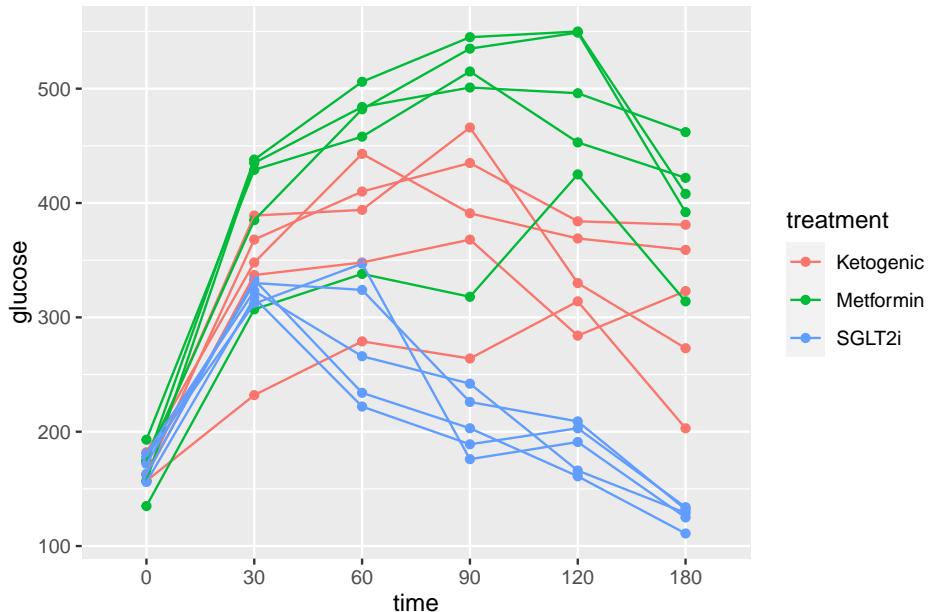
`pi3k_long[treatment == "Chow",]`) is the subset of rows in which entries in the column “treatment” take the value “Chow” using the “is equal” (“`==`”) operator

```
qplot(x = time,
      y = glucose,
      data = pi3k_long[treatment == "Chow", ],
      color = treatment) +
  geom_line(aes(group = id))
```



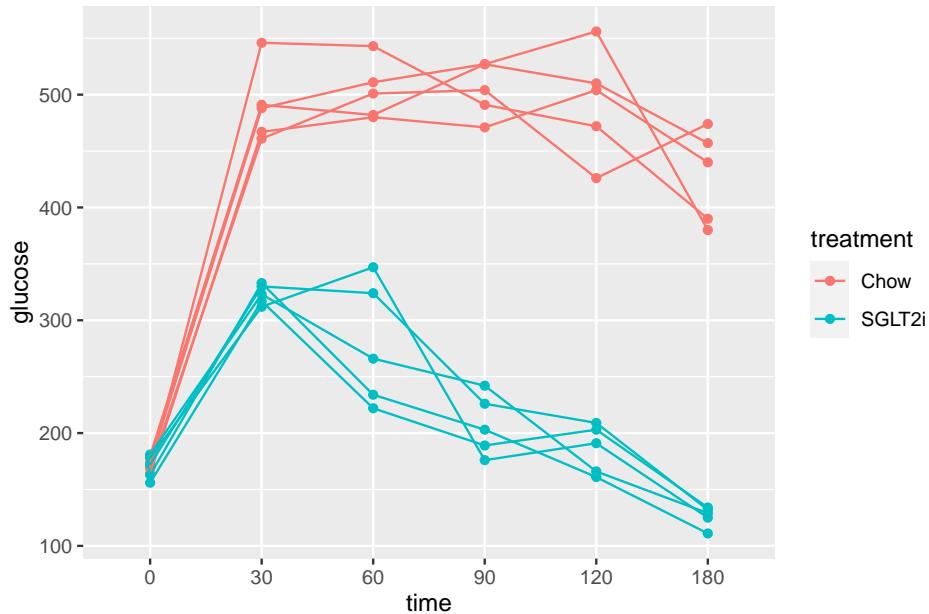
And the subset of rows in which entries in the column “treatment” take any value but “Chow” using the “not equal” operator (“!=”).

```
qplot(x = time,
      y = glucose,
      data = pi3k_long[treatment != "Chow", ],
      color = treatment) +
  geom_line(aes(group = id))
```



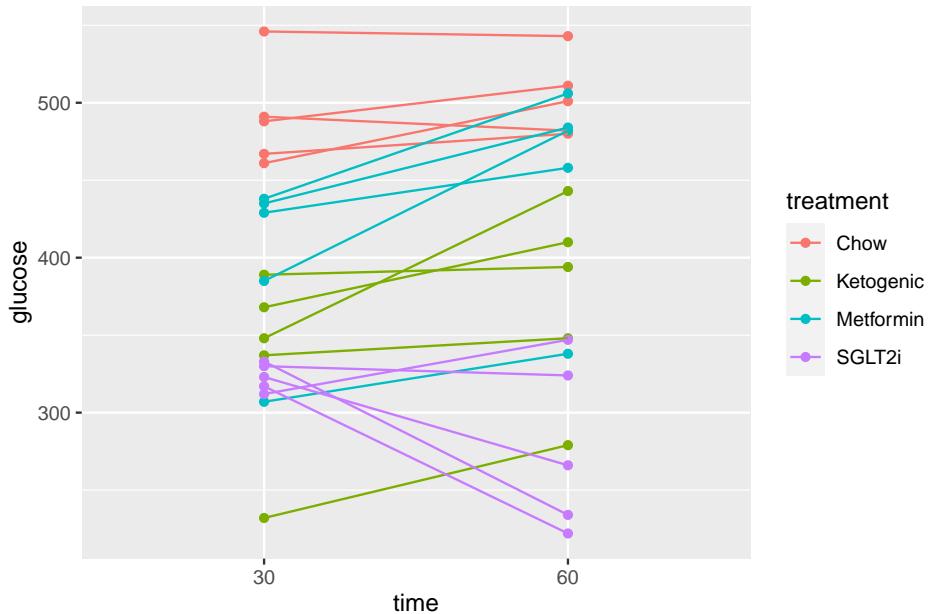
The subset of rows in which entries in the column “treatment” take either the value “Chow” or the value “SGLT2i” by combining two “is equal” (“==”) operators using the OR (“|”) boolean operator

```
qplot(x = time,
      y = glucose,
      data = pi3k_long[treatment == "Chow" | treatment == "SGLT2i",],
      color = treatment) +
  geom_line(aes(group = id))
```



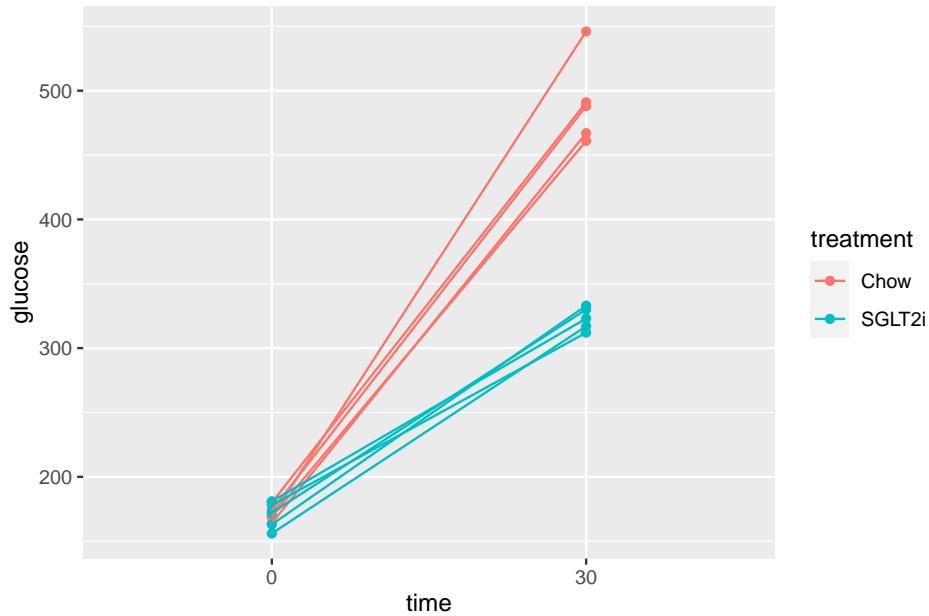
The subset of rows in which entries in the column “time” take either the value “30” or the value “60” using the “in a list” operator (%in%). The values in the “time” column look like integers but are actually treatment levels (which act like string or character variables).

```
qplot(x = time,
      y = glucose,
      data = pi3k_long[time %in% c("30", "60"),],
      color = treatment) +
  geom_line(aes(group = id))
```



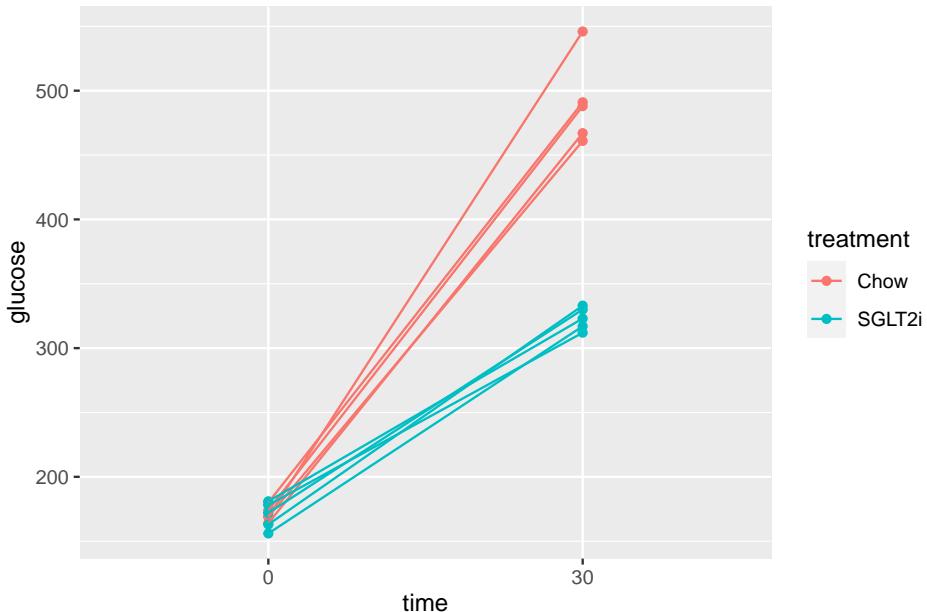
The subset of rows in which entries in the column “time_c” are less than or equal to 60 using the “less than or equal to” operator AND the value in the treatment column is in the list (“Chow”, “SGLT2i”). The two comparisons are combined with the AND (“`&`”) Boolean operator.

```
pi3k_long[, time_c := as.numeric(as.character(time))]
qplot(x = time,
      y = glucose,
      data = pi3k_long[time_c <= 30 & treatment %in% c("Chow", "SGLT2i"), ],
      color = treatment) +
      geom_line(aes(group = id))
```



The same result as above but using different operators. I would describe this as, the subset of rows in which entries in the column “time_c” are less than or equal to 30 using the “less than or equal to” operator AND the value in the treatment column is either “Chow” OR “SGLT2i”. The two comparisons are combined with the AND (“&”) Boolean operator. The order of operations is determined by the parentheses, as with all algebra.

```
pi3k_long[, time_c := as.numeric(as.character(time))]
qplot(x = time,
       y = glucose,
       data = pi3k_long[time_c <= 30 & (treatment == "Chow" | treatment == "SGLT2i"), ],
       color = treatment) +
       geom_line(aes(group = id))
```



3.3.5 Wrangling columns

3.3.5.1 Creating new columns that are functions of values in existing columnnes

3.3.5.2 Change the reference level of a factor

3.3.5.3 Converting a single column with all combinations of a 2×2 factorial experiment into two columns, each containing the two levels of a factor

Source: Tauriello, D., Palomo-Ponce, S., Stork, D. et al. TGF drives immune evasion in genetically reconstituted colon cancer metastasis. Nature 554, 538–543 doi:10.1038/nature25492

Source data

filename: “41586_2018_BFnature25492_MOESM10_ESM.xlsx”

sheet: “Fig. 4h-tumours”

The analysis of the data in Fig. 4h specifies a single X variable “Treatment” with four levels (or groups): “Con”, “Gal”, “aPD-L1”, and “Gal+aPD-L1”. These levels indicate that the design is actually **factorial** with two factors, each with two levels. The first factor has levels “no Gal” and “Gal”. The second factor has levels “no aPD-L1”, “aPD-L1”. The single column Treatment “flattens” the 2×2 factorial design to a 4×1 design. In general, we would want to analyze an

experiment like this as factorial model, because this allows us to make inferences about the *interaction effect* between the two factors. For these inferences, we need a standard error, or a confidence interval, or a *p*-value of the estimate, which we can easily get from the factorial model. In order to analyze the data with a factorial model, we need to create two new columns – one column is the factor variable containing the two levels of Gal and one column is the factor variable containing the two levels of aPD-L1.

```
gal_levels <- c("no Gal", "Gal")
tumor[, gal := ifelse(treatment == "Gal" | treatment == "Gal+aPD-L1",
                      gal_levels[2],
                      gal_levels[1])]

apd_levels <- c("no aPD-L1", "aPD-L1")
tumor[, apdl1 := ifelse(treatment == "aPD-L1" | treatment == "Gal+aPD-L1",
                        apd_levels[2],
                        apd_levels[1])]

# re-order factor levels
tumor[, gal:=factor(gal, gal_levels)]
tumor[, apdl1:=factor(apdl1, apd_levels)]
```

A way to check the results to make sure that our conversion is correct is to compute the sample size for the 2 x 2 combinations, but include the original treatment column in the by list.

```
tumor[!is.na(num_positive_per_mm), .(N=.N), by=.(treatment, gal, apdl1)]

##      treatment     gal     apdl1   N
## 1:       Con no Gal no aPD-L1 124
## 2:       Gal   Gal no aPD-L1  89
## 3:    aPD-L1 no Gal    aPD-L1 101
## 4: Gal+aPD-L1   Gal    aPD-L1  58
```

That looks good.

Bug alert If you break Rule #1, and type in the treatment level “Gal+aPD-L1” as “Gal + aPD-L1”, then you will get new columns containing junk.

```
##      treatment     gal     apdl1   N
## 1:       Con no Gal no aPD-L1 124
## 2:       Gal   Gal no aPD-L1  89
## 3:    aPD-L1 no Gal    aPD-L1 101
## 4: Gal+aPD-L1 no Gal no aPD-L1  58
```

Remember Rule #1. Always copy and paste any text that will be inserted into quotes. This is easily done here by typing `unique(tumor$treatment)` into the console. This function returns the unique values of the column “treatment” of the data.table “tumor”.

```
unique(tumor$treatment) [1] "Con" "Gal" "aPD-L1" "Gal+aPD-L1"
```

Now, copy the name of a level and paste into your code. Repeat until done.

3.3.6 Missing data

Source: Deletion of Cdkn1b in ACI rats leads to increased proliferation and pregnancy-associated changes in the mammary gland due to perturbed systemic endocrine environment

Source data

Supplement Figure 1F of this paper shows weight as a function of age class and genotype for the whole body and 8 organs. There are some missing weights in the Excel-archived data. These missing data are designated with a minus “-” sign. To import these data in correctly, use the `na =` argument in the `read_excel` function.

```
file_folder <- "Deletion of Cdkn1b in ACI rats leads to increased proliferation and pregnancy-ass
file_name <- "journal.pgen.1008002.s008.xlsx"
file_path <- here(data_folder, file_folder, file_name)

fig_s1f <- read_excel(file_path,
                      sheet = "all weights",
                      range = "A2:K57",
                      na = "-",
                      col_names = TRUE) %>%
  clean_names() %>%
  data.table()

fig_s1f[, genotype := factor(genotype, c("+/+", "-/-"))]
fig_s1f[, age_class := ifelse(age_at_sac_wks <= 6.0, "4-6", "8+")]

# View(fig_s1f)
```

Notes

1. In R, a value of “NA” represents missing.

2. The default value for `na` = is an empty (or blank) cell (not a space but a cell that is empty).
3. `na` = accepts a list of strings, for example `na = c("", "-99", "--")` that will all be read as na.

3.3.6.1 Handling missing data

3.3.6.1.1 Many base R functions used for summary measures require NA handling

```
mean(fig_s1f[, ovary]) # returns "NA"

## [1] NA

mean(fig_s1f[, ovary], na.rm = TRUE) # returns the mean

## [1] 0.2489524

sd(fig_s1f[, ovary]) # returns "NA"

## [1] NA

sd(fig_s1f[, ovary], na.rm = TRUE) # returns the mean

## [1] 0.151694

sum(fig_s1f[, ovary]) # returns "NA"

## [1] NA

sum(fig_s1f[, ovary], na.rm = TRUE) # returns the mean

## [1] 10.456
```

There are many ways to get the sample size for a particular variable. Be careful if using `length()` which counts NA as part of the vector of values.

3.3.6.1.2 The `!is.na` function is useful

```

length(fig_s1f[, ovary])

## [1] 55

length(fig_s1f[!is.na(ovary), ovary])

## [1] 42

```

Notes

1. `!is.na(ovary)` is taking the subset of rows of `fig_s1f` for which the value of “ovary” is not NA (`!is.na` is read “not is.na”)

This is especially useful if you are creating your own code uses counts. Here I create a table of means, standard error of the mean, and 95% CIs of the mean for each genotype group. But first, this script generates the wrong N for each group (since there are missing values), although the mean and SD are correct.

```

fig_s1f[, .(mean = mean(spleen, na.rm = TRUE),
            n = .N,
            sd = sd(spleen, na.rm = TRUE)),
        by = genotype]

##    genotype      mean      n       sd
## 1:     -/- 0.5801333 21 0.13680480
## 2:     +/+ 0.2956667 34 0.04460855

```

To compute the correct n , which will be necessary for computing the SE and the CI, use `!is.na`

```

spleen_summary <- fig_s1f[!is.na(spleen), .(mean = mean(spleen),
                                         n = .N,
                                         sd = sd(spleen)),
                           by = genotype]
spleen_summary[, se := sd/sqrt(n)]
spleen_summary[, lower := mean + se*qt(.025, (n-1))]
spleen_summary[, upper := mean + se*qt(.975, (n-1))]
spleen_summary

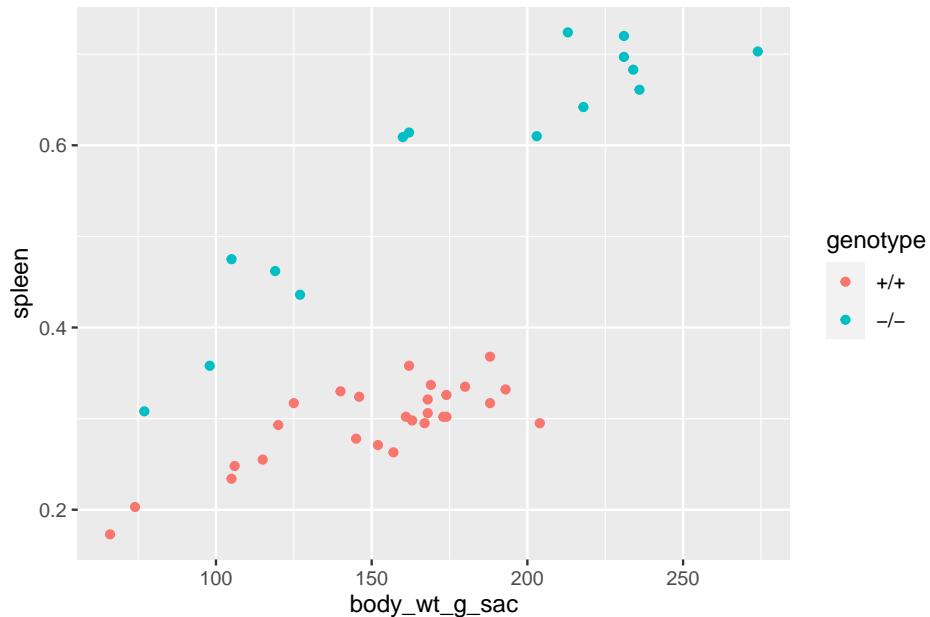
##    genotype      mean      n       sd       se      lower      upper
## 1:     -/- 0.5801333 15 0.13680480 0.03532285 0.5043734 0.6558933
## 2:     +/+ 0.2956667 27 0.04460855 0.00858492 0.2780201 0.3133132

```

3.3.6.1.3 ggplot functions automatically handle missing values

with a useful warning.

```
qplot(x = body_wt_g_sac,
      y = spleen,
      color = genotype,
      data = fig_s1f)
```



3.3.6.1.4 Regression model functions (lm, glm, gls, etc.) handle missing values by default

Missing data in regression model functions such as `lm` are handled using the argument `na.action` = and the default is “`na.omit`”, which omits any rows that contain a missing value in one or more of the model variables (it includes rows if these contain missing values only in the columns not included in the model). It’s as if the user took the subset of data including only the columns containing the model variables and then deleted any row with missing values.

Here is the coefficient table of the fit model object that did not explicitly tell the `lm` function how to handle missing data.

```
m1 <- lm(spleen ~ body_wt_g_sac + genotype,
          data = fig_s1f)
coef(summary(m1))
```

```

##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.04238009 0.0242993900 1.744081 8.902319e-02
## body_wt_g_sac 0.00167493 0.0001506493 11.118067 1.170042e-13
## genotype/-/ 0.23760586 0.0147600545 16.097898 8.072069e-19

```

Here is the coefficient table of the fit model object that did explicitly tell `lm` how to handle missing data, using the argument `na.action = "na.exclude"`. These coefficient tables are the same.

```

m2 <- lm(spleen ~ body_wt_g_sac + genotype,
          data = fig_sf1,
          na.action = "na.exclude")
coef(summary(m2))

##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.04238009 0.0242993900 1.744081 8.902319e-02
## body_wt_g_sac 0.00167493 0.0001506493 11.118067 1.170042e-13
## genotype/-/ 0.23760586 0.0147600545 16.097898 8.072069e-19

```

3.3.6.2 But...beware of fitted, predicted, or residual values from regression model functions unless you've explicitly told the function how to handle missing values

Use `na.action = "na.exclude"` if you want to add the fitted (or predicted) values or residuals as new columns in the original data object (`fig_sf1`). Compare the length of the fitted values vector from models `m1` (using the default “`na.omit`”) and `m2` (using the “`na.exclude`”).

```

length(fitted(m1))

## [1] 42

length(fitted(m2))

## [1] 55

```

There are 55 observations (rows in the data) but only 42 complete rows with no missing values. The vector of fitted values from `m1` has 42 fitted values. The vector of fitted values from `m2` has 55 elements, the 42 fitted values plus 13 NA elements.

This is important if we want to do something like add the fitted values (or residuals, or some function of these) to the original data object (`fig_sf1`). Here I compute the spleen weights adjusted to the mean body weight of the control (“`/+`”) group using the residuals from `m1` and `m2`.

```

mean_x_control <- mean(fig_s1f[genotype == "+/+", body_wt_g_sac])
b <- coef(m1)
fig_s1f[, spleen_adj_m1 := b[1] +
  b[2]*mean_x_control +
  b[3]*(as.integer(genotype)-1 +
  residuals(m1))]
fig_s1f[, spleen_adj_m2 := b[1] +
  b[2]*mean_x_control +
  b[3]*(as.integer(genotype)-1 +
  residuals(m2))]
# View(fig_s1f)

```

The computation of “spleen_adj_m1” returns a warning that the values of `residuals(m1)` were recycled (the first 42 elements of the new column were filled with the 42 residuals and the last 13 elements of the new column were filled with the first 13 residuals) – after the first row of missing data, all of these computed adjusted values are wrong. Using `residuals(m2)`, the adjusted values are matched to the correct row and the rows with missing variables do not have an adjusted value (because there is no residual to compute this).

3.4 Saving data

For many projects, it is uncommon to save data. I might save simulated data if it takes a long time (tens of minutes to hours or even days) to generate these and I simply want to work with the simulated data in the future and not have to regenerate it. Or I might save processed data if it takes a long time to import and process and I want to analyze the processed data in the future and not have to re-import and process it.

If the data will only be used in this or future R projects, the data can be saved as an R object using `saveRDS()`

```

outfile_name <- "Prenatal acoustic communication programs offspring for high post-hatch
save_file_path <- here(output_folder, outfile_name)
saveRDS(object = chick, file = save_file_path)

# to read this use
chick <- readRDS(save_file_path)

```

Reading a large .Rds file is very fast compared to reading the same data stored as a text file. However, if the data need to be imported into some other software, such as a spreadsheet, then save the data as a text file.

```
# save the data to output folder

# tab delimited
outfile_name <- "Prenatal acoustic communication programs offspring for high post-hatching temper
save_file_path <- here(output_folder, outfile_name)
write.table(chick, save_file_path, sep="\t", quote=FALSE)

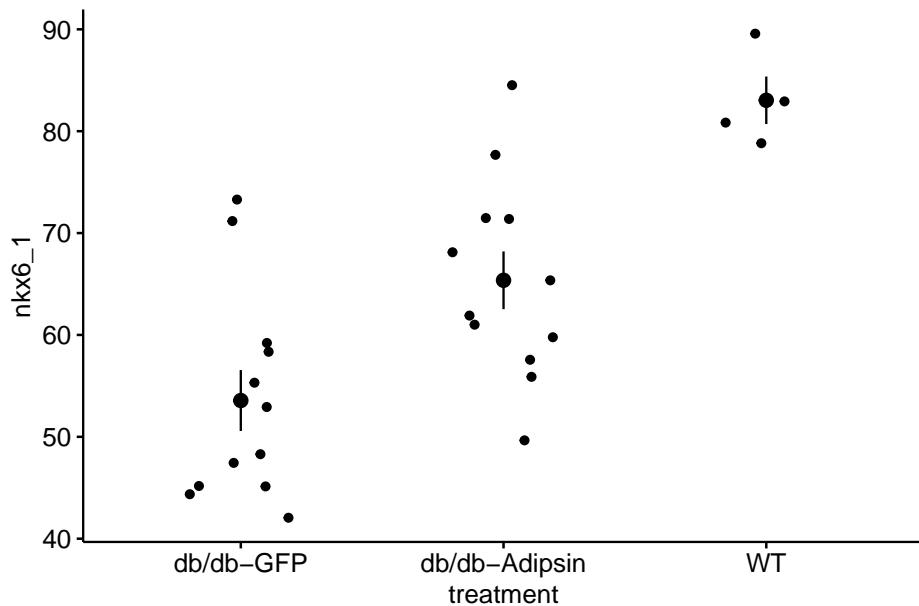
# comma delimited
outfile_name <- "Prenatal acoustic communication programs offspring for high post-hatching temper
save_file_path <- here(output_folder, outfile_name)
write.table(chick, save_file_path, sep=",", quote=FALSE)
```

Look at your project directory to make sure the file is where it should be! We used `write.table()` to create a tab-delimited text file using `sep="\t"` to specify tabs to separate the row elements. `"\t"` is the standard character string for a tab. Check in your output folder and open the file in a text editor.

3.5 Exercises

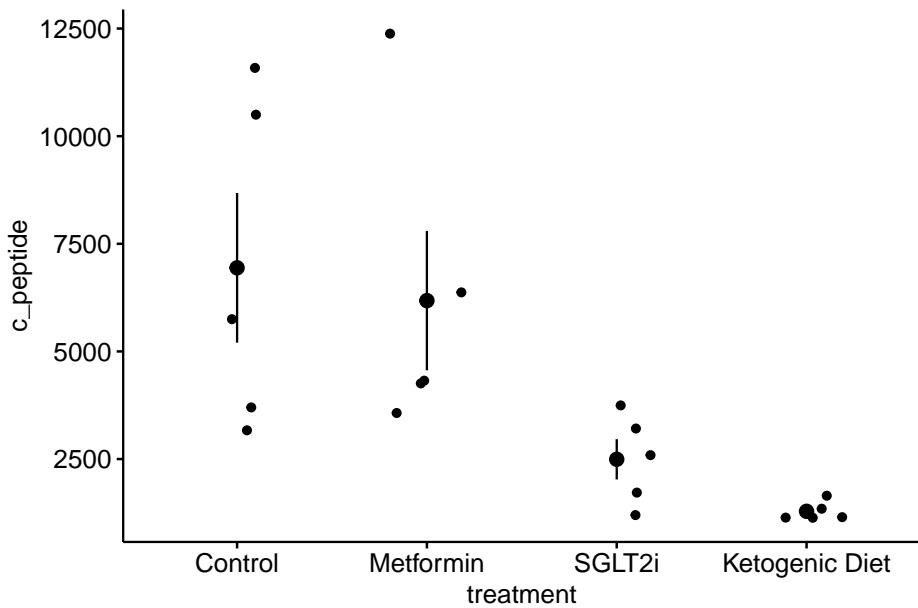
1. Import and pretty-good-plot the data for Figure 2i of the Adipsin paper. You will need to download and archive the Excel file for “Figure 2”. Store this within the “Adipsin preserves beta cells...” folder.
 - The data are the percent of cells staining for NKKX6.1, which is a transcription factor protein that regulates beta cell development in the pancreas. Beta cells sense glucose levels in the blood and secrete insulin. Disruption of the insulin signaling system results in Diabetes mellitus.
 - The data are in wide format, with each treatment group in a separate column. The data need to be melted into long format with a new column called “treatment”.
 - This will give you a pretty good plot of the data (if the data object is named “adipsin_fig2i”)

```
ggstripchart(data = adipsin_fig2i,
             x = "treatment",
             y = "nkkx6_1",
             add = "mean_se")
```



- Import and quick pretty-good-plot the data for Figure 3b of the PI3K paper. You will need to download and archive the Excel file for “Figure 3”. Store this within the “Suppression of insulin feedback enhances...” folder.

- The data are c-peptide levels in response to the treatments. C-peptide is cleaved from the pro-insulin polypeptide and circulates in the blood and is a marker of how much insulin is being produced by the beta cells of the pancreas.
- The data are in wide format, with each treatment group in a separate column. The data need to be melted into long format with a new column called “treatment”.
- Modify the code from exercise 1 to pretty-good-plot the data as in exercise 1.



Chapter 4

Plotting Models

So, along the lines of Sarah Susanka’s “Not So Big House,” Colbert asks the group, “What would a Pretty Good House look like?” – Michael Maines¹

When it comes to plotting, many researchers mindlessly generate plots that are easily generated by the software and look like the typical plots published in the field. The resulting plot is comforting because it is familiar, not because it effectively communicates what a good plot should communicate – the model results.

Plots should be the focus of both the reader and researcher. Instead of mindless plotting, a researcher should ask a series of questions of every plot

1. What is the point of each element in a plot?
2. Are these the points that I most want to communicate?
3. Are there better practices for communicating these points?
4. Are the points that I want to communicate that are not covered by these elements?

The answer to these questions should inform what is and what is not plotted. The result is a pretty good plot. The idea of a pretty good plot is borrowed from the “pretty good house” concept that grew out of a collaborative group of builders and architects in Northern New England. The “pretty good house” combines best practices for building an earth friendly, high performance home at a reasonable cost. There is no pretty good house governing body that awards certificates of achievement but, instead, a set of metrics and a collection of building practices that can achieve these.

A typical pretty good plot contains some combination of

¹“The Pretty Good House - Finding the right balance between construction cost and energy performance”. <https://www.greenbuildingadvisor.com/article/the-pretty-good-house>

1. Modeled effects with confidence intervals. “Effects” are the coefficients of a model, or contrasts constructed from the model, such as pairwise differences between the means of the levels of a factor. Inferences are typically made from the estimated effects
2. Modeled means and standard errors or confidence intervals.
3. Raw data points or a summary distribution of these.

4.1 Pretty good plots show the model and the data

The data to introduce best practices in plotting come from “The enteric nervous system promotes intestinal health by constraining microbiota composition”². The researchers found that zebrafish with a *sox10* mutation lacked an enteric nervous system and developed a microbiota-dependent inflammation. The paper includes several experiments to probe the hypothesis that the ENS regulates microbial community composition and, in turn, inflammatory status. The data here are from Fig. 2 of the paper, which reports the results of one of a set of experiments to test the hypothesis that microbiota from *sox10* mutants (that induce inflammation) are necessary and sufficient to induce inflammation in wildtype guts. In this experiment, homogenized intestines and their microbial community from four different donor groups were added to the flasks housing the zebrafish. The response variable is neutrophil count. Neutrophils are a white blood cell that increase in number during inflammation. The four treatment levels are the different donors of intestinal microbes: wt (wild type), gf (germ free, so no microbes are transferred), iap_mo (a control “for the possibility that nonbacterial factors such as host pro-inflammatory cytokines rather than microbial derived factors cause transmissible intestinal inflammation”), and *sox10*.

4.1.1 Pretty good plot component 1: Modeled effects plot

Biologists infer the biological consequences of a treatment by interpreting the magnitude and sign of treatment “effects”, such as the differences in means among treatment levels. Why then do we mostly plot treatment level means, where effects can only be inferred *indirectly*, by mentally computing differences in means? Instead, our primary plots should be effects plots, which *directly* communicate treatment effects, and the uncertainty in the estimates of these effects.

The y-axis contains all pairwise comparisons among the four treatment levels. The x-axis is the response, which is the ratio of the means of the two groups

²Rolig, A.S., Mittge, E.K., Ganz, J., Troll, J.V., Melancon, E., Wiles, T.J., Alligood, K., Stephens, W.Z., Eisen, J.S. and Guillemin, K., 2017. The enteric nervous system promotes intestinal health by constraining microbiota composition. PLoS biology, 15(2), p.e2000689

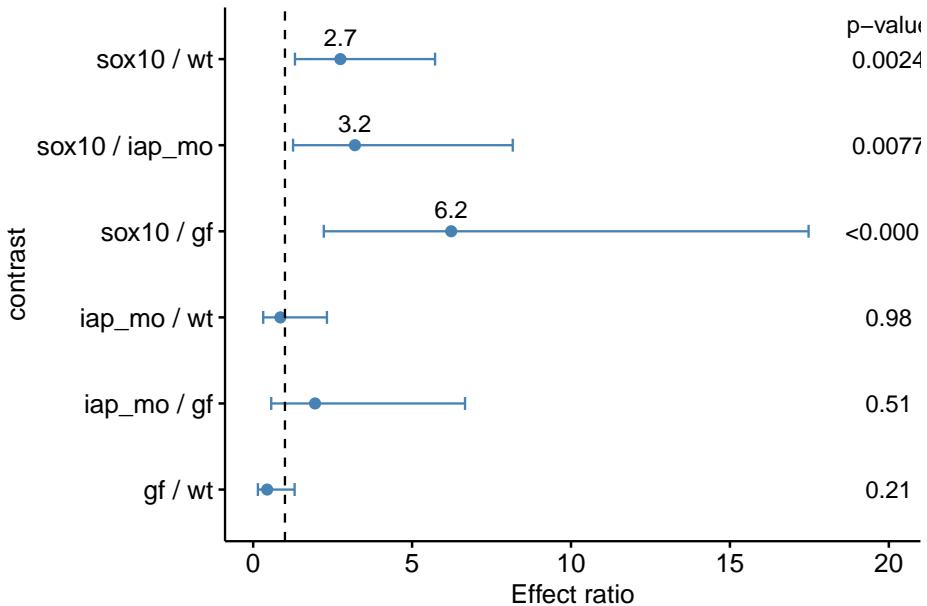


Figure 4.1: Effects Plot

in the comparison. For example, the top comparison shows that guts in fish exposed to sox10 donors have 2.7X more neutrophils per length of gut than guts in fish exposed to wild type donors. The bars are 95% confidence intervals, with is the range of effects that are compatible with the observed data at the 95% level (confidence intervals are discussed in depth in chapter xxx.). The small end of the interval for the sox10/wt comparison is 1.31, meaning that effects as small as 31% increased neutrophil count are compatible with the data. It is up to the research community to decide if 2.7X or 1.31X are physiologically meaningful effects. *p*-values from the hypothesis tests are included.

4.1.2 Pretty good plot component 2: Modeled mean and CI plot

Often the means of the treatment levels are meaningful, for example, if neutrophils per length of gut is a standard measure then researchers working in this area will be familiar with usual and unusual values. The data used in Fig 4.1 are used to plot means and confidence intervals of the mean using a **bar chart**, which is a pretty good chart type for measures such as counts in which negative values are prohibited and zero is meaningful.

Fig. 4.2 plots the *modeled* means, represented by the tops of the bars, the modeled 95% confidence intervals of each mean, represented by the error bars,

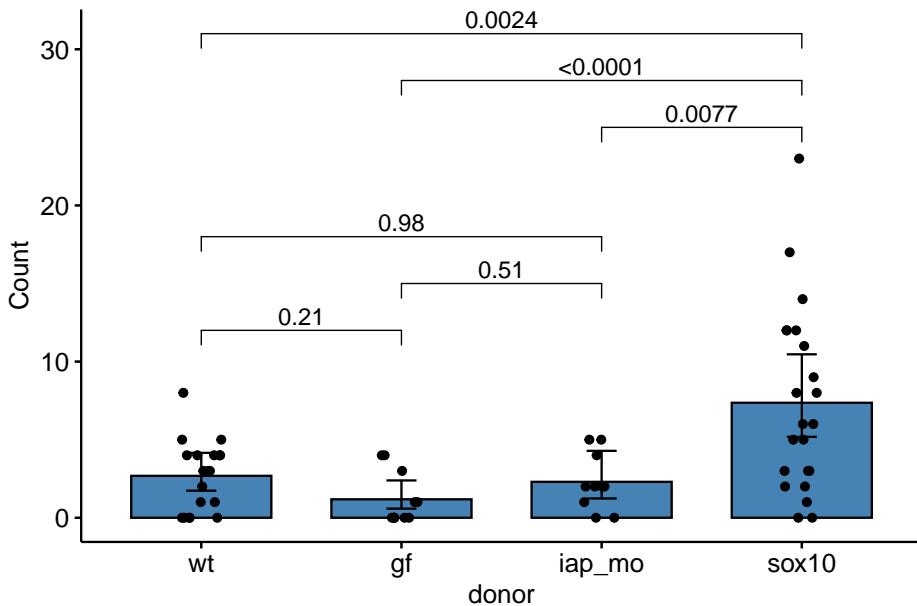


Figure 4.2: Mean and error plot

and the p -values for all pairwise comparisons. What do I mean by *modeled* means and error intervals?

1. Modeled means and error intervals are estimated from the statistical model. Many published plots are of raw means and error intervals, meaning that the mean and error for each treatment level is computed only using the response measures in that treatment level.
2. A modeled mean will often be equal to the raw mean, but this will not always be the case, for example if there are covariates in the model (Chapter xxx).
3. Modeled error intervals are never the same as the raw error intervals, and are commonly conspicuously different. Almost always, we should plot modeled means and error intervals, since these represent the statistics that are relevant to inference.

Fig. 4.2 also plots the raw count data as “jittered” black dots. “Showing the data” is a pretty good feature of a plot because it allows the reader to get a sense of the underlying sample size and distribution including outliers, which can be used to mentally model check the published statistical analysis. For example, the jittered dots in Fig. 4.2 suggest a **heterogeneity** of variances; specifically, the treatment level with the largest mean has a conspicuously higher variance. This pattern violates the assumptions of a general linear model and should raise

a red flag to a reader if the researchers used a general linear model to analyze the data.

What a mean-and-error plot fails to show, at least directly, are the effects. To infer the effects from the plot, a reader must perform mental math – either compute the difference or the ratio between pairs of means. This mental math is easy enough if the comparisons are between individual treatment levels but much harder if the comparisons are between pooled sets of treatment levels, for example in a factorial experimental design. The mental math that is excessively difficult is the reconstruction of some kind of error interval of the contrasts, for example the 95% confidence intervals in Fig. ?? and it is this interval that is necessary for a researcher to infer the range of biological consequences that are compatible with the experiment. The inclusion of the p -values for all pairwise comparisons gives the significance level of these contrasts, but of the kinds of summary results that we could present (contrasts, error intervals, p -values), the p -values are the least informative.

4.1.3 Combining Effects and Modeled mean and CI plots – an Effects and response plot.

If one wants to show both effects and the data, then these can be combined.

If the means do not have any importance in understanding the results, the effects plot can be combined with some kind of a plot summarizing the distribution, such as a boxplot.

Regardless, the effects plot is the most important component as this is the illustration of the story a researcher wants to tell.

4.2 Some comments on plot components

1. **Alternatives to barplots make good plots for the supplement, not the main paper.** A prominent trend over the last few years has been the replacement of bar plots with plots that “show the data”, such as jitter plots or dot plots, or that show summaries of the distribution, such as box plots or violin plots. These plot types were developed for exploratory data analysis, not to communicate the results of experiments. All of these plots fail to communicate the results of the statistical model and, because of this, are inferior to an effects plot, and even a mean-and-error plot, if the mean and error are the modeled values. Box/Violin/Dot/Jitter plots are a useful supplement to an effects plot, either combined with the effects plot as above, or as a supplementary figure.
2. Standard error bars, computed from the raw data, can have absurd implications. For example, I sometimes see standard error bars cross $y = 0$ for a response that cannot be negative, such as a count. Even if the standard

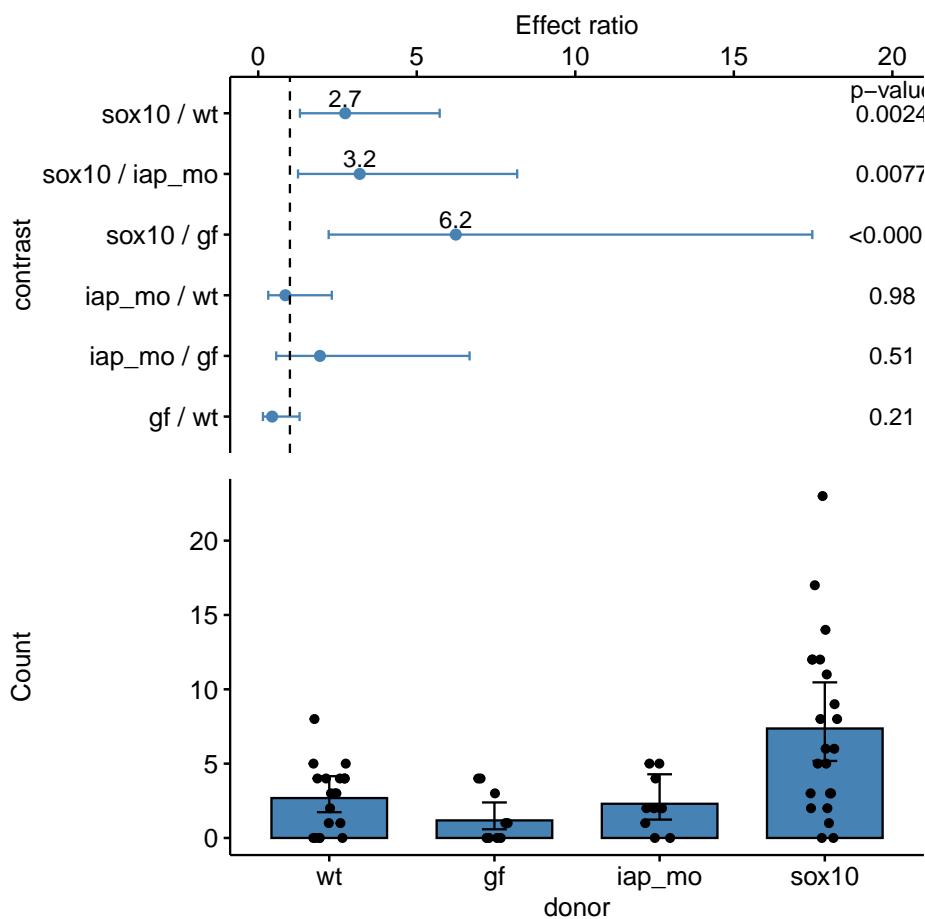


Figure 4.3: A pretty good plot

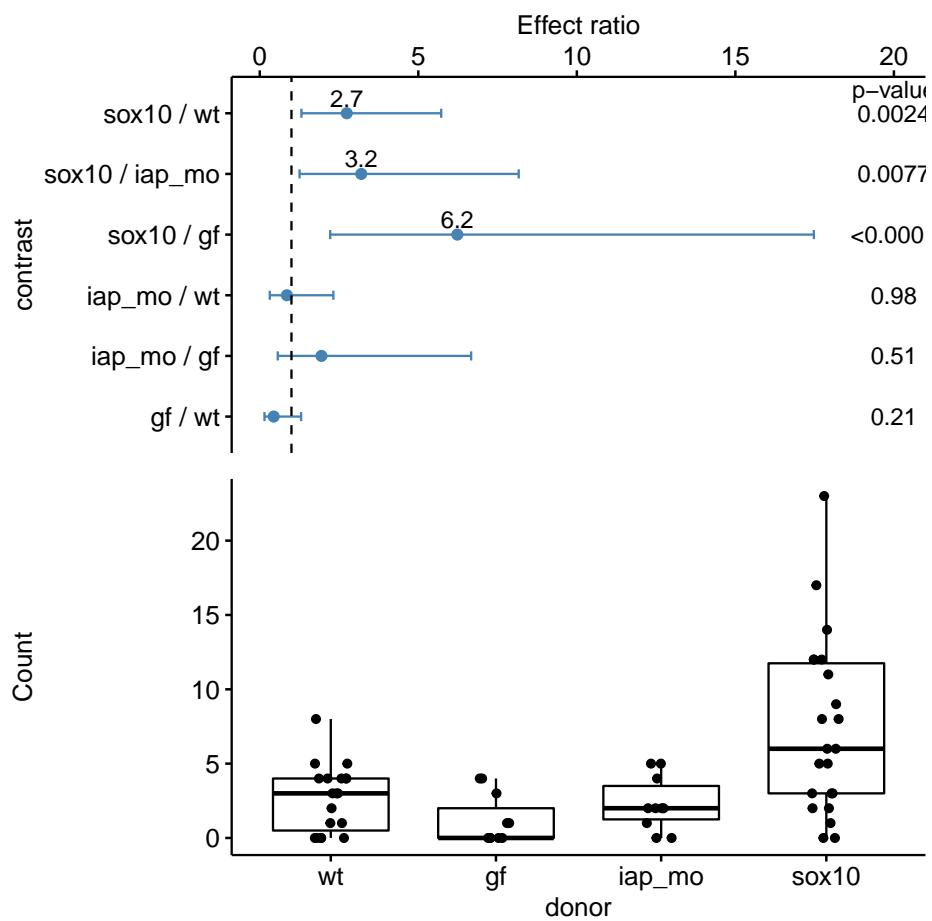


Figure 4.4: Another pretty good plot

error bar doesn't cross zero, it is common to see standard error bars that imply (but do not explicitly show) 95% confidence intervals that cross zero, again for responses that cannot be negative. A standard error bar or confidence interval that crosses zero implies that negative means are compatible with the data. This is an absurd implication for responses that cannot have negative values (or are "bounded by" zero). Explicit or implicit error bars that cross zero are especially common for count responses with small means. *If* a researcher plots confidence intervals, these should be computed using a method that avoids absurd implications, such methods include the bootstrap and generalized linear models.

3. **Stars add minimal value.** Many researchers add star symbols to a plot indicating the level of significance of a particular paired comparison. An uncommon, but better, alternative would be to add the actual p-value (as above). Adding a p-value (or stars) does communicate model results, and so adds value to a mean-and-error or box/violin/jitter plot. However, much more value would be added by simply reporting an effects plot or a combined effects-and-response plot.

4.3 Working in R

A reasonable goal of any research project should be a script to generate the final plots entirely within the R environment and not rely on external drawing software to add finishing features. `ggplot2` is one of the major plotting environments in R and the one that seems to have the strongest following, especially among new R users. `ggplot2` has the ability to generate extremely personalized and finished plots. However, creating a plot with multiple layers (bars, lines, error intervals, raw data points, p-values, text annotations) can often require many hours of googling.

`ggpubr` is an extension to `ggplot2` (it calls `ggplot2` functions under the hood) and provides many canned functions for producing the kinds of ggplots that are published in biological journals. With one line of script, a researcher can generate a publishable plot that is as good or better than many published plot.

Here I show how to add custom (`ggplot2`) features to a `ggpubr` plot

Throughout this book, `ggpubr` is used to create a basic plot and then additional features are added to the basic plot using `ggplot2` functions.

4.3.1 Unpooled SE bars and confidence intervals

`ggplot2` and `ggpubr` default to unpooled error intervals (standard error bars and confidence intervals).

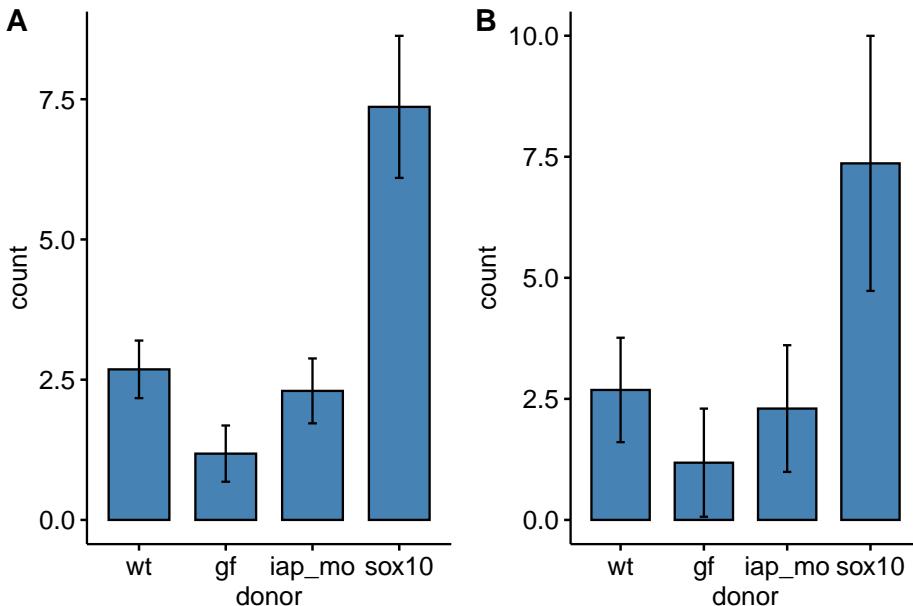


Figure 4.5: (A) Mean and 1 SE error bar. (B) Mean and 95% CI.

```
gg1 <- ggbarplot(data = exp2d,
                   x = "donor",
                   y = "count",
                   add = c("mean_se"),
                   fill = "steelblue")
gg2 <- ggbarplot(data = exp2d,
                   x = "donor",
                   y = "count",
                   add = c("mean_ci"),
                   fill = "steelblue")
plot_grid(gg1, gg2, ncol=2, labels="AUTO")
```

4.3.2 Adding bootstrap intervals

A bootstrap CI uses resamples of the data to estimate the interval and is a better choice than the default CI for data such as counts and proportions. The plot below uses ggpubr to create a stripchart of the data and the color of the data points are “de-emphasized” – in order to emphasize the mean and CI – by making them more transparent (using the argument `alpha`). `alpha` is added

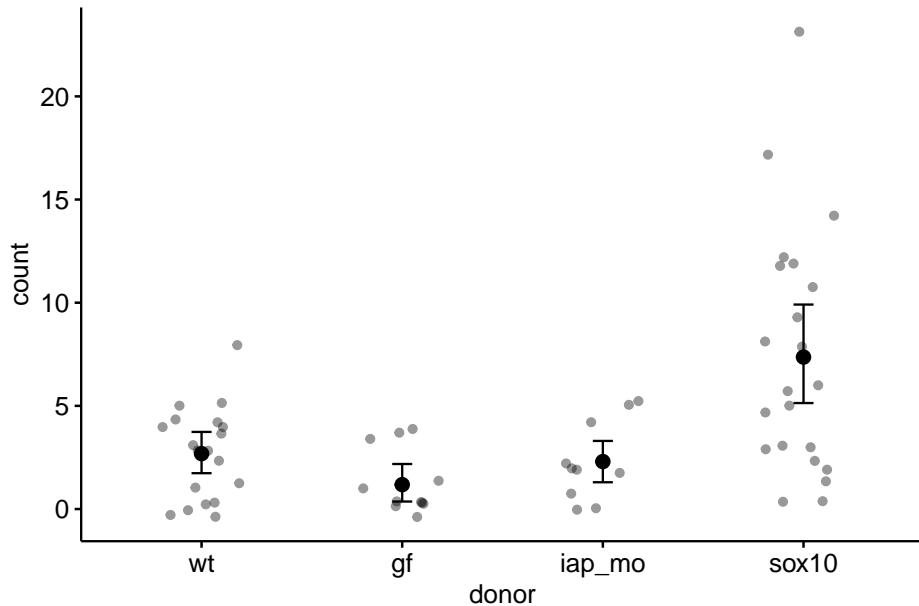


Figure 4.6: Sample means with bootstrapped 95% confidence intervals.

before the argument to add the mean in order to no de-emphasize the mean.

```
set.seed(1)
gg.boot <- ggstripchart(data=exp2d,
  x = "donor",
  y = "count",
  alpha = 0.4,
  add = "mean"
) +
  stat_summary(fun.data = "mean_cl_boot",
  geom = "errorbar",
  width = 0.1) +
NULL
gg.boot
```

4.3.3 Adding modeled means and error intervals

This section is extremely important for implementing the work flow advocated in this text. The goal is to plot the modeled means with some sort of error interval, typically a confidence interval, *and* to show the data or a summary of the data in a single plot. The procedure is

1. fit the model
2. use the fit model to estimate the modeled means and confidence limits using `emmeans` from the `emmeans` package.
3. use the `emmean` object to estimate the contrasts of interests using the `contrast` function from `emmeans`.
4. Use the objects from steps 2 and 3 to plot the modeled means

Step 1: Fit the model. A negative binomial, generalized linear model with log-link is fit to the count data.

```
m1 <- glm.nb(count ~ donor, data=exp2d)
coef(summary(m1))
```

	Estimate	Std. Error	z value	Pr(> z)
## (Intercept)	0.9873867	0.2229971	4.4278004	9.519895e-06
## donorgf	-0.8203326	0.4227008	-1.9406930	5.229553e-02
## donoriap_mo	-0.1544775	0.3878578	-0.3982839	6.904209e-01
## donorsox10	1.0091672	0.2862047	3.5260325	4.218353e-04

- The estimates and SE are on the **link scale**, which means they are in log-transformed space (or “log space”). Exponentiate these with `exp(x)` to **backtransform** these to the the **response scale** which is the scale of the measurement (number of neutrophils).

Step 2: Estimate the modeled means and confidence levels. The second step is to pass the fit model object (`m1`) to `emmeans` to estimate the modeled means.

```
m1.emm <- emmeans(m1, specs="donor", type="response")
m1.emm
```

## donor	response	SE	df	asymp.LCL	asymp.UCL
## wt	2.68	0.599	Inf	1.734	4.16
## gf	1.18	0.424	Inf	0.585	2.39
## iap_mo	2.30	0.730	Inf	1.235	4.28
## sox10	7.36	1.321	Inf	5.181	10.47
##					
## Confidence level used:	0.95				
## Intervals are back-transformed from the log scale					

- We specify the means that we want to estimate with “specs =”. Here, we want to estimate the means of the levels of `donor`.

- Because the linear predictor of the model is on the log scale, we use the “type” argument to specify that we want the means to be backtransformed to the **response scale**, which is the scale of the measurement (number of cells)
- It can be useful to convert the emmeans table `m1.emm` to a `data.table` (or `data.frame` or `tibble`) using `m1.emm <- data.table(m1.emm)`. **Bug alert** If you do this, the object cannot be passed to the next step, the `contrast` function. So if you want the emmeans table as a `data.table`, assign it to a different name, for example `m1.emm_dt <- data.table(m1.emm)`.

Step 3: Compute the contrasts, with p-values and confidence levels.
 Contrasts among levels, or combinations of levels, are computed by passing the `emmeans` object (`m1.emm`) to the `contrast` function.

```
m1.pairs <- contrast(m1.emm, method="revpairwise", adjust="none") %>%
  summary(infer=c(TRUE, TRUE))
m1.pairs

##   contrast      ratio     SE   df asymp.LCL asymp.UCL z.ratio p.value
##  gf / wt      0.440 0.186 Inf    0.192     1.01 -1.941  0.0523
##  iap_mo / wt  0.857 0.332 Inf    0.401     1.83 -0.398  0.6904
##  iap_mo / gf  1.946 0.933 Inf    0.761     4.98  1.389  0.1647
##  sox10 / wt   2.743 0.785 Inf    1.566     4.81  3.526  0.0004
##  sox10 / gf   6.231 2.501 Inf    2.837    13.68  4.558 <.0001
##  sox10 / iap_mo 3.202 1.167 Inf    1.567     6.54  3.192  0.0014
##
## Confidence level used: 0.95
## Intervals are back-transformed from the log scale
## Tests are performed on the log scale
```

- Here, we set “method” to “revpairwise” in order to compute contrasts among all pairs of levels of `donor`. There are $m = 4$ levels and so $m(m - 1)/2 = 6$ pairwise contrasts. “revpairwise” is used instead of “pairwise” because the former sets the direction of the contrasts that include the reference as non-reference level minus reference level.
- I use the “adjust” argument to specify no p -value adjustment for multiple tests.
- the contrast object is then piped (`%>%`) to the `summary` function, where I pass to the argument “`infer`”, that I want both the confidence intervals (the first `TRUE`) and p -values (the second `TRUE`)
- this step isn’t necessary if we were plotting only modeled means and CIs but 1) we almost always want contrasts with a fit model and so that is done here as part of the uninterrupted work flow that this book advocates and 2) we do use the p -values and CIs from this table (`m1.pairs`) in the final plot below.

- **Bug alert** again, the emmeans table m1.emm must be passed to `contrast` as an emmeans object. If you have converted this object to a data.table, you will get an error. See the last note in Step 2.

Step 4: Plot the modeled means and 95% error intervals.

The code below first creates the stripchart using the `ggpubr` function and then adds the confidence intervals using `geom_errorbar` and means using `geom_point`. The stripchart uses the data in the `exp2d` data.table. The errorbar and mean use the values in `m1.emm` object created by the `emmeans` function. The `geom_errorbar` and `geom_point` functions require an “aesthetic” to tell ggplot which column contains the y values of the points to plot (the “x” values are still in the column “donor”, which is a column in both the `exp2d` data.table and `m1.emm`). The name of the column containing the “y” values in `m1.emm` is “response”.

```
set.seed(1)
gg.nb <- ggstripchart(data=exp2d,
                       x="donor",
                       y="count",
                       alpha = 0.4) +
  ylab("Neutrophil count") +
  geom_errorbar(data=summary(m1.emm),
                aes(y=response,
                    ymin=asym.LCL,
                    ymax=asym.UCL),
                width=0.1) +
  geom_point(data=summary(m1.emm),
             aes(y=response),
             size=2) +
  NULL
gg.nb
```

Some notes on the plot code

- A column name passed to a `ggpubr` function must be in quotes but a column name passed to a `ggplot2` function cannot be in quotes
- **Bug alert.** The data passed to `ggplot2` must be a `data.frame`. In order for the `ggplot2` functions to use the `m1.emm` object, the object has to be passed as `summary(m1.emm)`.
- **Bug alert.** Because the `m1.emm` table does not have a column named “count”, which is the “y” column specified in `ggstripchart`, you must supply a new “y” column name to the `aes` function of `geom_errorbar` and `geom_point`. This is the name of the column in the emmeans table containing the modeled means. In `m1.emm`, this name is “response” but it can take different names in different emmeans tables, depending on the fit model.

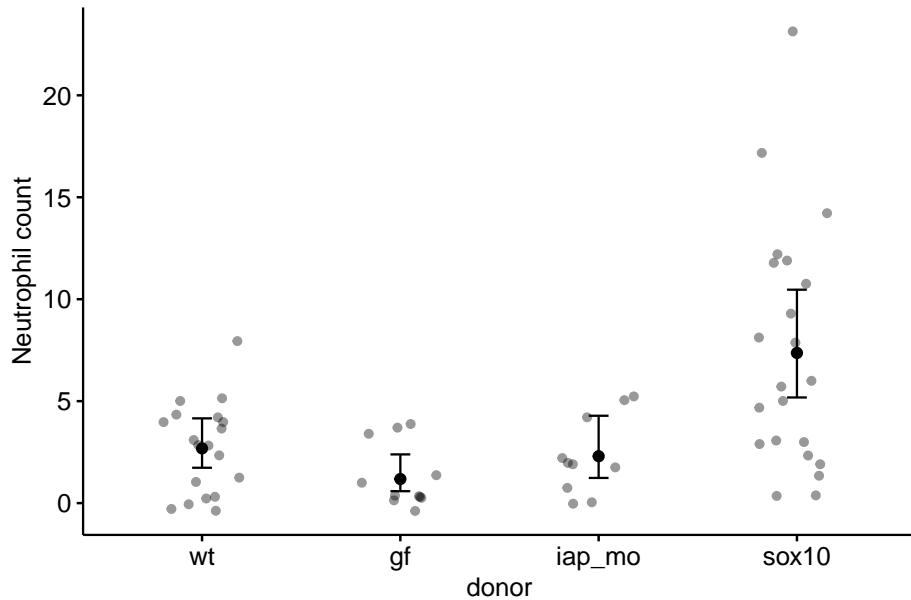


Figure 4.7: Modeled means and 95% confidence interval computed from a negative binomial generalized linear model.

4.3.4 Adding p-values

In this section, I show how to add p -values to a `ggpubr` plot using `stat_compare_means`. Because this function has only a limited set of models that can be used to compute the p -values, I don't find it very useful and instead recommend adding custom p -values from the fit model (or from a permutation test) using the method in the next section.

For this example, a “`t.test`” is used to compute the p -values. The mean and error are the sample-based estimates because these, and not the modeled estimates, are consistent with the `t-test` p -values.

```
compare_list <- list(c("sox10", "iap_mo"), c("sox10", "gf"), c("sox10", "wt"))
gg.sample <- ggstripchart(data=exp2d,
                           x="donor",
                           y="count",
                           alpha = 0.4,
                           add=c("mean_ci")) +
  stat_compare_means(method = "t.test", comparisons=compare_list) +
  ylab("Neutrophil count") +
  NULL
gg.sample
```

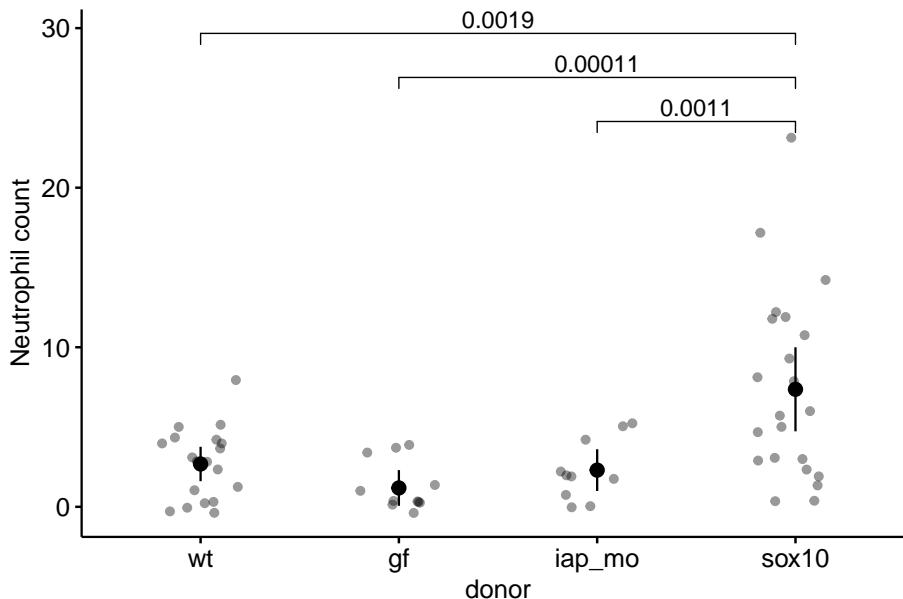


Figure 4.8: t-test p-values for the plot of sample means and CIs. The p-values were computed using ggpubr's function stat_compare_means.

Notes on the code

- The pairs to compare with a p -value are specified with `comparison =`. The order of the pairs in the list function determine the order plotted from bottom (lowest on the y-axis) to top (highest on the y-axis).
- **It is important to know what exactly is being computed when analyzing data and reporting results** and “t test” is not sufficient to know this. The t-test could be the classic t-test or a Welch test. In this example, there are multiple comparisons and the standard error of the test statistic could be the pooled estimate from the linear model, or a pairwise estimate computed separately for each pair. And, given the multiple comparisons, the p-values could be adjusted or not. These kinds of questions can be checked with a function's help page. `?stat_compare_means` doesn't answer these questions but suggests `compare_means`, which also doesn't answer these questions. The script below has checks to see what p-values the function is returning. Run it in your session by changing the value of `check_it` to TRUE.

```
# checks on the p-value
# t-tests using SE pooled over all four groups
check_it <- FALSE
```

```

if(check_it==TRUE){
  m1.lm <- lm(count~donor, data=exp2d)
  m1.lm.emm <- emmeans(m1.lm, specs="donor")
  contrast(m1.lm.emm, method="trt.vs.ctrl", ref=4, adjust="none") # pooled SD

  pairwise.t.test(exp2d$count, exp2d$donor, p.adjust.method="none", pool.sd=FALSE) # n
  # compare
  t.test(count~donor, data=exp2d[donor=="wt" | donor=="sox10"]) # matches, this is Welch's
  t.test(count~donor, data=exp2d[donor=="wt" | donor=="sox10"], var.equal=TRUE)
}

```

So, the p -values returned by `stat_compare_means(method="t.test")` are computed from independent (not pooled over the four groups) Welch t-tests.

4.3.5 Adding custom p-values

If we want to add permutation p -values to the plot with bootstrapped CIs (4.6) or add p -values from the generalized linear model to the plot of modeled means and CIs (4.7), we need to use the function `stat_pvalue_manual` from the `ggpubr` package. In order to implement this, we need to add a step to the work flow path above

Step 5: Add group columns and a column of formatted p-values to the contrast table

The `stat_pvalue_manual` function needs to read a data frame with a columns labeled “group1” and “group2” that contain the pairs of levels to compare with a plotted p -value and a column “ p ” containing the nicely formatted p -values to add to the plot. There is no R function to create this table, but here is a script to add these to the contrast object returned by the `contrast` function of `emmeans`. In this example, I use `m1.pairs` from above and add the p -values to the plot of modeled means and CIs (4.7).

First, we need these functions. Run these two lines to define the functions `odd` and `even`

```

odd <- function(x) x%%2 != 0
even <- function(x) x%%2 == 0

```

Second, we need to use these functions to add the columns. There are several R packages that provide functions to format p -values. Here, I use the function `pvalString` from the `lazyWeave` package. This script also uses `str_split` from the package `stringr`.

```
# convert m1.pairs to a data.table and assign to a new object, in order to
# keep a clean copy of m1.pairs
m1.pvalues <- data.table(m1.pairs)
# if the linear model is from a glm with log link, use this
groups <- unlist(str_split(m1.pvalues$contrast, " / "))
# add the group1 and group 2 columns
m1.pvalues[, group1 := groups[odd(1:length(groups))]]
m1.pvalues[, group2 := groups[even(1:length(groups))]]
# create a column of nicely formatted p-values for display.
m1.pvalues[, p := pvalString(p.value)]
```

Bug alert notes on the script to build the p-value table, if you don't want your code to fail.

- The script to extract the pair of group labels `str_split(m1.pvalues$contrast, " / ")` has to be written so that the characters within the quotes matches the characters separating the groups in the “contrast” column of the contrast table (here, `m1.pairs`). This will typically be either a space-minus-space or a space-slash-space. If the model fit is `lm` and the response is not transformed, then the correct code is `str_split(m1.pvalues$contrast, " - ")`. Regardless, look at the table to check.
- In step 3 above, we took the contrast table object and passed it to the function `summary`, which converts the contrast table object to a `data.frame`. If we had skipped this step, `data.table(m1.pairs)` would fail. Instead, we'd have to use `data.table(summary(m1.pairs))`.

Now we can add the p-value to the `ggplot` object `gg.nb` created above. This is the beauty of a `ggplot` object (including those created by `ggpubr`), we can just keep adding stuff to it.

```
gg.nb <- gg.nb +
  stat_pvalue_manual(m1.pvalues[4:6,], # only show sox effects
                     label = "p",
                     y.position=c(31, 28, 25)) +
  NULL
gg.nb
```

Notes on adding manual *p*-values to the plot:

- The pairs of groups to compare are specified by indexing the rows of `m1.pvalues`. Above, I limit the comparisons to those in rows 4-6. If I wanted to specify non-contiguous rows, I could use something like `m1.pvalues[c(1,3,5),]`, for example.

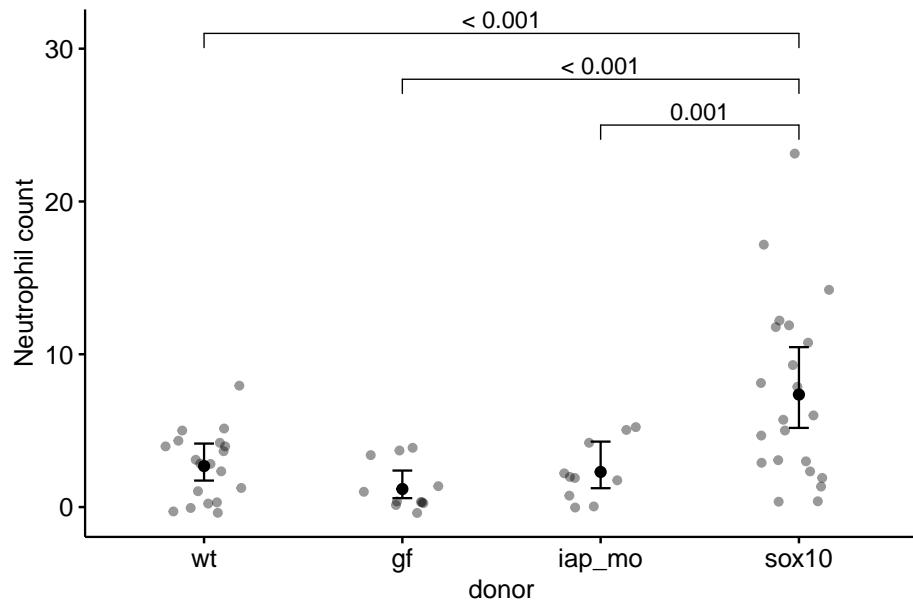


Figure 4.9: Effects and means plot. Top panel: Effects (top panel) of treatments on neutrophil count. Bottom panel: modeled means of treatment levels with 95% confidence intervals.

- The most manual part of adding manual p-values is setting the position for the brackets using the “position” argument. The values in this argument are the y-coordinates of the brackets. This may take some trial-and-error to position the brackets satisfactorily.

4.3.5.1 Modeled error intervals of the effect

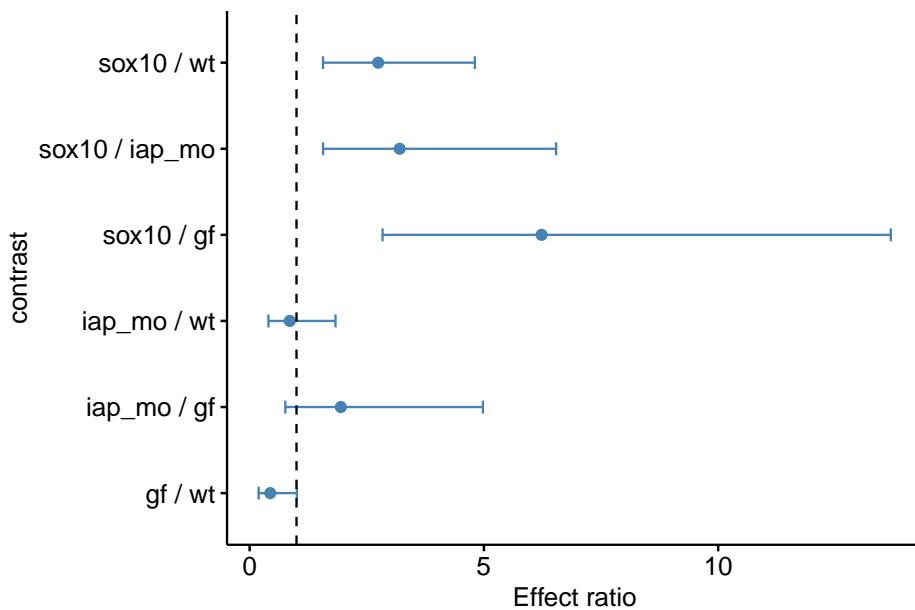
For the plot of effects, we use table of contrasts m1.pairs as the data.

```
gg.effects <- ggdotplot(data = m1.pairs,
                         x="contrast",
                         y="ratio",
                         color = "steelblue",
                         fill = "steelblue",
                         size=0.5) +
  geom_errorbar(aes(x=contrast,
                    ymin=asymp.LCL,
                    ymax=asymp.UCL),
                width=0.15,
                color="steelblue") +
```

```

ylab("Effect ratio") +
geom_hline(yintercept=1, linetype = 2) +
coord_flip() +
NULL
gg.effects

```



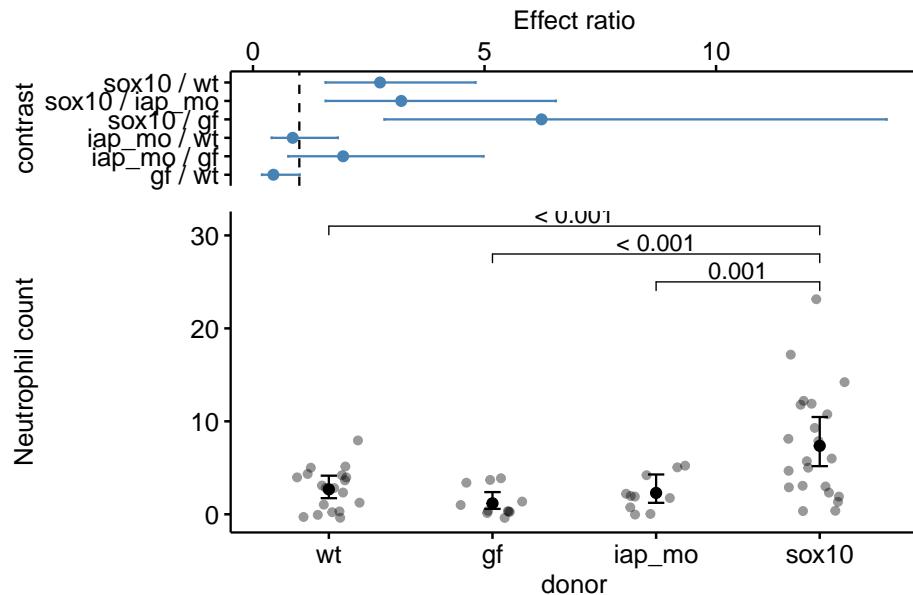
4.3.5.2 Combining effects and response plots

The ggplots are combined using `plot_grid` from the package `cowplot`

```

gg.effects <- gg.effects + scale_y_continuous(position="right")
plot_grid(gg.effects, gg.nb, nrow=2, align = "v", rel_heights = c(1, 2))

```



4.3.6 Plotting two factors

The data are from figure 6d. This solution requires computing either the raw or modeled means and errors and adding these to a base ggpubr plot. Many packages have summary statistics functions for means, standard deviations, and standard errors. This is easily done by simply computing the statistics using `data.table` functionality.

```
# compute raw statistics
# enclosing the line within parentheses prints the result to the console!
(exp6d.raw <- exp6d[!is.na(count), .(count=mean(count),
                                se=sd(count)/sqrt(.N)),
                                by=.(treatment, strain)])
)

##      treatment strain    count        se
## 1:   control     wt 13.08333 2.310904
## 2:   control   sox10 45.61538 6.259903
## 3: transplant     wt 16.35714 2.259552
## 4: transplant   sox10 18.33333 4.536274
```

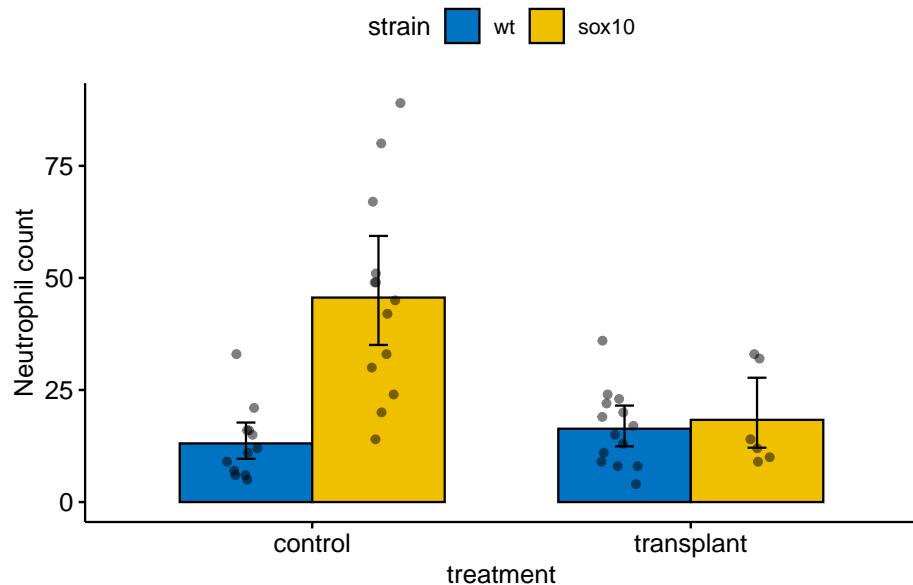
Modeled means, standard errors, and confidence limits are conveniently computed using the `emmeans` (“estimated marginal means”) function from the `emmeans` package.

```
# modeled statsistics
m1 <- glm.nb(count ~ treatment*strain, data=exp6d)
(m1.emm <- data.table(summary(emmeans(m1, specs=c("treatment", "strain")), type="response")))

##      treatment strain response      SE   df asymp.LCL asymp.UCL
## 1:    control     wt 13.08333 2.032161 Inf  9.649528 17.73907
## 2: transplant     wt 16.35714 2.289208 Inf 12.433129 21.51961
## 3:    control  sox10 45.61538 6.132974 Inf 35.048350 59.36837
## 4: transplant  sox10 18.33333 3.871911 Inf 12.119140 27.73391

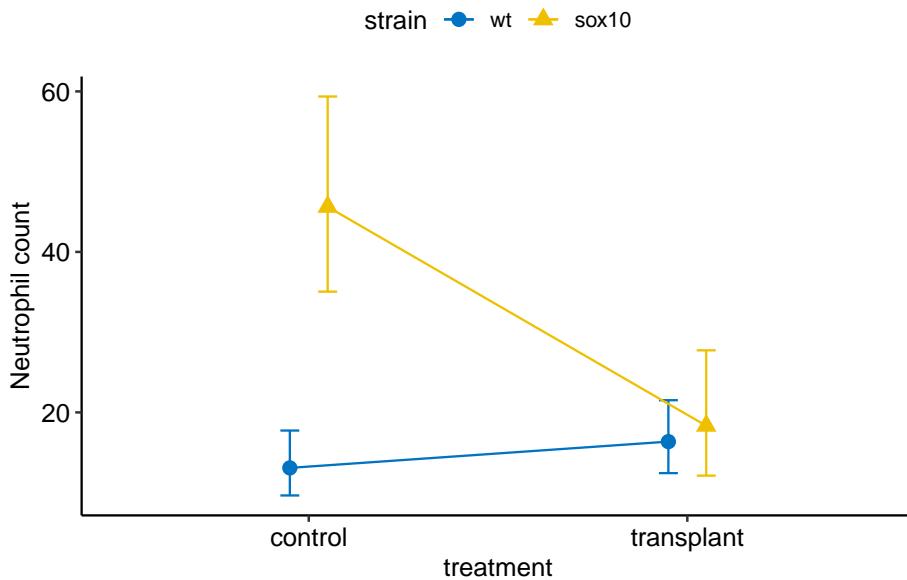
# change column "response" to "count" for the ggplot
setnames(m1.emm, old="response", new="count")

#pairs_i <- list(c("sox10", "iap_mo"), c("sox10", "gf"), c("sox10", "wt"))
pd = position_dodge(0.7)
ggbarplot(x="treatment",
           y="count",
           data=exp6d,
           add=c("mean"),
           color = "black",
           fill = "strain",
           palette = "jco",
           position = pd,
           size=0.5) +
  #stat_compare_means(method = "t.test", comparisons=pairs_i) +
  ylab("Neutrophil count") +
  # geom_dotplot(aes(fill=strain),
  #               binaxis='y', stackdir='center', position=pd, show.legend=FALSE,
  #               color="grey") +
  geom_point(aes(fill=strain), position=position_jitterdodge(jitter.width=0.2), show.legend=FALSE)
  geom_errorbar(data=m1.emm, aes(x=treatment, ymin=asympt.LCL, ymax=asympt.UCL, group=strain),
                position=pd, width=0.1) +
  NULL
```



4.3.7 Interaction plot

```
#pairs_i <- list(c("sox10", "iap_mo"), c("sox10", "gf"), c("sox10", "wt"))
pd = position_dodge(0.2)
ggplot(data=m1.emm, aes(x=treatment, y=count, shape=strain, color=strain, group=strain))
  geom_point(position=pd, size=3) +
  geom_errorbar(data=m1.emm, aes(x=treatment, ymin=asymp.LCL, ymax=asymp.UCL, group=strain))
  geom_line(position=pd) +
  ylab("Neutrophil count") +
  scale_color_jco() +
  theme_pubr() +
  NULL
```

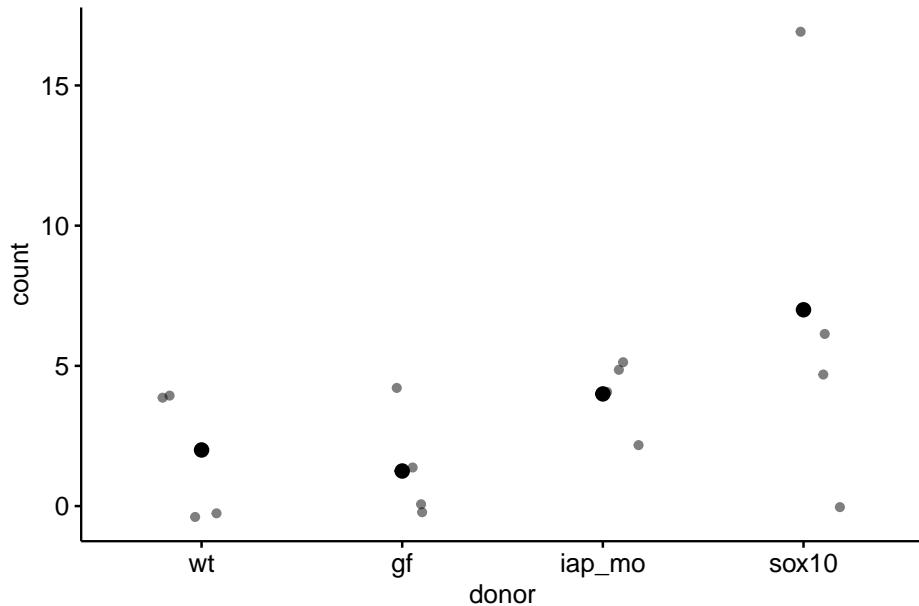


4.3.8 Plot components

4.3.8.1 Showing the data

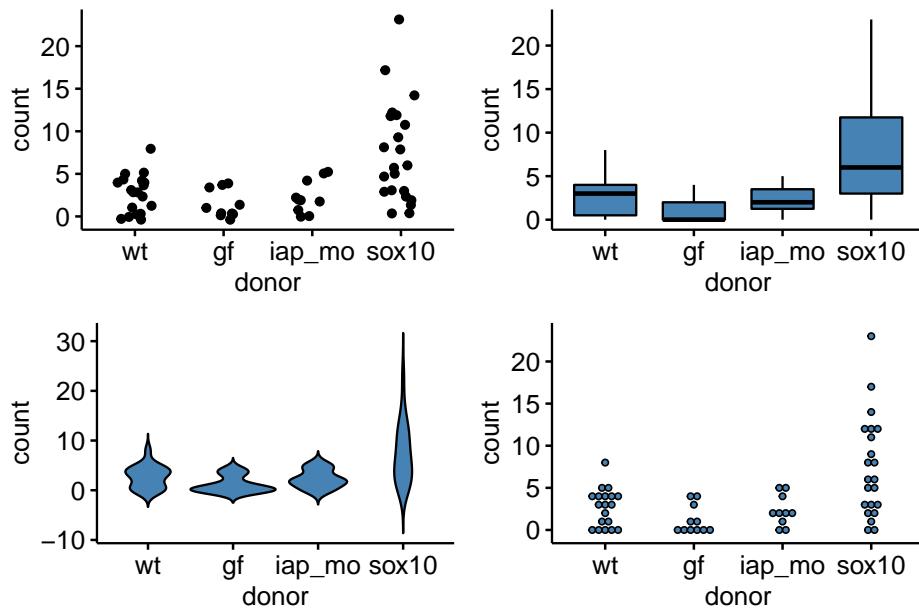
If there are only a few cases per group, there is little reason to summarize the distribution. Instead plot the individual points using a stripchart or a jitter plot

```
# sample 4 points from each group to make it a small n experiment
inc <- exp2d[, .(inc=sample(min(.I):max(.I), 4)), by=donor][, inc]
ggstripchart(x = "donor",
             y = "count",
             alpha = 0.5,
             add = "mean",
             data = exp2d[inc,])
```



With more points, a stripchart can be okay but with too many points the distribution might be obscured. Reasonable alternatives are a box plot, a violin plot, and a dotplot.

```
gg1 <- ggstripchart(x = "donor",
                     y = "count",
                     fill="steelblue",
                     data = exp2d)
gg2 <- ggboxplot(x = "donor",
                  y = "count",
                  fill="steelblue",
                  data = exp2d)
gg3 <- ggy violin(x = "donor",
                   y = "count",
                   fill="steelblue",
                   data = exp2d)
gg4 <- ggdotplot(x = "donor",
                  y = "count",
                  fill="steelblue",
                  data = exp2d)
plot_grid(gg1, gg2, gg3, gg4, nrow=2)
```



Part IV: Some Fundamentals of Statistical Modeling

Chapter 5

Variability and Uncertainty (Standard Deviations, Standard Errors, Confidence Intervals)

Uncertainty is the stuff of science. A major goal of statistics is measuring uncertainty. What do we mean by uncertainty? Uncertainty is the error in estimating a parameter, such as the mean of a sample, or the difference in means between two experimental treatments, or the predicted response given a certain change in conditions. Uncertainty is measured with a **variance** or its square root, which is a **standard deviation**. The standard deviation of a statistic is also (and more commonly) called a **standard error**.

Uncertainty emerges because of variability. In any introductory statistics class, students are introduced to two measures of variability, the “standard deviation” and the “standard error.” These terms are absolutely fundamental to statistics – they are the start of everything else. Yet, many biology researchers confuse these terms and certainly, introductory students do too.

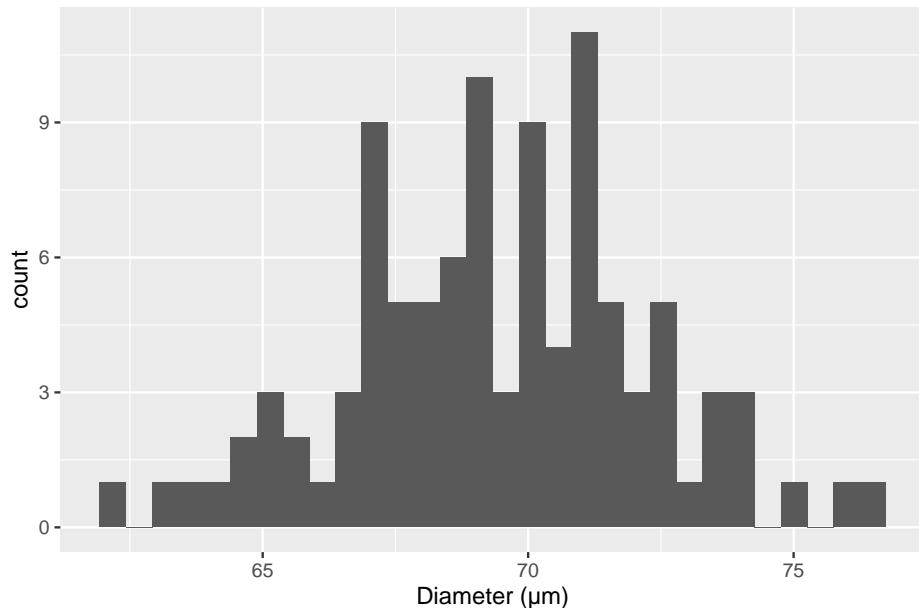
When a research biologist uses the term “standard deviation,” they are probably referring to the sample standard deviation which is a measure of the variability of a sample. When a research biologist uses the term “standard error,” they are probably referring to the standard error of a mean, but it could be the standard error of another statistics, such as a difference between means or a regression slope. An important point to remember and understand is that all standard errors *are* standard deviations. This will make more sense soon.

5.1 The sample standard deviation vs. the standard error of the mean

A standard deviation is the square root of the sampling variance.

5.1.1 Sample standard deviation

The sample standard deviation is a measure of the variability of a sample. For example, were we to look at a histological section of skeletal muscle we would see that the diameter of the fibers (the muscle cells) is variable. We could use imaging software to measure the diameter of a sample of 100 cells and get a **distribution** like this



The mean of this sample is $69.4\mu\text{m}$ and the standard deviation is $2.8\mu\text{m}$. The standard deviation is the square root of the variance, and so computed by

$$s_y = \sqrt{\frac{\sum_{i=1}^n (y_i - \bar{y})^2}{n - 1}} \quad (5.1)$$

Memorize this equation. To understand the logic of this measure of variability, note that $y_i - \bar{y}$ is the **deviation** of the i th value from the sample mean, so the numerator is the sum of squared deviations. The numerator is a sum over n items and the denominator is $n - 1$ so the variance is (almost!) an averaged squared deviation. More variable samples will have bigger deviations

and, therefore, bigger average squared deviations. Since the standard deviation is the square root of the variance, a standard deviation is the square root of an average squared deviation. This makes it similar in value to the averaged deviation (or average of the absolute values of the deviations since the average deviation is, by definition of a mean, zero).

5.1.1.1 Notes on the variance and standard deviation

1. Variances are additive but standard deviations are not. This means that the variance of the sum of two independent (uncorrelated) random variables is simply the sum of the variances of each of the variables. This is important for many statistical analyses.
2. The units of variance are the square of the original units, which is awkward for interpretation. The units of a standard deviation is the same as that of the original variable, and so is much easier to interpret.
3. For variables that are approximately normally distributed, we can map the standard deviation to the quantiles of the distribution. For example, 68% of the values are within one standard deviation of the mean, 95% of the values are within two standard deviations, and 99% of the values are within three standard deviations.

5.1.2 Standard error of the mean

A standard error of a statistic is a measure of the precision of the statistic. The standard error of the mean is a measure of the precision of the estimate of the mean. The standard error of a difference in means is a measure of the precision of the estimate of the difference in means. The smaller the standard error, the more precise the estimate. The standard error of the mean (SEM) is computed as

$$SEM = \frac{s_y}{\sqrt{n}} \quad (5.2)$$

The SEM is often denoted $s_{\bar{y}}$ to indicate that it is a standard deviation of the mean (\bar{y}).

5.1.2.1 The standard error of the mean can be thought of as a standard deviation of an infinitely long column of re-sampled means

In what sense is a standard error a standard deviation? This is kinda weird. If we sample 100 cells in the slide of muscle tissue and compute the mean diameter, how can the mean have a standard deviation? There is only one value!

To understand how the SEM is a standard deviation, imagine that we sample n values from $N(\mu, \sigma^2)$ (a normal distribution with mean μ and variance σ^2). The mean of our sample is an estimate of μ the standard deviation of sample is an estimate of σ) an infinite number of times and each time, we write down the mean of the new sample. The standard deviation of this infinitely long column of means is the standard error of the mean. Our observed SEM is an estimate of this true value because our observed standard deviation is an estimate of σ .

5.1.2.2 A standard deviation can be computed for any statistic – these are all standard errors.

The SEM is only one kind of standard error. A standard deviation can be computed for any statistic – these are all standard errors. For some statistics, such as the mean, the standard error can be computed directly using an equation, such as that for the SEM (equation (5.2)). For other statistics, a computer intensive method known as the **bootstrap** is necessary to compute a standard error. We will return to the bootstrap in Section 5.4.

5.1.2.3 Notes on standard errors

1. The units of a standard error are the units of the measured variable.
2. A standard error is proportional to sample variability (the sample standard deviation, s_y) and inversely proportional to sample size (n). Sample variability is a function of both natural variation (there really is variation in diameter among fibers in the quadriceps muscle) and measurement error (imaging software with higher resolution can measure a diameter with less error). Since the SEM is a measure of the precision of estimating a mean, this means this precision will increase (or the SEM will decrease) if 1) an investigator uses methods that reduce measurement error and 2) an investigator computes the mean from a larger sample.
3. This last point (the SEM decreases with sample size) seems obvious when looking at equation (5.2), since n is in the denominator. Of course n is also in the denominator of equation (5.1) for the sample standard deviation but the standard deviation does not decrease as sample size increases. First this wouldn't make any sense – variability is variability. A sample of 10,000 cell diameters should be no more variable than a sample of 100 cell diameters (think about if you agree with this or not). Second, this should also be obvious from equation (5.1). The standard deviation is the square root of an average and averages don't increase with the number of things summed since both the numerator (a sum) and denominator increase with n .

5.2 Using Google Sheets to generate fake data to explore the standard error

In statistics we are interested in estimated parameters of a **population** using measures from a **sample**. The goal in this section is to use Google Sheets (or Microsoft Excel) to use fake data to discover the behavior of sampling and to gain some intuition about uncertainty using standard errors.

5.2.1 Steps

1. Open Google Sheets
2. In cell A1 type “mu”. mu is the greek letter μ and is very common notation for the population value (the TRUE value!) of the mean of some hypothetical measure. In cell B1, insert some number as the value of μ . Any number! It can be negative or positive.
3. In cell A2 type “sigma”. sigma is the greek letter σ . σ^2 is very common (universal!) notation for the population (TRUE) variance of some measure or parameter. Notice that the true (population) values of the mean and variance are greek letters. This is pretty standard in statistics. In cell B2, insert some positive number (standard deviations are the positive square roots of the variance).
4. In cell A8 type the number 1
5. In cell A9 insert the equation “=A8 + 1”. What is this equation doing? It is adding the number 1 to the value in the cell above, so the resulting value should be 2.
6. In Cell B8, insert the equation “=normsinv(rand())*\$B\$2 + \$B\$1”. The first part of the equation creates a random normal variable with mean 0 and standard deviation 1. multiplication and addition transform this to a random normal variable with mean μ and standard deviation σ (the values you set in cells B1 and B2).
7. copy cell B8 and paste into cell B9. Now Higlight cells A9:B9 and copy the equations down to row 107. You now have 100 random variables sampled from a infinite population with mean μ and standard deviation σ .
8. In cell A4 write “mean 10”. In cell B4 insert the equation “=average(B8:B17)”. The resulting value is the **sample mean** of the first 10 random variables you created. Is the mean close to μ ?
9. In cell A5 write “sd 10”. In cell B5 insert the equation “=stdev(B8:B17)”. The result is the **sample standard deviation** of the first 10 random variables. Is this close to σ ?
10. In cell A6 write “mean 100”. In cell B6 insert the equation “=average(B8:B107)”. The resulting value is the **sample mean** of the all 100 random variables you created. Is this mean closer to μ than mean 10?
11. In cell A7 write “sd 100”. In cell B7 insert the equation “=stdev(B8:B107)”. The resulting value is the **sample standard deviation** of the all 100

random variables you created. Is this SD closer to σ than sd 10?

The sample standard deviation is a measure of the variability of the sample. The more spread out the sample (the further each value is from the mean), the bigger the sample standard deviation. The sample standard deviation is most often simply known as “The” standard deviation, which is a bit misleading since there are many kinds of standard deviations!

Remember that your computed mean and standard deviations are estimates computed from a sample. They are estimates of the true values μ and σ . Explore the behavior of the sample mean and standard deviation by re-calculating the spreadsheet. In Excel, a spreadsheet is re-calculated by simultaneously pressing the command and equal key. In Google, command-R recalculates but is painfully slow. Instead, if using Google Sheets, just type the number 1 into a blank cell, and the sheet recalculates quickly. Do it again. And again.

Each time you re-calculate, a new set of random numbers are generated and the new means and standard deviations are computed. Compare mean 10 and mean 100 each re-calculation. Notice that these estimates are variable. They change with each re-calculation. How variable is mean 10 compared to mean 100? The variability of the estimate of the mean is a measure of **uncertainty** in the estimate. Are we more uncertain with mean 10 or with mean 100? This variability is measured by a standard deviation. This **standard deviation of the mean** is also called the **standard error of the mean**. Many researchers are loose with terms and use “The” standard error to mean the standard error of the mean, even though there are many kinds of standard errors. In general, “standard error” is abbreviated as “SE.” Sometimes “standard error of the mean” is specifically abbreviated as “SEM.”

The standard error of the mean is a measure of the precision in estimating the mean. The smaller the value the more precise the estimate. The standard error of the mean *is* a standard deviation of the mean. This is kinda weird. If we sample a population one time and compute a mean, how can the mean have a standard deviation? There is only one value! And we compute this value using the sample standard deviation: $SEM = \frac{SD}{\sqrt{N}}$. To understand how the SEM is a standard deviation, Imagine recalculating the spread sheet an infinite number of times and each time, you write down the newly computed mean. The standard error of the mean is the standard deviation of this infinitely long column of means.

5.3 Using R to generate fake data to explore the standard error

note that I use “standard deviation” to refer to the sample standard deviation and “standard error” to refer to the standard error of the mean (again, we can

compute standard errors as a standard deviation of any kind of estimate)

5.3.1 part I

In the exercise above, you used Google Sheets to generate p columns of fake data. Each column had n elements, so the matrix of fake data was $n \times m$ (it is standard in most fields to specify a matrix as rows by columns). This is *much* easier to do in R and how much grows exponentially as the size of the matrix grows.

To start, we just generate a $n \times p$ matrix of normal random numbers.

```
# R script to gain some intuition about standard deviation (sd) and standard error (se)
# you will probably need to install ggplot2 using library(ggplot2)
n <- 6 # sample size
p <- 100 # number of columns of fake data to generate
fake_data <- matrix(rnorm(n*p, mean=0, sd=1), nrow=n, ncol=p) # create a matrix
```

the 3rd line is the cool thing about R. In one line I'm creating a dataset with n rows and p columns. Each column is a sample of the standard normal distribution which by definition has mean zero and standard deviation of 1. But, and this is important, any sample from this distribution will not have exactly mean zero and standard deviation of 1, because it's a sample, the mean and standard deviation will have some small error from the truth. The line has two parts to it: first I'm using the function "rnorm" (for random normal) to create a vector of $n \times m$ random, normal deviates (draws from the random normal distribution) and then I'm organizing these into a matrix (using the function "matrix")

To compute the vector of means, standard deviations, and standard errors for each column of `fake_data`, use the `apply()` function.

```
means <- apply(fake_data, 2, mean) # the apply function is super useful
sds <- apply(fake_data, 2, sd)
sems <- sds/sqrt(n)
```

`apply()` is a workhorse in many R scripts and is often used in R scripts in place of a for-loop (see below) because it takes fewer lines of code.

The SEM is the standard deviation of the mean, so let's see if the standard deviation of the means is close to the true standard error. We sampled from a normal distribution with SD=1 so the true standard is

```
1/sqrt(n)
```

```
## [1] 0.4082483
```

and the standard deviation of the p means is

```
sd(means)
```

```
## [1] 0.3731974
```

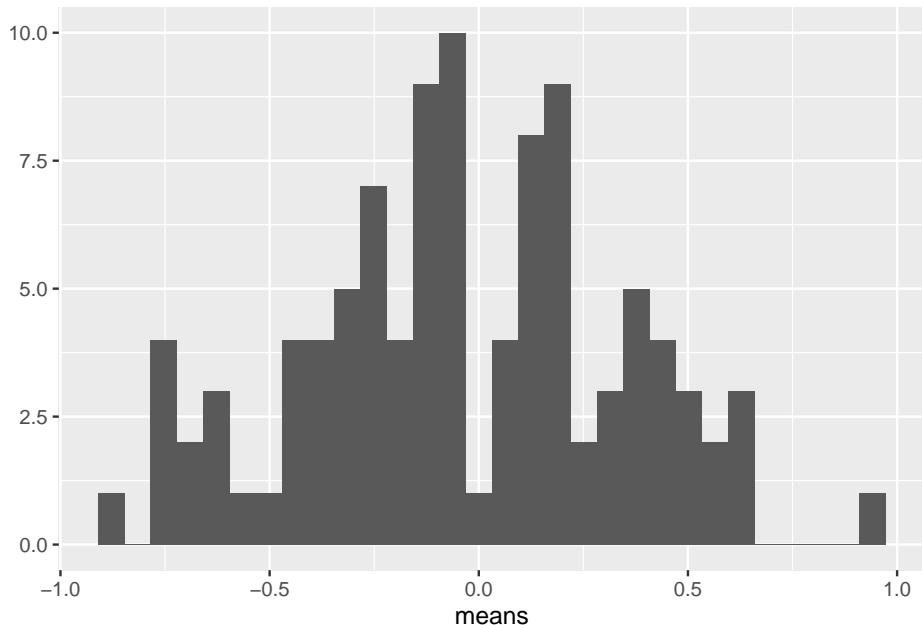
Questions

1. how close is `sd(means)` to the true SE?
2. change p above to 1000. Now how close is `sd(means)` to the true SE?
3. change p above to 10,000. Now how close is `sd(means)` to the true SE?

5.3.2 part II - means

This is a visualization of the spread, or variability, of the sampled means

```
qplot(means)
```



Compute the mean of the means

```
mean(means)
```

```
## [1] -0.039961
```

Questions

1. Remember that the true mean is zero. How close, in general, are the sampled means to the true mean. How variable are the means? How is this quantified?
2. change n to 100, then replot. Are the means, in general, closer to the true mean? How variable are the means now?
3. Is the mean estimated with $n = 100$ closer to the truth, in general, than the mean estimated with $n = 6$?
4. Redo with $n = 10000$

5.3.3 part III - how do SD and SE change as sample size (n) increases?

```
mean(sds)
```

```
## [1] 1.017144
```

Questions

1. what is the mean of the standard deviations when $n=6$ (set $p=1000$)
2. what is the mean of the standard deviations when $n=100$ (set $p=1000$)
3. when $n = 1000$? (set $p=1000$)
4. when $n = 10000$? (set $p=1000$)
5. how does the mean of the standard deviations change as n increases (does it get smaller? or stay about the same size)
6. repeat the above with SEM

```
mean(sems)
```

```
## [1] 0.4152472
```

Congratulations, you have just done a Monte Carlo simulation!

5.3.4 Part IV – Generating fake data with for-loops

A **for-loop** is used to iterate a computation.

```

n <- 6 # sample size
n_iter <- 10^5 # number of iterations of loop (equivalent to p)
means <- numeric(n_iter)
sds <- numeric(n_iter)
sems <- numeric(n_iter)
for(i in 1:n_iter){
  y <- rnorm(n) # mean=0 and sd=1 are default so not necessary to specify
  means[i] <- mean(y)
  sds[i] <- sd(y)
  sems[i] <- sd(y)/sqrt(n)
}
sd(means)

## [1] 0.4090702

mean(sems)

## [1] 0.3883867

```

Questions

1. What do `sd(means)` and `mean(sems)` converge to as `n_iter` is increased from 100 to 1000 to 10,000?
2. Do they converge to the same number?
3. Should they?
4. What is the correct number?

Question number 4 is asking what is $E(SEM)$, the “expected standard error of the mean”. There is a very easy formula to compute this. What is it?

5.4 Bootstrapped standard errors

The bootstrap is certainly one of the most valuable tools invented in modern statistics. But, it's not only a useful tool for applied statistics, it's a useful tool for understanding statistics. Playing with a parametric bootstrap will almost certainly induce an “aha, so that's what statisticians mean by ...” moment.

To understand the bootstrap, let's review a standard error. A *parametric* standard error of a mean is the *expected* standard deviation of an infinite number of means. A standard error of any statistic is the *expected* standard deviation of that statistic. I highlight *expected* to emphasize that parametric standard errors assume a certain distribution (not necessarily a Normal distribution, although

the equation for the SEM in Equation (5.2) assumes a normal distribution if the standard deviation is computed as in Equation (??)).

A bootstrapped standard error of a statistic is the **empirical** standard deviation of the statistic from a finite number of *samples*. The basic algorithm for a bootstrap is (here “the statistic” is the mean of the sample)

1. sample n values from a probability distribution
2. compute the mean
3. repeat step 1 and 2 many times
4. for a bootstrapped standard error, compute the standard deviation of the set of means saved from each iteration of steps 1 and 2.

The probability distribution can come from two sources:

1. A **parametric bootstrap** uses samples from a parametric probability distribution, such as a Normal distribution or a poisson distribution (remember, these are “parametric” because the distribution is completely described by a set of parameters). A good question is why bother? In general, one would use a parametric bootstrap for a statistic for which there is no formula for the standard error, but the underlying data come from a parametric probability distribution.
2. A **non-parametric** bootstrap uses *resamples from the data*. The data are resampled *with replacement*. “Resample with replacement” means to sample n times from the full set of observed values. If we were to do this manually, we would i) write down each value of the original sample on its own piece of paper and throw all pieces into a hat. ii) pick a paper from the hat, add its value to sample i , and return the paper to the hat. iii) repeat step ii n times, where n is the original sample size. The new sample contains some values multiple times (papers that were picked out of the hat more than once) and is missing some values (papers that were not picked out in any of the n picks). A good question is, why bother? A non-parametric bootstrap assumes no specific *parametric* probability distribution but it does assume the distribution of the observed sample is a good approximation of the true population distribution (in which case, the probability of picking a certain value is a good approximation to the true probability).

5.4.1 An example of bootstrapped standard errors using vole data

Let’s use the vole data to explore the bootstrap and “resampling”. The data are archived at Dryad Repository. Use the script in Section ?? to wrangle the data into a usable format.

1. URL: <https://datadryad.org/resource/doi:10.5061/dryad.31cc4>
2. file: RSBL-2013-0432 vole data.xlsx
3. sheet: COLD VOLES LIFESPAN

The data are the measured lifespans of the short-tailed field vole (*Microtus agrestis*) under three different experimental treatments: vitamin E supplementation, vitamin C supplementation, and control (no vitamin supplementation). Vitamins C and E are antioxidants, which are thought to be protective of basic cell function since they bind to the cell-damaging reactive oxygen species that result from cell metabolism.

Let's compute the standard error of the mean of the control group lifespan using both a parametric and a nonparametric bootstrap. To implement the algorithm above using easy-to-understand code, I'll first extract the set of lifespan values for the control group and assign it to its own variable.

```
control_voles <- vole[treatment=="control", lifespan]
```

[`treatment=="control",]` indexes the rows (that is, returns the row numbers) that satisfy the condition `treatment = "control"`. Or, put another way, it selects the `subset` of rows that contain the value “control” in the column “treatment”. `[, lifespan]` indexes the column labeled “lifespan”. Combined, these two indices extract the values of the column “lifespan” in the subset of rows that contain the value “control” in the column “treatment”. The resulting vector of values is assigned to the variable “control_voles”.

5.4.1.1 parametric bootstrap

```
# we'll use these as parameters for parametric bootstrap
n <- length(control_voles)
mu <- mean(control_voles)
sigma <- sd(control_voles)

n_iter <- 1000 # number of bootstrap iterations, or p
means <- numeric(n_iter) # we will save the means each iteration to this

for(iter in 1:n_iter){ # this line sets up the number of iterations, p
  fake_sample <- rnorm(n, mean=mu, sd=sigma)
  means[iter] <- mean(fake_sample)
}
(se_para_boot <- sd(means))

## [1] 30.49765
```

5.4.1.2 non-parametric bootstrap

```

n_iter <- 1000 # number of bootstrap iterations, or p
means <- numeric(n_iter) # we will save the means each iteration to this
inc <- 1:n # inc indexes the elements to sample. By setting inc to 1:n prior to the loop, the first
for(iter in 1:n_iter){ # this line sets up the number of iterations, p
  means[iter] <- mean(control_voles[inc]) # inc is the set of rows to include in the computation
  inc <- sample(1:n, replace=TRUE) # re-sample for the next iteration
}
(se_np_boot <- sd(means))

## [1] 32.47356

```

The parametric bootstrapped SEM is 30.5. The non-parametric bootstrapped SEM is 32.47. Run these several times to get a sense how much variation there is in the bootstrapped estimate of the SEM given the number of iterations. Compute the **parametric** standard error using equation (5.2) and compare to the bootstrapped values.

5.5 Confidence Interval

Here I introduce a **confidence interval** of a sample mean but the concept is easily generalized to any parameter. The mean of the Control voles is 503.4 and the SE of the mean is 31.61. The SE is used to construct the lower and upper boundary of a “ $1 - \alpha$ ” confidence interval using `lower <- mean(x) + qt(alpha/2, df = n-1)*se(x)` and `upper <- mean(x) + qt(1-(alpha/2), df = n-1)*se(x)`.

```

(lower <- mean(control_voles) + qt(0.05/2, df=(n-1))*sd(control_voles)/sqrt(n))

## [1] 440.0464

(upper <- mean(control_voles) + qt(1 - 0.05/2, df=(n-1))*sd(control_voles)/sqrt(n))

## [1] 566.7393

```

The function `qt` maps a probability to a t -value – this is the opposite of a t test, which maps a t -value to a probability. Sending $\alpha/2$ and $1-\alpha/2$ to `qt` returns the bounds of the confidence interval on a standardized scale. Multiplying these bounds by the standard error of the control vole lifespan pops the bounds onto the scale of the control vole lifespans.

We can check our manual computation with the linear model

```
confint(lm(control_voles ~ 1))

##           2.5 %   97.5 %
## (Intercept) 440.0464 566.7393
```

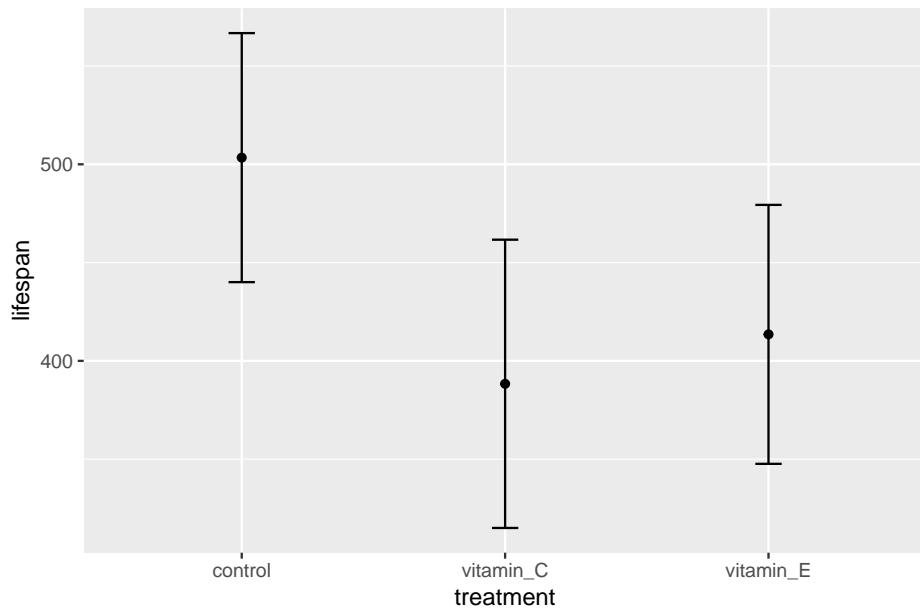
5.5.1 Interpretation of a confidence interval

Okay, so what *is* a confidence interval? A confidence interval of the mean is a measure of the uncertainty in the estimate of the mean. A 95% confidence interval has a 95% probability (in the sense of long-run frequency) of containing the true mean. It is not correct to state that “there is a 95% probability that the true mean lies within the interval”. These sound the same but they are two different probabilities. The first (correct interpretation) is a probability of a procedure – if we re-do this procedure (sample data, compute the mean, and compute a 95% CI), 95% of these CIs will contain the true mean. The second (incorrect interpretation) is a probability that a parameter (μ , the true mean) lies within some range. The second (incorrect) interpretation of the CI is correct only if we also assume that *any* value of the mean is equally probable (Greenland xxx), an assumption that is absurd for almost any data.

Perhaps a more useful interpretation of a confidence interval is, a confidence interval contains the range of true means that are compatible with the data, in the sense that a t -test would not reject the null hypothesis of a difference between the estimate and any value within the interval (this interpretation does not imply anything about the true value) (Greenland xxx). The “compatibility” interpretation is very useful because it implies that values outside of the interval are less compatible with the data.

Let’s look at the confidence intervals of all three vole groups in light of the “compatibility” interpretation.

```
vole_ci <- vole[, .(lifespan = mean(lifespan),
                  lo = mean(lifespan) + sd(lifespan)/sqrt(.N)*qt(.025, (.N-1)),
                  up = mean(lifespan) + sd(lifespan)/sqrt(.N)*qt(.975, (.N-1)),
                  N = .N),
                  by = .(treatment)]
ggplot(data=vole_ci, aes(x=treatment, y=lifespan)) +
  geom_point() +
  geom_errorbar(aes(x=treatment, ymin=lo, ymax=up),
                width=0.1) +
  NULL
```



Chapter 6

P-values

A general perception of a “replication crisis” may thus reflect failure to recognize that statistical tests not only test hypotheses, but countless assumptions and the entire environment in which research takes place. Because of all the uncertain and unknown assumptions that underpin statistical inferences, we should treat inferential statistics as highly unstable local descriptions of relations between assumptions and data, rather than as providing generalizable inferences about hypotheses or models. And that means we should treat statistical results as being much more incomplete and uncertain than is currently the norm.¹

A *p*-value is a measure of the compatibility between observed data and the null model. Here, “compatibility” is a probability, specifically, the probability of sampling a test-statistic as or more extreme than the observed test statistic, if all the assumptions used to compute the *p*-value are true.

To deconstruct what this means, and the implications of the meaning, let’s use the experiment from Figure 2i in the study on the browning of white adipose tissue in mice that was introduced in Chapter 2 (Analyzing experimental data with a linear model).

Data source: ASK1 inhibits browning of white adipose tissue in obesity

A linear model with `liver_tg` as the response variable and `treatment` as the single *X*-variable was fit to the data. Figure ?? is the plot of the modeled means, 95% confidence intervals of the mean, and *p*-value of the significance test of the effect of treatment on liver triacylglycerol.

The coefficients of the model, and the standard error, 95% confidence interval, test-statistic, and *p*-value of each coefficient are shown in Table ?? Recall from Chapter 2 that, with this model, the coefficient for the intercept term is the

¹Amrhein, V., Trafimow, D. and Greenland, S., 2019. Inferential statistics as descriptive statistics: There is no replication crisis if we don’t expect replication. *The American Statistician*, 73(sup1), pp.262-270.

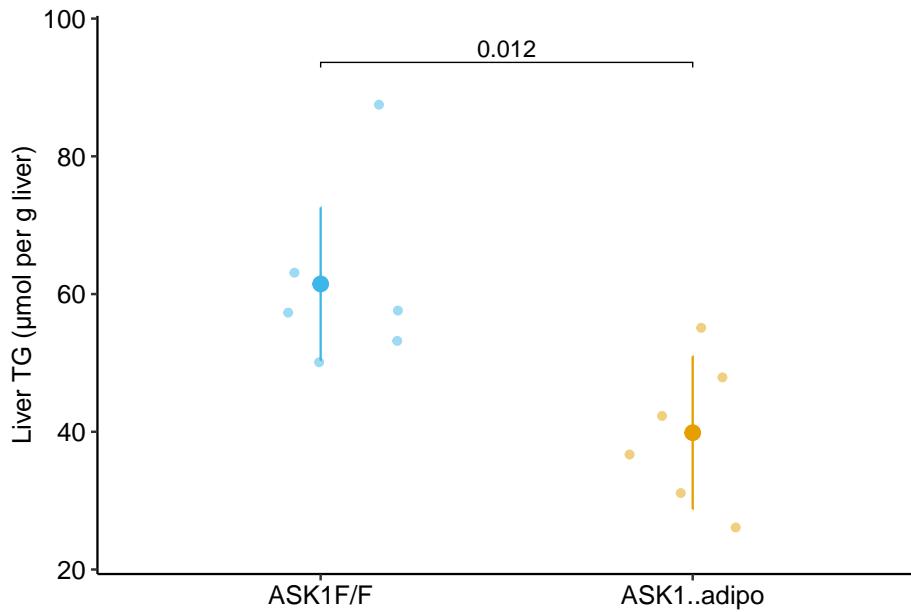


Figure 6.1: UCP1 expression, relative to the mean level in the control group. Mean (circle) and 95% confidence interval (line) are shown. Unadjusted p-values are from linear model with sh-RNA treatment and LPS treatment fully crossed.

mean `liver_tg` for the control group, which is the estimate of the true mean for a mice with functional ASK1 protein. And the coefficient for the treatment`ASK1Δadipo` term is the difference in means between the knockout and control group, which is the estimate for the true effect of knocking out ASK1 in the adipose tissue. The *p*-value for this “effect” term is 0.012. How do we interpret this number?

Estimate

Std. Error

t value

$\Pr(>|t|)$

2.5 %

97.5 %

(Intercept)

61.5

4.98

12.3

6.1. A P-VALUE IS THE PROBABILITY OF SAMPLING A VALUE AS OR MORE EXTREME THAN THE TEST STATISTIC

0.000	
50.4	
72.6	
treatmentASK1Δadipo	
-21.6	
7.05	
-3.1	
0.012	
-37.3	
-5.9	

6.1 A *p*-value is the probability of sampling a value as or more extreme than the test statistic if sampling from a null distribution

The test statistic in the table above is a *t*-value. For this specific model, this *t*-value is precisely the *t*-value one would get if they executed a classical Student *t*-test on the two groups of liver TG values. Importantly, this is not generally true. For many of the models in this text, a *t*-value is computed but this is *not* the *t*-value that would be computed in a classical *t*-test.

When we do a *t*-test, we get a *p*-value. The probability returned in a *t*-test is $p = \text{prob}(t \geq t_{\text{obs}} | H_0)$. Read this as “the *p*-value is the probability of observing a *t*-value that is greater than or equal to the observed *t*-value, given the null model is true.” Probability, in this text, is a long run frequency of sampling. The specific probability associated with the effect of treatment on liver TG is the long-run frequency of observing a *t*-value as big or bigger than the observed *t*-value (the one you actually got with your data) if the null is true. Let’s parse this into “long run frequency of observing a *t*-value as big or bigger than the observed *t*-value” and “null is true”.

A thought experiment: You open a google sheet and insert 12 standard, normal random deviates (so the true mean is zero and the true variance is one) in Column A, rows 1-12. You arbitrarily assign the first six values (rows 1-6) to treatment A and the second six values (rows 7-12) to treatment B. You use the space immediately below these data to compute the mean of treatment A, the mean of treatment B, the difference in means (A - B), and a *t*-value. Unfortunately, google sheets doesn’t have a *t*-value function so you’d have to compute this yourself. Or not, since this is a thought experiment. Now “fill right” or copy and paste these functions into 999 new columns. You now have

1000 t -tests. The expected value of the difference in means is zero (why?) but the actual values will form a normal distribution about zero. Most will be close to zero (either in the negative or positive direction) but some will be further from zero. The expected t -value will also be zero (why?) and the distribution of these 1000 t -values will look normal but the tails are a little fuller. This row of t -values is a null distribution, because in generating the data we used the exact same formula for the values assigned to A and the values assigned to B. Now think of a t -value in your head, say 0.72 (remember that t -values will largely range from about -3 to +3 although the theoretical range is $-\infty$ to $+\infty$). What is the probability of observing a t of 0.72 *or bigger* if the null is true? Look at the row of t -values! Count the number of $t \geq 0.72$ and then divide by the total number of t -values in the row (1000) and you have a probability computed as a frequency. But remember the frequentist definition is the long run frequency, or the expected frequency at the limit (when you've generated not 1000 or even 1,000,000 but an infinite number of columns and t -values).

Some asides to the thought experiment: First, why “as big or bigger” and not just the probability of the value itself? The reason is that the probability of finding the exact t is 1/infinity, which doesn’t do us much good. So instead we compute the probability of finding t as big, or bigger, than our observed t . Second, the t -test probability described above is a “one-tail probability”. Because a difference can be both in the positive direction and the negative direction, we usually want to count all the $t \geq 0.72$ and the $t \leq -0.72$ and then add these two counts to compute the frequency of *as extreme or more extreme* values. This is called a “two-tailed probability” because we find extremes at both tails of the distribution. Third, we don’t really count $t \geq 0.72$ but take advantage of the beautiful mathematical properties of the theoretical t distribution, which allows us to compute the frequentist probability (expected long range frequency) given the t -value and the degrees of freedom using the t -distribution.

Now what do I mean with the phrase “null is true”? Most people equate “null is true” with “no difference in means” but the phrase entails much more than this. Effectively, the phrase is short for “all assumptions used to compute” the p -value (see the Sander Greenland quote at the start of this chapter). A p -value is based on modeling the real data with a theoretical sample in which all the values were randomly sampled from the same distribution and the assignment of the individual values to treatment was random. Random sampling from the same distribution has three important consequences. First, random assignment to treatment group means that the expected means of each group are the same, or put differently, the expected difference in means between the assigned groups is zero. Second, random assignment to treatment *also* means that the expected variances of the two groups are equal. And third, random sampling means that the values of each point are independent – we cannot predict the value of one point knowing information about any other point. **Here is what is super important about this:** a low p -value is more likely if *any* one of these consequences is untrue about our data. A low p -value could arise from a difference in true means, or it could arise from a difference in true variances, or

it could arise if the Y values are not independent of each other. This is why we need certain assumptions to make a p -value meaningful for empirical data. By assuming independent error and homogenous (equal) variances in our two samples, a low p -value is evidence of unequal means.

Let's summarize: A pretty good definition of a p -value is: the long-run frequency of observing a test-statistic as large or larger than the observed statistic, if the null were true. A more succinct way to state this is

$$p = \text{prob}(t \geq t_o | H_o) \quad (6.1)$$

where t is a hypothetically sampled t -value from a null distribution, t_o is the observed t -value, and H_o is the null hypothesis. Part of the null hypothesis is the expected value of the parameter estimated is usually (but not always) zero – this can be called the nil null. For example, if there is no ASK1 deletion effect on liver TG levels, then the expected difference between the means of the control and knockout mice is zero. Or,

$$\mathbb{E}(\bar{Y}_{knockout} - \bar{Y}_{control} | H_o) = 0.0 \quad (6.2)$$

6.2 Pump your intuition – Creating a null distribution

The mean `liver_tg` in the knockout treatment is 21.6 μmol less than the mean `liver_tg` in the control treatment. This is the measured effect, or the **observed differences in means**. How confident are we in this effect? Certainly, if the researchers did the experiment with *two* control (ASK1F/F) treatment groups, they would measure some difference in their means simply because of finite sampling (more specifically, the many, many random effects that contribute to liver TG levels will differ between the two control groups). So let's reframe the question: are the observed differences unusually large compared to a distribution of differences that would occur if there were no effect? That is, if the “null were true”. To answer this, we compare our observed difference to this **null distribution**. This comparison gives the probability (a long-run frequency) of “sampling” a random difference from the null distribution of differences that is as large, or larger, than the observed difference.

What is a null distribution? It is the distribution of a statistic (such as a difference in means, or better, a t -value) if the null were true. Here, I hope to pump your intuition by generating a null distribution that is relevant to the ASK1 liver TG data. See if you can understand the script before reading the explanation below.

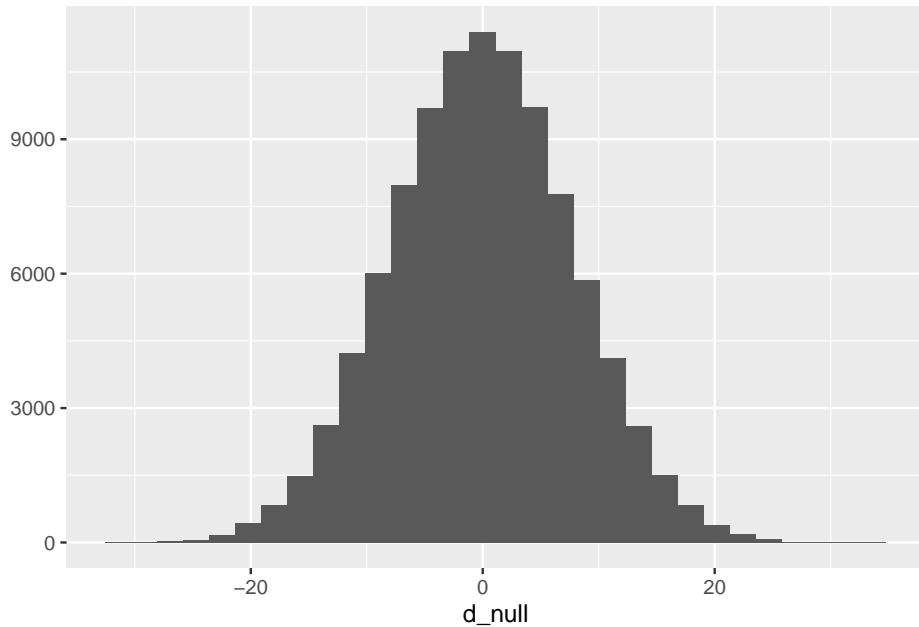


Figure 6.2: Null distribution for the difference in means of two samples from the same, infinitely large population with a true mean and standard deviation equal to the observed mean and standard deviation of the ASK1 liver TG data.

```
seed <- 1
n_iter <- 10^5 # number of iterations

mu <- mean(fig_2i[treatment == "ASK1F/F", liver_tg])
sigma <- sd(fig_2i[treatment == "ASK1F/F", liver_tg])

n <- nrow((fig_2i[treatment == "ASK1F/F",]))

sample1 <- matrix(rnorm(n*n_iter, mean=mu, sd=sigma), nrow=n) # 100,000 samples (each .)
sample2 <- matrix(rnorm(n*n_iter, mean=mu, sd=sigma), nrow=n) # 100,000 samples

d_null <- apply(sample2, 2, mean) - apply(sample1, 2, mean)

qplot(d_null)
```

What have we done above? We've simulated an infinitely large population of mice that have a distribution of liver TG levels similar to that of the mice assigned to the control (ASK1F/F) group. The true mean (μ) and standard deviation (σ) of the simulated TG level are equal to the observed mean and standard deviation of the TG levels of the control mice Then, the script:

1. randomly sample 6 values from this population of simulated lifespans and assign to sample1. We sample 6 values because that is the sample size of our control in the experiment.
2. randomly sample 6 values from this population of simulated lifespans and assign to sample2.
3. compute the difference $\bar{Y}_{sample2} - \bar{Y}_{sample1}$.
4. repeat 1-3 100,000 times, each time saving the difference in means.
5. plot the distribution of the 100,000 differences using a histogram

The distribution of the differences is a null distribution. Notice that the mode of the null distribution is at zero, and the mean (-0.03607) is close to zero (if we had set n to infinity, the mean would be precisely zero). *The expected difference between the means of two random samples from the same population is, of course, zero.* Don't gloss over this statement if that is not obvious. The tails extend out to a little more than +20 and -20. What this means is that it would be uncommon to randomly sample a value from this distribution of differences *as or more extreme* than our observed difference, -21.6. By "more extreme", I mean any value more negative than -21.6 or more positive than 21.6. So it would be uncommon to sample a value from this distribution whose *absolute value* is as or more extreme than 21.6. How uncommon would this be?

```
diff_obs <- fig_2i_m1_coef["treatmentASK1Δadipo", "Estimate"]
null_diff_extreme <- which(abs(d_null) > abs(diff_obs))
n_extreme <- length(null_diff_extreme)
(p_d_null = n_extreme/n_iter)

## [1] 0.00519
```

In the 100,000 runs, only 519 generated data with an absolute difference as large or larger than 21.6 (an "absolute difference" is the absolute value of the difference). The frequency of differences as large or larger than our observed difference is 0.00519. This frequency is the *probability* of sampling a difference as or more extreme than the observed difference "under the null". It is a p -value, but it is not the p -value in the coefficient table. This is because the p -value in the coefficient table is computed from a distribution of t -values, not raw differences. This raises the question, what is a t -distribution, and a t -value, more generally?

6.3 A null distribution of t -values – the t distribution

A t -test is a test of differences between two values. These could be

1. the difference between the means of two samples (a "two-sample" t -test)

2. the difference between a mean of a sample and some pre-specified value (a “one-sample” t -test)
3. the difference between a coefficient from a linear model and zero

A t -test compares an observed t -value to a t -distribution. The null distribution introduced above was a distribution of mean differences under the null. A distribution of mean differences under the null is very specific to the mean and standard deviation of the population modeled and the sample size of the experiment. This isn’t generally useful, since it will be unique to every study (at least it wasn’t generally useful prior to the time of fast computers. One could, and some statisticians do, compute p -values using the algorithm above). A t -distribution is a way of transforming a null distribution of mean differences, which is unique to the study, into a distribution that is a function of sample size only.

A t -distribution is a distribution of t -values under the null, where a t -value is a difference *standardized by its standard error*. For a two-sample t -test, this is

$$t = \frac{\bar{y}_2 - \bar{y}_1}{SE_{\bar{y}_2 - \bar{y}_1}} \quad (6.3)$$

The numerator is **the effect** while the denominator is the precision of the estimate. Like many test statistics, a t -value is a signal-to-noise ratio – the effect is the signal and the SE of the difference is the noise.

A t distribution looks like a standard, normal distribution, except the tails are heavy, meaning there are more large-ish values than the normal. Like the standard normal distribution, large t -values are unlikely under the null and, therefore, a large t has a low probability – or p -value – under the null.

Looking at the equation for the two-sample t -test above, it is easy to see that three features of an experiment are associated with large t and small p -values: 1) big effect size (the numerator of the equation), 2) small sample standard deviations (which results in small standard errors of the difference, the denominator of equation (6.3), and 3) large sample size (which results in small standard errors of the difference). As a quick-and-dirty generalization, absolute t -values greater than 3 are uncommon if the null is true.

The p -value for a t -test comes from comparing the observed t to a null t distribution and “counting” the values that are more extreme than the observed t . The p -value is the relative frequency of these more extreme values (relative to the total number of t -values in the distribution). I have “counting” in quotes because nothing is really counted – there are an infinite number of t -values in the t -distribution. Instead, the t -distribution function is integrated to compute the fraction of the total area under the curve with t -values more extreme than the observed value. In a **two-tailed test**, this fraction includes both tails (positive t -values more positive than $|t|_{observed}$ and negative t -values more negative than $-|t|_{observed}$.

Let's modify the simulation of a null distribution of mean differences to generate a null distribution of t -values. I show the script, but don't just cut and paste the code. Spend time thinking about what each line does. Explore it by copying parts and pasting into console.

```
seed <- 1
n_iter <- 10^5 # number of iterations

mu <- mean(fig_2i[treatment == "ASK1F/F", liver_tg])
sigma <- sd(fig_2i[treatment == "ASK1F/F", liver_tg])

n <- nrow((fig_2i[treatment == "ASK1F/F",]))

sample1 <- matrix(rnorm(n*n_iter, mean=mu, sd=sigma), nrow=n) # 100,000 samples (each size n)
sample2 <- matrix(rnorm(n*n_iter, mean=mu, sd=sigma), nrow=n) # 100,000 samples

#way no. 1 - compute the t-tests manually
mean_diffs <- apply(sample2, 2, mean) - apply(sample1, 2, mean) # what is the apply function returning?
se_mean_diffs <- sqrt(apply(sample2, 2, sd)^2/n + apply(sample1, 2, sd)^2/n)
t_dis <- mean_diffs/se_mean_diffs

#way no. 2 - compute the t-tests using the linear model
fake_data <- rbind(sample1, sample2)
treatment <- rep(c("control", "knockout"), each = n)
t_dis2 <- numeric(n_iter)
for(iter in 1:n_iter){
  y <- fake_data[, iter]
  fake_m1 <- lm(y ~ treatment)
  t_dis2[iter] <- coef(summary(fake_m1))["treatmentknockout", "t value"]
}

# plot the null distribution of t-values
qplot(t_dis2)
```

Now let's use this null distribution of t -values to compute a p -value

```
# what is the p-value?
# the p-value is the number of t-values in t_dis that are as large
# or larger than the observed t. Large, negative t-values
# are as unlikely under the null as large, positive t-values.
# To account for this, we want to use absolute values in our counts
# this is a "two-tail test"

# first assign the observed t-value
```

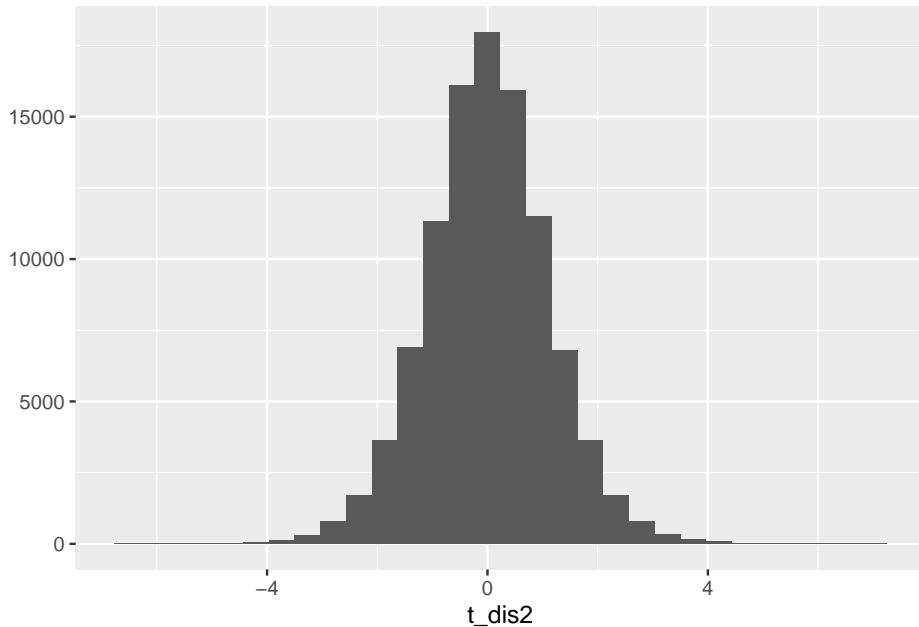


Figure 6.3: Null distribution of t-values. The simulation generated 10,000 t-tests with a true null.

```
t_obs <- fig_2i_m1_coef["treatmentASK1Δadipo", "t value"]

# now count the number of t-values in t_dis as big or bigger than this
# include the observed value as one of these (so add 1 to the count)
count <- sum(abs(t_dis) >= abs(t_obs))

# the p-value is the frequency of t_dis >= t_obs, so divide
# count by the total number of t-values in the distribution.
# Again add one since the observed value counts as a sample
(pASK1Δadipo <- count/(n_iter))

## [1] 0.01174
```

Hey that looks pretty good! Compare this to the p -value in the coefficient table above.

A p -value can be computed by counting the number of simulated t -values, *including the observed value*, that are equal to or more extreme (in either the positive or negative direction) than the observed t . Including the observed t , there are 1174 values that are more extreme than that observed. An approximate measure of p is this count divided by 100,001 (why is 1 added to the

denominator?), which is 0.01174. This simulation-based p -value is very (very!) close to that computed from the observed t -test.

6.4 P-values from the perspective of permutation

A very intuitive way to think about p -values as a frequency is **random permutation**. A permutation is a re-arrangement of items. If there is an effect of ASK1 deletion on liver TG, then the arrangement of the values in the `treatment` column matters. If there is no effect of ASK1 deletion on liver TG, then the arrangement of the values in the `treatment` column does not matter.

Think about the structure of the liver TG data: there are two columns, `treatment`, which contains the assigned treatment, and `liver_tg`. The values in the `treatment` column were randomly assigned prior to the start of the experiment. If there is a negative effect of ASK1 deletion on liver TG, then assignment matters – the values in the `liver_tg` column for the ASK1Δadipo rows will be smaller than, on average, the values in the ASK1F/F rows. That is, a specific value of `liver_tg` is what it is because of the value of `treatment` in the same row. Assignment to ASK1F/F or ASK1Δadipo changes the expected value of `liver_tg`. But, if there were no true effect, then assignment to ASK1F/F or ASK1Δadipo does not change the expected value of `liver_tg`. The expected value of every cell in the `liver_tg` column would be the same regardless of what is in the `treatment` column.

In our thought experiment, let's leave the values in the `treatment` column be, and just randomly re-arrange or permute the values in the `liver_tg` column. What is the new expected difference in liver TG between the rows assigned to ASK1F/F and the rows assigned to ASK1Δadipo? The expected difference is Zero. Because the `liver_tg` values were randomly re-arranged, they *cannot* be caused by treatment assignment.

A permutation is a random re-arrangement of values in a column. Consider the many thousands of permutations of the values in the `liver_tg` column. A difference in means can be computed from each of these permutations and a distribution of differences can be generated. Is the observed difference extreme relative to the other values in this distribution? This is a permutation test – it compares an observed statistic to a distribution of the statistic computed over many thousands of permutations.

Let's create a script for a permutation test

```
set.seed(1)
n_iter <- 5000 # number of random permutations
```

```

y <- fig_2i[, liver_tg]
x <- fig_2i[, treatment]

d_dist_perm <- numeric(n_iter)

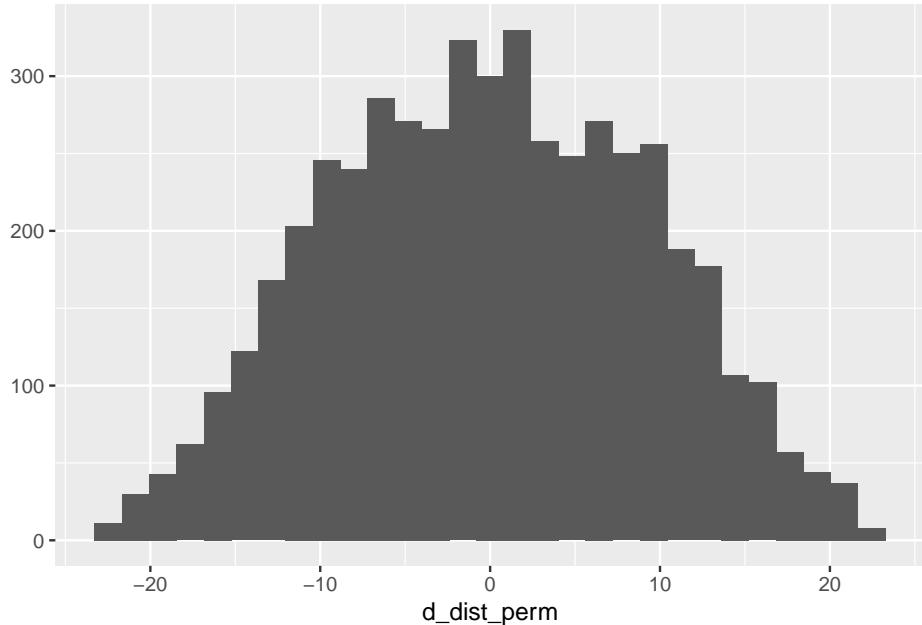
for(iter in 1:n_iter){
  xbar1 <- mean(y[x == "ASK1F/F"])
  xbar2 <- mean(y[x == "ASK1Δadipo"])

  d_dist_perm[iter] <- xbar2 - xbar1

  # permute y
  y <- sample(y, replace=FALSE)
  # note that, when i=1, the first "permutation" is the original arrangement
}

qplot(d_dist_perm)

```



From this distribution of distances generated by random permutation of the response, we can compute a permutation p -value.

```

{rpvalue-d-dist-perm-p } (p_permute <- sum(abs(d_dist_perm) >=
abs(d_dist_perm[1]))/n_iter)

```

6.5 Parametric vs. non-parametric statistics

A statistic such as the difference in mean liver TG between ASK1 Δ adipo and ASK1F/F groups does not have “a” *p*-value. A *p*-value is the probability of observing an event *given a model of how the event was generated*. For the *p*-value in the coefficient table above, the event is sampling a *t*-value from a modeled *t* distribution that is as or more extreme than the observed *t*-value. The model generating the null distribution of *t*-values includes random sampling from a distribution that is defined by specific parameters (in this case, a mean and a variance), these parameters define the location and shape of the distribution of values that could be sampled. A *p*-value computed from a distribution that is defined by a set of parameters is a **parametric *p*-value**.

For the *p*-value computed using the permutation test, the event is the probability of computing a difference of means *from a randomly permuted set of Y* as or more extreme than the observed difference of means. The distribution of differences from the permuted *Y* data sets was not generated by any of the known distributions (normal, Poisson, binomial, etc.) given a specific value of parameters. Consequently, the permutation *p*-value is **non-parametric**.

The validity of all *p*-values depends on a set of model assumptions, which differ from model to model. The permutation *p*-value has fewer assumptions than a parametric *p*-value because no distribution is assumed (the permutation *p*-value is **distribution-free**).

6.6 frequentist probability and the interpretation of p-values

6.6.1 Background

There are at least three different meanings of **probability**.

1. **subjective probability** is the probability that an individual assigns to an event based on prior knowledge and the kinds of information considered reliable evidence. For example, if I asked a sample of students, what is the probability that a 30c homeopathic medicine could clear a *Streptococcus* infection from your respiratory system, their answers would differ because of variation in their knowledge of basic science, including chemistry and physics, their knowledge of what homeopathic medicines are, and how they weight different kinds of evidence.
2. **classical probability** is simply one divided by the number of possible unique events. For example, with a six-sided die, there are six possible unique events. The probability of rolling a 2 is $\frac{1}{6}$ and the probability of rolling an odd number is $\frac{1}{2}$.

3. **frequentist probability** is based on the concept of **long run frequency**. If I roll a die 10 times, the frequency of rolling a 2 will be approximately $\frac{1}{6}$. If I roll the die 100 times, the frequency of rolling a two will be closer to $\frac{1}{6}$. If I roll the die 1000 times, the frequency of rolling the die will be even closer to $\frac{1}{6}$. So the frequentist definition is the expected frequency given an infinite number of rolls. For events with continuous outcomes, a frequentist probability is the long run frequency of *observing an outcome equal to or more extreme than that observed*.

6.6.2 This book covers frequentist approaches to statistical modeling and when a probability arises, such as the *p*-value of a test statistic, this will be a frequentist probability.

When we do a *t*-test, we get a *p*-value. There are several ways to think about this probability. The most compact way is $P(\text{data}|\text{null})$, which is literally read as the probability of the data given the null (or “conditional” on the null), but is really short for *the probability of the data, or something more extreme than the data, given that the null hypothesis is true*. The “probability of the data” is kinda vague. More specifically, we mean the probability of some statistic about the data such as the difference in means between group A and group B or the *t*-value associated with this difference. So, a bit more formally, the probability returned in a *t*-test is $\text{prob}(t \geq t_{\text{obs}}|H_0)$. This is the long run frequency of observing a *t*-value as big or bigger than the observed *t*-value (the one you actually got with your data) if the null is true. Let’s parse this into “long run frequency of observing a *t*-value as big or bigger than the observed *t*-value” and “null is true”.

A thought experiment: You open a google sheet and insert 12 standard, normal random deviates (so the true mean is zero and the true variance is one) in Column A, rows 1-12. You arbitrarily assign the first six values (rows 1-6) to treatment A and the second six values (rows 7-12) to treatment B. You use the space immediately below these data to compute the mean of treatment A, the mean of treatment B, the difference in means (A - B), and a *t*-value. Unfortunately, google sheets doesn’t have a *t*-value function so you’d have to compute this yourself. Or not, since this is a thought experiment. Now “fill right” or copy and paste these functions into 999 new columns. You now have 1000 *t* tests. The expected value of the difference in means is zero (why?) but the actual values will form a normal distribution about zero. Most will be close to zero (either in the negative or positive direction) but some will be further from zero. The expected *t*-value will also be zero (why?) and the distribution of these 1000 *t*-values will look normal but the tails are a little fuller. This row of *t*-values is a null distribution, because in generating the data we used the exact same formula for the values assigned to A and the values assigned to B. Now think of a *t*-value in your head, say 0.72 (remember that *t*-values will largely

range from about -3 to +3 although the theoretical range is $-\infty$ to $+\infty$. What is the probability of observing a t of 0.72 or bigger if the null is true? Look at the row of t -values! Count the number of $t \geq 0.72$ and then divide by the total number of t -values in the row (1000) and you have a probability computed as a frequency. But remember the frequentist definition is the long run frequency, or the expected frequency at the limit (when you've generated not 1000 or even 1,000,000 but an infinite number of columns and t -values).

Some asides to the thought experiment: First, why “as big or bigger” and not just the probability of the value itself? The reason is that the probability of finding the exact t is 1/infinity, which doesn't do us much good. So instead we compute the probability of finding t as big, or bigger, than our observed t . Second, the t -test probability described above is a “one-tail probability”. Because a difference can be both in the positive direction and the negative direction, we usually want to count all the $t \geq 0.72$ and the $t \leq -0.72$ and then add these two counts to compute the frequency of *as extreme or more extreme* values. This is called a “two-tailed probability” because we find extremes at both tails of the distribution. Third, we don't really count $t \geq 0.72$ but take advantage of the beautiful mathematical properties of the theoretical t distribution, which allows us to compute the frequentist probability (expected long range frequency) given the t -value and the degrees of freedom using the t -distribution.

Now what do I mean with the phrase “null is true”? Most people equate “null is true” with “no difference in means” but the phrase entails much more than this. Effectively, the phrase means that the p -value is based on modeling the real data with a theoretical sample in which all the points were randomly sampled from the same distribution and that the assignment of the individual points to treatment was random. This model means the theoretical sample has three properties: First, random assignment to treatment after sampling from the same distribution means that the expected means are the same, or put differently, the expected difference in means between the assigned groups is zero. Second, random assignment to treatment after sampling from the same distribution *also* means that the expected variances of the two groups are equal. And third, random sampling means that the values of each point are independent – we cannot predict the value of one point knowing information about any other point. **Here is what is super important about this:** if we get a really low p -value, any one of these consequences may be untrue about our data, for example it could be that the true means of the two treatment groups really are different, or it could mean it is the variances that differ between the two groups, or it could mean that the data (or technically, the errors) are not independent of each other. This is why we need certain assumptions to make a p -value meaningful for empirical data. By assuming independent error and homogenous (equal) variances in our two samples, a low p -value is evidence of unequal means.

6.6.3 Two interpretations of the p -value

Since we want to be working scientists who want to use p -values as a tool, we need to know how to interpret (or use) the p -value to make reasonable inferences and how to avoid mis-interpreting the p -value and making unreasonable or even incorrect inferences. Ronald Fisher, the inventor of frequentist statistics, developed an interpretation of the p -value that is probably most useful for academic and applied research programs. Neyman and Pearson (Neyman-Pearson) gave the p -value a different interpretation, one that is probably most useful for industrial quality control. Today's biology researchers use an interpretation that is an odd hybrid of the two, which often leads to silly inference. Regardless, understanding the distinction between Fisher and Neyman-Pearson will inform how we write up our results in a manuscript. I'll describe these in the context of the two-sample t -test.

6.6.3.1 Fisher's interpretation

Fisher was working in the context of agricultural experiments, the goal of which was to discover better agricultural practices. Does this new fertilizer work better than our old fertilizer? This is the context of much of modern biosciences and clinical medicine. Fisher thought of p as evidence against the null; the smaller the p the stronger the evidence that the two sampling distributions differ, which, in an experimental context, implies a treatment effect. If an experiment results in a large p -value, we can move on and test other fertilizers. If an experiment results in a small p -value, we want to pursue this new fertilizer more. Do more experiments! Fisher never thought of a single experiment as definitive. The decision to move on or pursue is only partly informed by the p -value and Fisher offered no rule about what p -value lies on the threshold of this decision. When pressed, Fisher might say that $p = 0.05$ is a reasonable threshold.

6.6.3.2 Neyman-Pearson interpretation

Neyman-Pearson thought of p as the necessary and sufficient information to make a decision between accepting the null (or at least not rejecting the null) or rejecting the null and accepting an alternative hypothesis. This decision balances two sorts of errors: Type I (false positives), which they called α , and Type II (false negatives), which they called β . A false positive means the null was rejected but there really is no effect. A false negative means that the null was not rejected but there actually is an effect. α is set by the experimenter and is the long-term frequency (or "rate") of false positives **when the null is true** that the experimenters are willing to accept. This is easily understood in the context of manufacturing. I've just made a batch of beer that I now need to ship. I sample 10 cans and test the quality against a norm. If $p < \alpha$, we reject

the null in favor of the alternative – something may be wrong with the batch, it differs from the norm. We throw the beer away. If $p > \alpha$, we do not reject the null, nor the beer! We ship it.

After setting α , the experimenter designs the experiment to achieve an acceptable rate of β . Since β is the false negative rate then $1 - \beta$ is the rate of not making a false negative error, that is, the rate of rejecting the null when there really is an effect. This is called the **power** of the experiment. An experiment with high power will have a low probability of a Type II error. An experiment with low power will have a high probability of a Type II error. Power is partly determined by sample size, the bigger the sample the smaller the p -value, all other things equal (think about why in the context of the formula for the t -value). Power is a function of error variance, both the natural variance and the component added because of measurement error (think about why in the context of the formula for the t -value). Power is also a function of α . If we set a low α (say, $\alpha = 0.01$), the test is conservative. We are more likely to fail to reject the null even if the null is false. A researcher can increase power by increasing sample size, using clever strategies to reduce measurement error, or increasing alpha.

An experimenter sets α , computes the sample size needed to achieve a certain level of power ($1 - \beta$), and then does the experiment. A thoughtful researcher will set α after considering and weighing the pros and cons of different levels of α . If false positives have costly consequences (expense, time, deleterious side-effects), then set α to a low value, such as 0.01 or 0.001. For example, if an initial screen has identified a previously unknown candidate that potentially functions in the focal system of the researcher, then a researcher might decide to set a low α (0.001) in the initial tests of this candidate to avoid devoting time, personnel, and expense to chasing a phantom (a false-positive candidate). If false positives have trivial consequences, then set α to a high value, such as 0.05, or 0.1, or even 0.2. For example, if the initial tests of a candidate in a functional system are cheap and fast to construct, then a researcher might choose to set a high α for the screen that identifies candidates. False positive candidates don't cost the lab much effort to identify them as false, but missing positive candidates because of a small α (which results in low power) at the screen stage costs the researcher the discovery of a potentially exciting component of the functional system.

In Fisher's interpretation, there is no α , no β , no alternative hypothesis, and no sharp decision rule. Instead, in Fisher, p is a continuous measure of evidence against the null and its value is interpreted subjectively by an informed and knowledgeable expert using additional information to make decisions. Neyman-Pearson rejected Fisher's conception of p as evidence against the null arguing that a single experimental p -value is too noisy without embedding it into a more formal system of decision making that maintains long-term type I error rates at α , given a certain power. In Neyman-Pearson, p is compared to a threshold, α and this alone makes the decision. In Neyman-Pearson, p is **not treated as continuous information**. $p = 0.00000001$ is no more evidence to use to reject

the null than $p = 0.049$.

6.6.4 NHST

Most biology researchers today interpret p using a combination of Fisher and Neyman-Pearson concepts in what has become known as Null Hypothesis Significance Testing (NHST).

1. Nearly all papers in biology either explicitly state something like “P values < 0.05 were considered to be statistically significant” or implicitly use 0.05 as the “level of significance” (α). Comparing a p -value to a pre-defined α is Neyman-Pearson.
2. Unlike Neyman-Pearson, there is no evidence that researchers are thoughtfully considering the level of α for each experiment. Instead, researchers mindlessly choose $\alpha = 0.05$ because this is what everyone else uses.
3. Unlike Neyman-Pearson, but somewhat in the spirit of Fisher, researchers, journals, and textbooks, advocate trichotomizing a statistically significant p into “significance bins” – three asterisks for $p < 0.001$, two asterisks for $0.001 < p < 0.01$, and one asterisk for $0.01 < p < 0.05$. This is not Neyman-Pearson. Again, Neyman-Pearson developed a system to control the long-run frequency of Type I error, which is controlled by a strict use of α . If an observed p -value is in the *** bin or the * bin is meaningless in a system using Neyman-Pearson. There is only “accept” ($p \geq \alpha$) or “reject” ($p < \alpha$).
4. Many researchers report exact p -values when $p < 0.05$ but “n.s.” (not significant) when $p > 0.05$. Reporting exact p -values is Fisher. Reporting n.s. is Neyman-Pearson.
5. Many researchers further polychromotomize the p -value space just above 0.05 by using language such as “marginally significant”. If Neyman-Pearson and Fisher got together and spawned a love-child, this would be it.

6.7 Some major misconceptions of the p -value

Setting the type I error rate α to 0.05 is so pervasive that I’m going to simply use “0.05” instead of “alpha” in discussing misconceptions.

6.7.1 Misconception: p is the probability that the null is true *and* $1 - p$ is probability that the alternative is true

Many researchers believe that if $p > 0.05$ then “there is no effect.” A frequentist hypothesis test cannot show that an effect doesn’t exist, only that the null has

a low probability of producing a test statistic as extreme or more extreme than the observed effect.

Many researchers believe that if $p < 0.05$ then “there is an effect.” Again, a frequentist hypothesis test cannot show that an effect exists, only that the null has a low probability of producing a test statistic as extreme or more extreme than the observed effect.

1. The statement “There is no effect of predators on feeding behavior” is not a valid conclusion of a frequentist hypothesis test.
2. The statement “We found no effect of predators on feeding behavior” is misleading because a frequentist hypothesis test can neither find an effect nor find no effect.

The two errors above are gross misconceptions that are pervasive in the biology literature. A more subtle issue is the belief that a low p -value shows that the researcher’s explanatory hypothesis is correct. For example, researchers believe the result “the prey fish fed 14.2 (95% CI: 9.2, 19.2) minutes shorter in the presence of the predator fish” confirms their hypothesis that prey modulate feeding duration as a function of their ability to assess the risk of predation. Some alternative explanations:

1. The predator fish also competes with the prey fish for the prey fish’s food and with less food the prey fish spends less time feeding because it gives up when food density drops below some amount.
2. The predator fish is introduced to the prey tank by hand and odorant molecules from the researcher’s hands are detected by the prey and the prey reduces feeding duration because of these odorants.

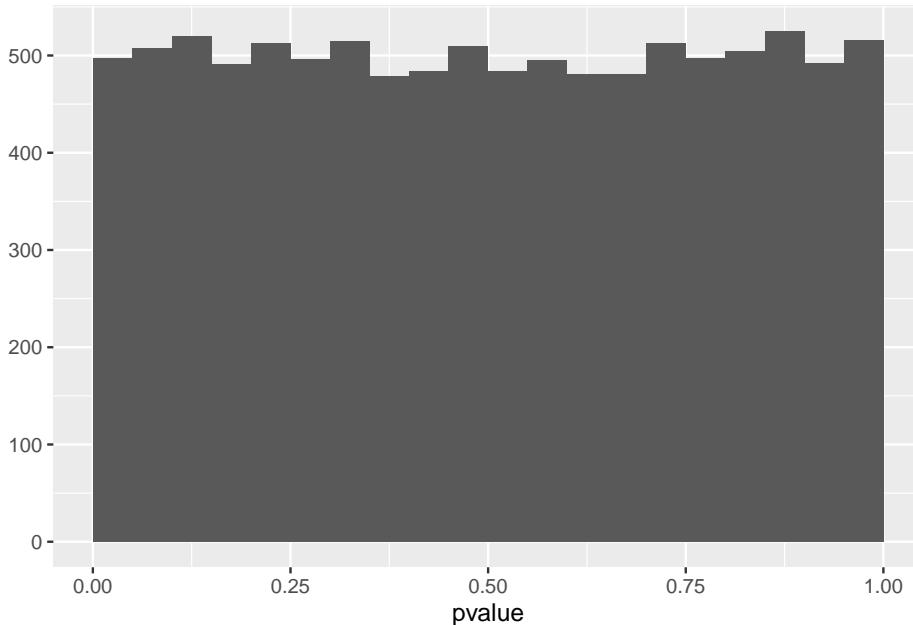
Importantly, no single experiment confirms an explanatory hypothesis. Instead, alternative explanations require multiple experiments with different controls to “rigorously probe” the preferred hypothesis.

6.7.2 Misconception: a p -value is repeatable

Many researchers believe that a p -value is a precise measure – that if the experiment were replicated, a similar p would result. This belief requires at least two misconceptions. First, if the null were true, then *any* p -value is equally likely. $p = 0.00137$ is just as likely as $p = 0.492$. In other words, if the null were true, the p -value is not replicable at all! Second, the p -value is highly dependent on the sample, and can be highly variable among replications, but there is no true p -value, so there can be no estimate or standard error. Let’s explore these.

6.7.2.1 What is the distribution of p -values under the null?

I often ask students, “if the null were true, what is the most likely p -value?” or “if the null were true, what kind of p -values would we expect, that is what is the expected distribution”. A common answer is $p = 0.5$ is the most likely value and something like a normal curve, except the tails abruptly stop at 0 and 1, is the expected distribution.



6.7.2.2 The incredible inconsistency of the p -value

How replicable is the conclusion of an experiment if the p -value for a t -test is 0.03? If our conclusion is based on $p < 0.05$, then the conclusion is not very replicable. The simulation below shows the results of 15 replicates of an experiment with true power of 40%. There are five “significant” results (one less than expected) but several replicates have very high p -values.

6.7.3 Misconception: 0.05 is the lifetime rate of false discoveries

An important and widespread misconception is that if a researcher consistently uses $\alpha = 0.05$, then the frequency of incorrectly concluding an effect exists, or “discovering” an effect, over the lifetime of the researcher, will be 5%. This is incorrect. α is the rate of false positive if the null hypothesis is true, so our lifetime “false discovery” rate could only be 5% if everything we ever tested has

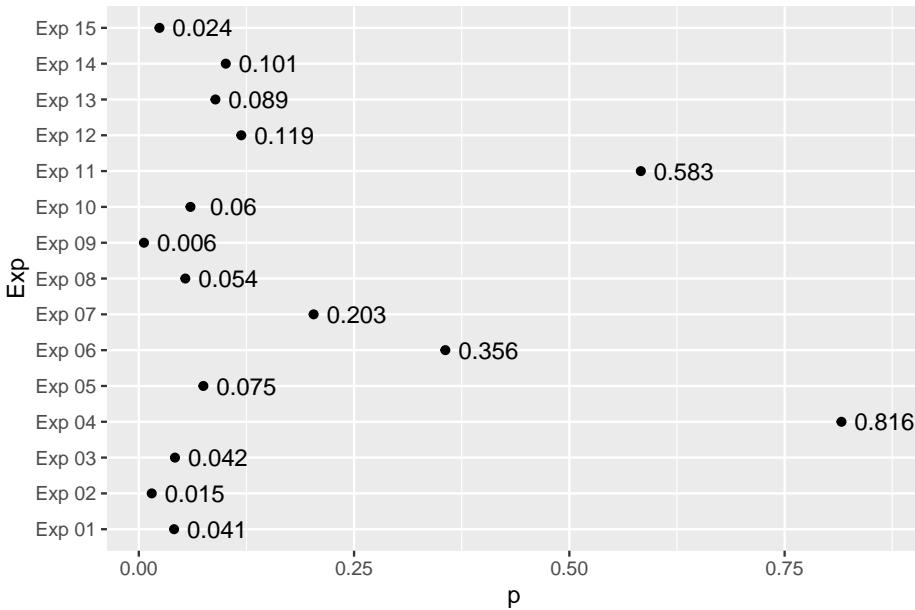


Figure 6.4: Variability of p -values when the power is 0.4

no true effect! More generally, the **false discovery rate** is the frequency of false positives divided by the frequency of positives (the sum of false and true positives). This differs from the Type I error rate, which is the frequency of false positives divided by the frequency of tests *in which the null is true*.

Imagine we test

1. 1000 null hypotheses over a lifetime
2. 60% are true nulls, this means there are 600 true nulls and 400 true effects
3. alpha is 5%. This means we expect to find $p \leq 0.05$ 30 times (0.05×600) when the null is true
4. power is 25%. This means we expect to find $p \leq 0.05$ 100 times (0.25×400) when the null is false
5. We have made $30 + 100 = 130$ “discoveries” (all experiments with $p \leq 0.05$), but
6. 30 of the 130, or 23%, are “false discoveries”. This is the false discovery rate.

Think about this. If the null is never true, you cannot have a false discovery—every $p \leq 0.05$ is a true discovery (the false discovery rate is 0%). And if the null is always true, every $p < 0.05$ is a false discovery (the false discovery rate is 100%).

6.7.4 Misconception: a low p -value indicates an important effect

Many researchers write results as if they believe that a small p -value means the effect is big or important. This may misconception may arise because of the ubiquitous use of “significant” to indicate a small p-value and “very” or “extremely” or “wicked” significant to indicate a really small p-value. Regardless, this is a misconception. A small p-value will usually result when there is high power (but can occur even if power is low) and power is a function of effect size, variability (the standard deviation), and sample size. A small p could result from a large effect size but can also result with a small effect size if the sample size is big enough.

This is easy to simulate (see script below). Let’s model the effect of the genotype of a gene on height

```
set.seed(1)
rho <- 0.5
n <- 10^4
genotype <- c("+/+", "+/-", "-/-")
Sigma <- diag(2)
Sigma[1,2] <- Sigma[2,1] <- rho
X <- rmvnorm(n, mean=c(0,0), sigma=Sigma)
colnames(X) <- c("X1", "X2")
beta <- c(0.05, 0.05)
y <- X%*%beta + rnorm(n)
fit <- lm(y ~ X)
coefficients(summary(fit))

##             Estimate Std. Error    t value    Pr(>|t|)
## (Intercept) 0.007472959 0.01007946 0.7414046 4.584656e-01
## XX1         0.044304824 0.01154709 3.8368830 1.253725e-04
## XX2         0.048228101 0.01170855 4.1190490 3.835033e-05
```

6.7.5 Misconception: a low p -value indicates high model fit or high predictive capacity

On page 606, of Lock et al “Statistics: Unlocking the Power of Data”, the authors state in item D “The p-value from the ANOVA table is 0.000 so the model as a whole is effective at predicting grade point averages.” This is incorrect. A p-value is not a measure of the predictive capability of a model because the p-value is a function of the signal, noise (unmodeled error), and *sample size* while predictive ability is a function of just the signal:noise ratio. If the signal:noise ratio is tiny, the predictive ability is small but the p-value can be tiny if the sample size is large. This is easy to simulate (see script below). The whole-model

p-value is exceptionally small (0.00001002) but the relative predictive ability, measured by the R^2 , is near zero (0.002).

```

set.seed(1)
rho <- 0.5
n <- 10^4
Sigma <- diag(2)
Sigma[1,2] <- Sigma[2,1] <- rho
X <- rmvnorm(n, mean=c(0,0), sigma=Sigma)
colnames(X) <- c("X1", "X2")
beta <- c(0.05, -0.05)
y <- X%*%beta + rnorm(n)
fit <- lm(y ~ X)
summary(fit)

##
## Call:
## lm(formula = y ~ X)
##
## Residuals:
##    Min      1Q  Median      3Q     Max
## -3.6449 -0.6857  0.0148  0.6756  3.6510
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.007473  0.010079   0.741 0.458466
## XX1         0.044305  0.011547   3.837 0.000125 ***
## XX2        -0.051772  0.011709  -4.422 9.9e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.008 on 9997 degrees of freedom
## Multiple R-squared:  0.0023, Adjusted R-squared:  0.002101
## F-statistic: 11.52 on 2 and 9997 DF,  p-value: 1.002e-05

```

6.8 What the p -value does not mean

1. p is not the probability of the null being true. More formally, this probability is $Prob(null|data)$ but our p -value is $P(data|null)$. These are not the same. $P(null|data)$ is the probability of the null being true given the data. $P(data|null)$ is the probability of our data, or something more extreme than our data, conditional on a true null.
2. $1 - p$ is not the probability of the alternative
3. p is not a measure of effect size.

4. p in one experiment is not the same level of evidence against the null as in another experiment
5. p is not a great indicator of which is more likely, H_0 or H_1 .
6. If one treatment level has $p < 0.05$ and another treatment level has $p > 0.05$, this is not evidence that the treatment levels have different effects on the outcome.

6.9 Recommendations

1. Simply report the exact p -value, along with a CI of the estimate.
 - P -values are noisy, there is little reason to report more than two significant digits (report “ $p = 0.01$ ” not “ $p = 0.0108$ ”) although some journals recommend more than two significant digits.
 - For high p -values, report “ $p = 0.23$ ” not “ $p = n.s.$ ”.
 - For small p -values, there is little reason to report more than one significant digit (report “ $p = 0.0002$ ” not “ $p = 0.00018$ ”).
 - For really small p -values, there is little reason to report the exact p -value (report “ $p < 0.0001$ ” and not “ $p = 2.365E - 11$ ”). Recognize that “really small” is entirely arbitrary. Rafael Irizarry suggested that p -values less than something like the probability of being killed by lightning strike should be reported as “ $p < m$ ”, where m is the probability of being killed by lightning strike². According Google University, this is 0.00000142 in one year or 0.00033 in one lifetime. This text will use $p < ls$ ” for p -values less than 0.0001 – the lifetime probability of being killed by lightning strike in someone that spends too much time in doors analyzing data.
2. If $p < 0.05$ (or some other α) do not report this as “significant” – in fact, avoid the word “significant”. In the english language, “significant” implies big or important. Small p -values can result even with trivially small effects if n is big or sample variation is small. The phrase “ASK1 knockout had a significant effect on reducing liver TG ($p = 0.011$)” is
 - potentially misleading, if we interpret “significant” to mean “having a large effect on the regulation of liver TG”,
 - wrong, if we interpret “significant” to mean “there is an ASK1 knockout effect”. A low p -value is evidence that the effect of ASK1 knockout is not zero, but I would wager that knocking out any gene expressed in white adipose cells will have *some* effect (however small) on liver TG.
3. If a decision needs to be made (“do we devote time, expense, and personnel to pursue this further?”), then a p -value is a useful tool. If p is smaller

²<https://twitter.com/rafalab/status/1310610623898808320>

than say 0.001, this is pretty good evidence that the data is not a fluke of sampling, **as long as we are justifiably confident in all the assumptions that went into computing this p -value**. A replicate experiment with a small p -value is better evidence. If p is closer to 0.01 or 0.05, this is only weak evidence of a fluke because of the sampling variability of p . A replicate experiment with a small p -value is *much* better evidence.

6.9.1 Primary sources for recommendations

1. ””. Statistical tests, P values, confidence intervals, and power: a guide to misinterpretations.
2. “Q: Why do so many colleges and grad schools teach $p = 0.05$? A: Because that’s still what the scientific community and journal editors use. Q: Why do so many people still use $p = 0.05$? A: Because that’s what they were taught in college or grad school.” – ASA Statement on Statistical Significance and P-Values
3. “We then discuss our own proposal, which is to abandon statistical significance. We recommend dropping the NHST paradigm—and the p -value thresholds intrinsic to it—as the default statistical paradigm for research, publication, and discovery in the biomedical and social sciences.” – Abandon Statistical Significance
4. “We conclude, based on our review of the articles in this special issue and the broader literature, that it is time to stop using the term “statistically significant” entirely. Nor should variants such as “significantly different,” “ $p < 0.05$,” and “nonsignificant” survive, whether expressed in words, by asterisks in a table, or in some other way.” – Moving to a World Beyond “ $p < 0.05$ ”
5. “We agree, and call for the entire concept of statistical significance to be abandoned.”—Scientists rise up against statistical significance

6.10 Problems

Problem 1 – simulate the distribution of p under the null. There are many ways to do this but a straightforward approach is to

1. Create a $2n \times m$ matrix of random normal deviates with mean 0 and sd 1
2. Do a t -test on each column, with the first n values assigned to one group and the remaining n values assigned to the second group. Save the p -value from each.
3. Plot a histogram of the p -values.
4. What is the distribution? What is the most likely value of p ?

Problem 2 – simulate power. Again, many ways to do this but following up on Problem 1. 1. Create a $2n \times m$ matrix of random normal deviates with mean

0 and sd 1 2. Add an effect to the first n values of each column. Things to think about a. what is a good effect size to add? The effect/sd ratio, known as Cohen's d, is a relative (or standardized) measure of effect size. Cohen suggest 0.2, 0.5, and 0.8 as small, medium, and large standardized effects. b. should the same effect be added to each individual? Yes! It is the random component that captures the individual variation in the response. 3. Do a t -test on each column of the matrix, using the first n values in group 1 and the remaining n values in group 2. Save the p-values for each. 4. Compute the power, the relative frequency $p \leq 0.05$. 5. Repeat with different values of n , effect size, and sd, but only vary one at a time. How does power vary with these three parameters?

Chapter 7

Errors in inference

7.1 Classical NHST concepts of wrong

As described in chapter (p-values), two types of error occur in classical Neyman-Pearson hypothesis testing, and in the NHST version that dominates modern practice. **Type I** error occurs when the null hypothesis is true but the *p*-value of the test is less than α . This is a *false positive*, where a *positive* is a test that rejects the null. **Type II** error occurs when the null hypothesis is false but the *p*-value of the test is greater than α . This is a *false negative*, where a *negative* is a test that accepts (or fails to reject) the null. **Power** is not an error but the frequency of true, positive tests (or the frequency of avoiding Type II error). α is not an error but the rate of Type I error that a researcher is willing to accept. Ideally, a researcher sets α based on an evaluation of the pros and cons of Type I and Type II error for the specific experiment. In practice, researchers follow the completely arbitrary practice of setting $\alpha = 0.05$.

Why should a researcher care about α and power? Typically, most researchers don't give α much thought. And power is considered only in the context of calculating a sample size for an experiment for a grant proposal. But researchers should care about rates of Type I error and power because these (and similar concepts) can help guide decisions about which model to fit to a specific dataset.

7.1.1 Type I error

In classical Neyman-Pearson hypothesis testing, an important property of a hypothesis test is the **size of a test**, which may include an entire procedure that culminates in a hypothesis test. “Size” is a weird name for the probability of rejecting the null when the null is true. Size is not α . α is the *nominal value* – it's what a researcher wants. Size is the *actual value* – it's what a researcher gets.

It would probably come as a surprise to most researchers to learn that the size of some common tests used with data that look like the researcher's data is not 0.05. "used with data that look like the researcher's data" is important here – a *t*-test doesn't have one size. With data that conform to the assumptions (independence, homogeneity, normality), the size of a *t*-test is α . But with any violation, especially when the sample size differs between groups, the size of the *t*-test can move away from α . A test that has a size that is less than α is "conservative" (fewer nulls are rejected than we think, so the *status quo* is more often maintained). A test that has a size that is greater than α is "liberal" (more nulls are rejected than we think, so the *status quo* is less often maintained). More conservative tests reduce power. More liberal tests *artificially* increase power and increase our rate of false rejection, which can mean "false discovery" if *p*-values are used as the arbiter of discovery.

7.1.1.1 Size example 1: the size of a *t*-test vs. a permutation test, when the data meet the assumptions

```
set.seed(1)
n <- 10
n_iter <- 10000
p_t <- numeric(n_iter)
p_perm <- numeric(n_iter)

treatment <- rep(c("cn", "tr"), each = n)
for(iter in 1:n_iter){
  sample_1 <- rnorm(n, mean = 10, sd = 1)
  sample_2 <- rnorm(n, mean = 10, sd = 1)
  y <- c(sample_1, sample_2)
  m1 <- lm(y ~ treatment) # no data statement necessary because both variables in work
  p_t[iter] <- coef(summary(m1))["treatmenttr", "Pr(>|t|)"]

  m2 <- lm(y ~ treatment,
             perm = "Prob",
             settings = FALSE)
  p_perm[iter] <- coef(summary(m2))["treatment1", "Pr(Prob)"]
}
size_t <- sum(p_t < 0.05)/n_iter
size_perm <- sum(p_perm < 0.05)/n_iter
size_table <- data.table(Method = c("lm", "perm"),
                           Size = c(size_t, size_perm))
knitr::kable(size_table, digits = 4)
```

Method

Size

lm

0.0489

perm

0.0488

7.1.1.2 Size example 2: the size of a t -test vs. a permutation test, when the data have a right skewed distribution

```
set.seed(1)
n <- 10
n_iter <- 10000
p_t <- numeric(n_iter)
p_perm <- numeric(n_iter)

treatment <- rep(c("cn", "tr"), each = n)
for(iter in 1:n_iter){
  # qplot(rnegbin(n = 10^4, mu = 100, theta = 1))
  sample_1 <- rnegbin(n, mu = 100, theta = 1)
  sample_2 <- rnegbin(n, mu = 100, theta = 1)
  y <- c(sample_1, sample_2)
  # qplot(x=treatment, y = y)
  m1 <- lm(y ~ treatment) # no data statement necessary because both variables in workspace
  p_t[iter] <- coef(summary(m1))["treatmenttr", "Pr(>|t|)"]

  m2 <- lmp(y ~ treatment,
             perm = "Prob",
             settings = FALSE)
  p_perm[iter] <- coef(summary(m2))["treatment1", "Pr(Prob)"]
}
size_t <- sum(p_t < 0.05)/n_iter
size_perm <- sum(p_perm < 0.05)/n_iter
size_table <- data.table(Method = c("lm", "perm"),
                           Size = c(size_t, size_perm))
knitr::kable(size_table, digits = 4)
```

Method

Size

lm

0.0438

```
perm
```

```
0.0504
```

7.1.1.3 Size example 3: the size of a t -test vs. a permutation test, when the data have heterogenous variance and the sample size is unequal

```
set.seed(1)
n1 <- 10
n2 <- n1/2
n_iter <- 10000
p_t <- numeric(n_iter)
p_perm <- numeric(n_iter)

treatment <- rep(c("cn", "tr"), times = c(n1, n2))
for(iter in 1:n_iter){
  # qplot(rnegbin(n = 10^4, mu = 100, theta = 1))
  sample_1 <- rnorm(n1, mean = 10, sd = 0.5)
  sample_2 <- rnorm(n2, mean = 10, sd = 1)
  y <- c(sample_1, sample_2)
  # qplot(x=treatment, y = y)
  m1 <- lm(y ~ treatment) # no data statement necessary because both variables in work
  p_t[iter] <- coef(summary(m1))["treatmenttr", "Pr(>|t|)"]

  m2 <- lmp(y ~ treatment,
             perm = "Prob",
             settings = FALSE)
  p_perm[iter] <- coef(summary(m2))["treatment1", "Pr(Prob)"]
}
size_t <- sum(p_t < 0.05)/n_iter
size_perm <- sum(p_perm < 0.05)/n_iter
size_table <- data.table(Method = c("lm", "perm"),
                           Size = c(size_t, size_perm))
knitr::kable(size_table, digits = 4)
```

```
Method
```

```
Size
```

```
lm
```

```
0.1150
```

```
perm
```

```
0.1211
```

7.1.2 Power

In classical Neyman-Pearson hypothesis testing, an important property of a hypothesis test is the **power of a test**. “Power” is the probability of rejecting the null when the null is false. A common way to think about power is, power is a test’s ability to “detect” an effect if it exists. This makes sense using Neyman-Pearson but not Fisher (Using Fisher, a p -value is not a detector of an effect – a reasoning brain is). Using Fisher, we could say that power is the sensitivity of a test (it takes less sample to provide the same signal).

7.1.2.1 Power example 1: the power of a t -test vs. a permutation test, when the data meet the assumptions

```
set.seed(1)
n <- 10
n_iter <- 10000
p_t <- numeric(n_iter)
p_perm <- numeric(n_iter)

treatment <- rep(c("cn", "tr"), each = n)
for(iter in 1:n_iter){
  sample_1 <- rnorm(n, mean = 10, sd = 1)
  sample_2 <- rnorm(n, mean = 11, sd = 1)
  y <- c(sample_1, sample_2)
  m1 <- lm(y ~ treatment) # no data statement necessary because both variables in workspace
  p_t[iter] <- coef(summary(m1))["treatmenttr", "Pr(>|t|)"]

  m2 <- lmp(y ~ treatment,
              perm = "Prob",
              settings = FALSE)
  p_perm[iter] <- coef(summary(m2))["treatment1", "Pr(Prob)"]
}
power_t <- sum(p_t < 0.05)/n_iter
power_perm <- sum(p_perm < 0.05)/n_iter
power_table_normal <- data.table(Method = c("lm", "perm"),
                                    Power = c(power_t, power_perm))
knitr::kable(power_table_normal, digits = 3)
```

Method

Power

lm

0.554

perm

0.554

7.1.2.2 Power example 2: the power of a t -test vs. a permutation test, when the data look like typical count data

```
set.seed(1)
n <- 10
n_iter <- 10000
p_t <- numeric(n_iter)
p_perm <- numeric(n_iter)

treatment <- rep(c("cn", "tr"), each = n)

for(iter in 1:n_iter){
  # qplot(rnegbin(n = 10^4, mu = 100, theta = 1))
  sample_1 <- rnegbin(n, mu = 100, theta = 1)
  sample_2 <- rnegbin(n, mu = 300, theta = 1)
  y <- c(sample_1, sample_2)
  # qplot(x=treatment, y = y)
  m1 <- lm(y ~ treatment) # no data statement necessary because both variables in work
  p_t[iter] <- coef(summary(m1))["treatmenttr", "Pr(>|t|)"]

  m2 <- lmp(y ~ treatment,
             perm = "Prob",
             settings = FALSE)
  p_perm[iter] <- coef(summary(m2))["treatment1", "Pr(Prob)"]
}

power_t <- sum(p_t < 0.05)/n_iter
power_perm <- sum(p_perm < 0.05)/n_iter
power_table_count <- data.table(Method = c("lm", "perm"),
                                  Power = c(power_t, power_perm))
knitr::kable(power_table_count, digits = 3)
```

Method

Power

lm

0.512

perm

0.584

7.2 A non-Neyman-Pearson concept of power

Size and power are concepts specific to the Neyman-Pearson hypothesis testing framework. Size and power also have limited (or no) use in a research program in which the null hypothesis is never (or rarely) strictly true. That said, the concept of size and power are useful. For example, what if we framed power as the distribution of p -values instead of the frequency of p -values less than α .

Table ?? shows the p -value at the 10th, 25th, 50th, 75th, and 90th percentile of the set of p -values computed in Power Example 2 above (count data). The n th percentile is the value in an ordered set of numbers in which $n\%$ are less than the value and $100 - n\%$ are greater than the value. The 50th percentile is the median. The table shows that at all percentiles except the 90th, the permutation p -value is smaller than the t -test p -value. And, importantly, the value at 75% for both is ~ 0.12 . This means that for experiments that generate data something like the fake data generated in Power Example 2, the permutation test is more sensitive to the incompatibility between the null model and the data than the t -test, except in the random samples when both methods fail.

```
quantile_list <- c(0.1, 0.25, 0.5, 0.75, 0.9)
percentiles_t <- quantile(p_t, quantile_list)
percentiles_perm <- quantile(p_perm, quantile_list)

alt_power_table <- data.table(method = c("t-test", "permutation"),
                               rbind(percentiles_t,
                                     percentiles_perm))
knitr::kable(alt_power_table, digits = c(1, 4, 3, 3, 2, 2))
```

method	10%	25%	50%	75%	90%
t-test	0.0045	0.016	0.047	0.13	0.28
permutation	0.0045	0.016	0.047	0.12	0.28

permutation

0.0012

0.007

0.032

0.12

0.32

7.2.1 Estimation error

7.2.2 Coverage

This text advocates reporting a confidence interval with each reported effect size. An important property of an **estimator** is **coverage probability**, often shortened to “coverage”.

7.2.3 Type S error

Instead of framing the “size” concept as the rate of Type I error, what if we framed this as the rate that an estimate is in the correct direction (meaning, the sign of an effect is the same as the true value). And,

7.2.4 Type M error

Part V: Introduction to Linear Models

Chapter 8

An introduction to linear models

Chapter 2 (Analyzing experimental data with a linear model) is an introduction to how to use a linear model to estimate treatment effects, with only a few explanations of what a linear model *is*. This chapter introduces the linear model more generally and expands on different goals of linear modeling, including description and prediction, in addition to explanation (the estimate of treatment effects, or causal effects more generally).

All students are familiar with the idea of a linear model from learning the equation of a line, which is

$$Y = mX + b \tag{8.1}$$

where m is the slope of the line and b is the Y -intercept. It is useful to think of equation (8.1) as a function that maps values of X to values of Y . Using this function, if we input some value of X , we always get the same value of Y as the output.

A linear model is a function, like that in equation (8.1), that is fit to a set of data, often to model a process that generated the data or something like the data. The line in Figure 8.1A is just that, a line, but the line in Figure 8.1B is a linear model fit to the data in Figure 8.1B.

8.1 Two specifications of a linear model

8.1.1 The “error draw” specification

A linear model is commonly specified using an “measurement error model”.

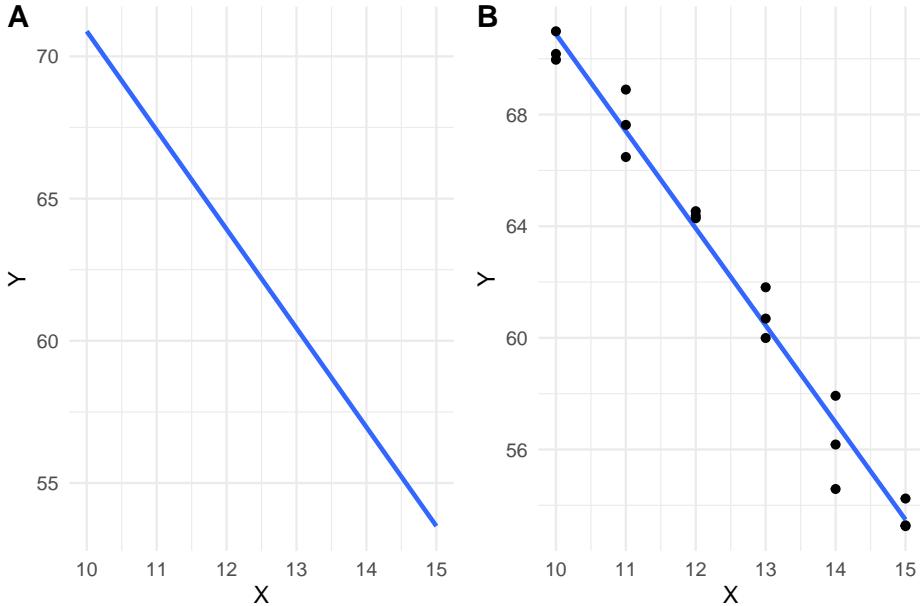


Figure 8.1: A line vs. a linear model. (A) the line $y = -3.48X + 105.7$ is drawn. (B) A linear model fit to the data. The model coefficients are numerically equal to the slope and intercept of the line in A.

$$Y = \beta_0 + \beta_1 X + \varepsilon \quad (8.2)$$

$$\varepsilon \sim N(0, \sigma^2) \quad (8.3)$$

The first line of this specification has two components: the **linear predictor** $Y = \beta_0 + \beta_1 X$ and the **error** ε . The linear predictor component looks like the equation for a line except that 1) β_0 is used for the intercept and β_1 for the slope and 2) the intercept term precedes the slope term. This re-labeling and re-arrangement make the notation for a linear model more flexible for more complicated linear models. For example $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \varepsilon$ is a model where Y is a function of two X variables.

The linear predictor is deterministic or **systematic**. As with the equation for a line, the linear predictor component of a linear model is a function that maps a specific value of X to a value of Y . This mapped value is the **expected value**, or expectation, given a specific input value of X . This is often written as $E[Y|X]$, which is read as “the expected value of Y given X ”, where “given X ” means a specific value of X . This text will often use the word **conditional** in place of “given”. For example, I would read $E[Y|X]$ as “the expected value of Y conditional on X ”. It is important to recognize that $E[Y|X]$ is a **conditional**

mean – it is the mean value of Y when we observe that X has some specific value x (that is $X = x$).

The second line of the specification (8.3) is read as “epsilon is distributed as Normal with mean zero and variance sigma squared”. This line explicitly specifies the distribution of the error component of line 1. The error component of a linear model is a random “draw” from a normal distribution with mean zero and variance σ^2 . The second line shows that the error component of the first line is **stochastic**. Using the error-model specification, we can think of any measurement of Y as an expected value plus some random value sampled from a normal distribution with a specified variance. Because the stochastic part of this specification draws an “error” from a population, I refer to this as the **error-draw** specification of the linear model.

8.1.2 The “conditional draw” specification

A second way of specifying a linear model is using a “sampling model”.

$$y_i \sim N(\mu_i, \sigma^2) \quad (8.4)$$

$$E(Y|X) = \mu \quad (8.5)$$

$$\mu_i = \beta_0 + \beta_1 x_i \quad (8.6)$$

The first line states that the response variable Y is a random variable independently drawn from a normal distribution with mean μ and variance σ^2 . This first line is the **stochastic** part of the statistical model. The second line simply states that μ (the greek letter “mu”) from the first line is the conditional mean (or expectation). The third line states how μ_i is generated given that $X = x_i$. This is the linear predictor, which is the **systematic** (or deterministic) part of the statistical model. It is systematic because the same value of x_i will always generate the same μ_i . Using the sampling-draw specification, we can think of any measurement of Y as a random draw from a specified distribution. Because it is Y and not some “error” that is drawn from a specified distribution, I refer to this as the **conditional-draw** specification of the linear model.

8.1.3 Comparing the two ways of specifying the linear model

These two ways of specifying the model encourage slightly different ways of thinking about how the data (the response variable Y) were generated. The error-draw specification “generates” data by 1) constructing what y_i “should be” given x_i (this is the conditional expectation), then 2) adding some error e_i drawn from a normal distribution with mean zero and some specified variance.

The conditional-draw specification “generates” data by 1) constructing what y_i “should be” given x_i , then 2) drawing a random variable from some specified distribution *whose mean is this expectation*. This random draw is not “error” but the measured value y_i . For the error draw generation, we need only one hat of random numbers, but for the conditional draw generation, we need a hat for each value of x_i .

Here is a short script that generates data by implementing both the error-draw and conditional-draw specifications. See if you can follow the logic of the code and match it to the meaning of these two ways of specifying a linear model.

```

n <- 5
b_0 <- 10.0
b_1 <- 1.2
sigma <- 0.4
x <- 1:n
y_expected <- b_0 + b_1*x

# error-draw. Note that the n draws are all from the same distribution
set.seed(1)
y_error_draw <- y_expected + rnorm(n, mean = 0, sd = sigma)

# conditional-draw. Note that the n draws are each from a different
# distribution because each has a different mean.
set.seed(1)
y_conditional_draw <- rnorm(n, mean = y_expected, sd = sigma)

data.table(X = x,
           "Y (error draw)" = y_error_draw,
           "Y (conditional draw)" = y_conditional_draw)

##      X Y (error draw) Y (conditional draw)
## 1: 1    10.94942    10.94942
## 2: 2    12.47346    12.47346
## 3: 3    13.26575    13.26575
## 4: 4    15.43811    15.43811
## 5: 5    16.13180    16.13180

```

The error-draw specification is not very useful for thinking about data generation for data analyzed by generalized linear models, which are models that allow one to specify distribution families other than Normal (such as the binomial, Poisson, and Gamma families). In fact, thinking about a model as a predictor plus error can lead to the misconception that in a generalized linear model, the error (or residuals from the fit) has a distribution from the non-Normal distribution modeled. This cannot be true because the distributions modeled using generalized linear models (other than the Normal) do not have

8.2. A LINEAR MODEL CAN BE FIT TO DATA WITH CONTINUOUS, DISCRETE, OR CATEGORICAL X VARIABLES

negative values (some residuals must have negative values since the mean of the residuals is zero). Introductory biostatistics textbooks typically only introduce the error-draw specification because introductory textbooks recommend data transformation or non-parametric tests if the data are not approximately normal. This is unfortunate because generalized linear models are extremely useful for real biological data.

Although a linear model (or statistical model more generally) is a model of a data-generating process, linear models are not typically used to actually generate any data. Instead, when we use a linear model to understand something about a real dataset, we think of our data as one realization of a process that generates data like ours. A linear model is a model of that process. That said, it is incredibly useful to use linear models to create fake datasets for at least two reasons: to probe our understanding of statistical modeling generally and, more specifically, to check that a model actually creates data like that in the real dataset that we are analyzing.

8.2 A linear model can be fit to data with continuous, discrete, or categorical X variables

In the linear model fit to the data in Figure 8.1B, the X variable is **continuous**, which can take any real number between the minimum X and maximum X in the data. For biological data, most variables that are continuous are positive, real numbers (a zero is not physically possible but could be recorded in the data if the true value is less than the minimum measurable amount). One exception is a **composition** (the fraction of a total), which can be zero. Negative values can occur with variables in which negative represent a direction (work, electrical potential) or a rate. **Discrete** variables are numeric but limited to certain real numbers. Most biological variables that are discrete are counts, and can be zero, but not negative. **Categorical variables** are non-numeric descriptions of a measure. Many of the categorical variables in this text will be the experimentally controlled treatment variable of interest (the variable *treatment* containing the values “wild type” and “knockout”) but some are measured covariates (the variable *sex* containing the values “female” and “male”).

8.2.1 Fitting linear models to experimental data in which the X variable is continuous or discrete

A linear model fit to data with a numeric (continuous or discrete) X is classical **regression** and the result is typically communicated by a regression line. The experiment introduced in Chapter 9 Linear models with a single, continuous X is a good example. In this experiment, the researchers designed an experiment to measure the effect of warming on the timing of photosynthetic activity.

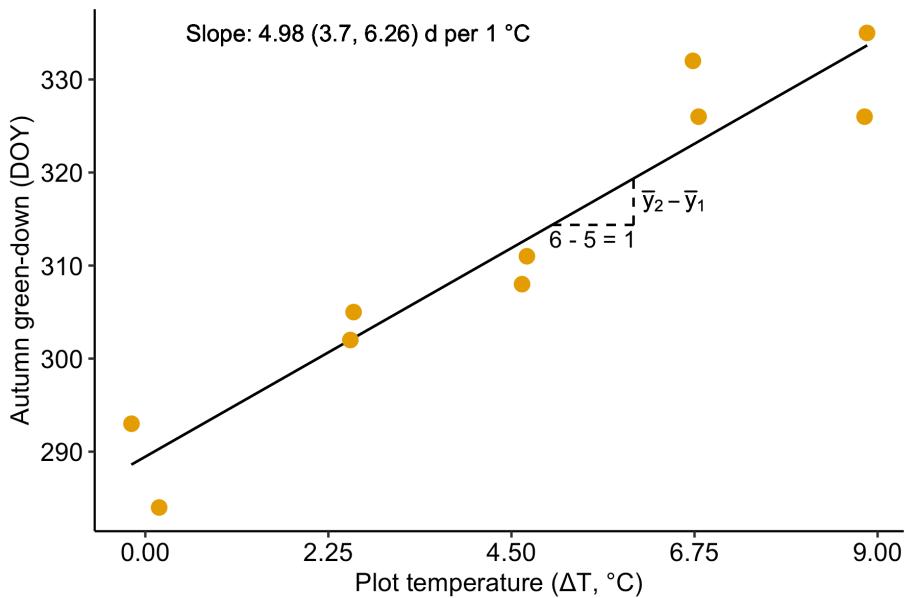


Figure 8.2: Illustration of the slope in a linear model with a numeric X. The slope (the coefficient of X) is the difference in expected value for any two X that are one unit apart. This is illustrated for the points on the line at $x = 5$ and $x = 6$.

Temperature was experimentally controlled at one of five settings (0, 2.25, 4.5, 6.75, or 9 °C above ambient temperature) within twelve, large enclosures. The response variable in the illustrated example is Autumn “green-down”, which is the day of year (DOY) of the transition to loss of photosynthesis. The intercept and slope parameters of the regression line (Figure 8.2) are the coefficients of the linear model. The slope (4.98 days per 1 °C added warmth) estimates the **effect of warming on green-down DOY**. What is not often appreciated at the introductory biostatistics level is that **the slope is a difference in conditional means**. Any point on a regression line is the expected value of Y at a specified value of X , that is, the conditional mean $E(Y|X)$. **The slope is the difference in expected values for a pair of points that differ in X by one unit.**

$$b_1 = E(Y|X = x + 1) - E(Y|X = x - 1) \quad (8.7)$$

I show this in Figure 8.2 using the points on the regression line at $x = 5$ and $x = 6$. Thinking about a regression coefficient as a difference in conditional means is especially useful for understanding the coefficients of a categorical X variable, as described below.

8.2. A LINEAR MODEL CAN BE FIT TO DATA WITH CONTINUOUS, DISCRETE, OR CATEGORICAL X VARIABLE

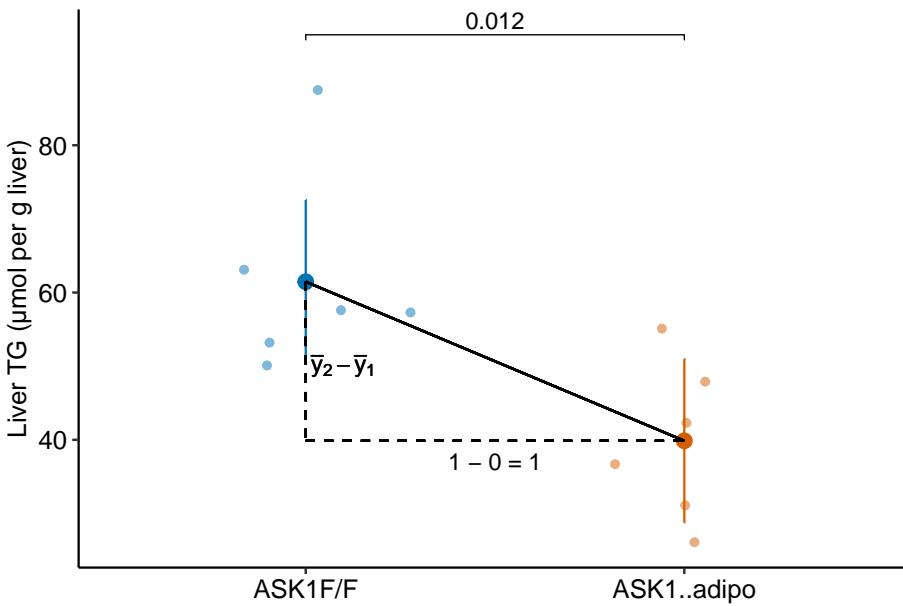


Figure 8.3: Illustration of the slope in a linear model with categorical X. The slope (the coefficient of X) is the difference in conditional means.

8.2.2 Fitting linear models to experimental data in which the X variable is categorical

Linear models can be fit to experimental data in which the X variable is categorical – this is the focus of this text! For the model fit to the data in Figure 8.1B, the coefficient of X is the slope of the line. Perhaps surprisingly, 1) we can fit a model like equation (8.3) to data in which the X variable is categorical and 2) the coefficient of X is a slope. How is this possible? The slope of a line is $\frac{y_2 - y_1}{x_2 - x_1}$ where (x_1, y_1) and (x_2, y_2) are the graph coordinates of any two points on the line. What is the denominator of the slope function $(x_2 - x_1)$ when X is categorical?

The solution to using a linear model with categorical X is to recode the factor levels into numbers. An example of this was outlined in Chapter 2 (Analyzing experimental data with a linear model). The value of X for individual mouse i is a number that indicates the treatment assignment – a value of 0 is given to mice with a functional ASK1 gene and a value of 1 is given to mice with a knocked out gene. The regression line goes through the two group means (Figure 8.3). With the (0, 1) coding, $\bar{x}_{ASK1adipo} - \bar{x}_{ASK1F/F} = 1$, so the denominator of the slope is equal to one and the slope is simply equal to the numerator $\bar{y}_{ASK1adipo} - \bar{y}_{ASK1F/F}$. The coefficient (which is a slope!) is the difference in conditional means.

8.3 Statistical models are used for prediction, explanation, and description

Researchers typically use statistical models to understand relationships between one or more Y variables and one or more X variables. These relationships include

1. Descriptive modeling. Sometimes a researcher merely wants to describe the relationship between Y and a set of X variables, perhaps to discover patterns. For example, the arrival of a spring migrant bird (Y) as a function of sex (X_1) and age (X_2) might show that males and younger individuals arrive earlier. Importantly, if another X variable is added to the model (or one dropped), the coefficients, and therefore, the precise description, will change. That is, the interpretation of a coefficient as a descriptor is *conditional* on the other covariates (X variables) in the model. In a descriptive model, there is no implication of causal effects and the goal is not prediction. Nevertheless, it is very hard for humans to discuss a descriptive model without using causal language, which probably means that it is hard for us to think of these models as *mere description*. Like natural history, descriptive models are useful as patterns in want of an explanation, using more explicit causal models including experiments.
2. Predictive modeling. Predictive modeling is very common in applied research. For example, fisheries researchers might model the relationship between population density and habitat variables to predict which subset of ponds in a region are most suitable for brook trout (*Salvelinus fontinalis*) reintroduction. The goal is to build a model with minimal prediction error, which is the error between predicted and actual values for a future sample. In predictive modeling, the X (“predictor”) variables are largely instrumental – how these are related to Y is not a goal of the modeling, although sometimes an investigator may be interested in the relative importance among the X for predicting Y (for example, collecting the data may be time consuming, or expensive, or environmentally destructive, so know which subset of X are most important for predicting Y is a useful strategy).
3. Explanatory (causal) modeling. Very often, researchers are explicitly interested in *how* the X variables are causally related to Y . The fisheries researchers that want to reintroduce trout may want to develop and manage a set of ponds to maintain healthy trout populations. This active management requires intervention to change habitat traits in a direction, and with a magnitude, to cause the desired response. This model is predictive – a specific change in X predicts a specific response in Y – because the coefficients of the model provide knowledge on how the system functions – how changes in the inputs *cause* change in the output. Causal

interpretation of model coefficients requires a set of strong assumptions about the X variables in the model. These assumptions are typically met in **experimental designs** but not **observational designs**.

With observational designs, biologists are often not very explicit about which of these is the goal of the modeling and use a combination of descriptive, predictive, and causal language to describe and discuss results. Many papers read as if the researchers intend explanatory inference but because of norms within the biology community, mask this intention with “predictive” language. Here, I advocate embracing explicit, explanatory modeling by being very transparent about the model’s goal and assumptions.

8.4 What do we call the X and Y variables?

The inputs to a linear model (the X variables) have many names. In this text, the X variables are typically

- **treatment variables** – this term makes sense only for categorical variables and is often used for variables that are a **factor** containing the treatment assignment (for example “control” and “knockout”)
- **factor variables** (or simply, **factors**) – again, this term makes sense only for categorical variables
- **covariates** – this term is usually used for the non-focal X variables in a statistical model.

A linear model is a regression model and in regression modeling, the X variables are typically called

- **independent variables** (often shortened to IV) – “independent” in the sense that in a statistical model at least, the X are not a function of Y .
- **predictor variables** (or simply, “predictors”) – this makes the most sense in prediction models.
- **explanatory variables** – this term is usually applied in observational designs and is best used if the explicit goal is causal modeling.

In this text, the output of a linear model (the Y variable or variables if the model is multivariate) will most often be called either of

- **response variable** (or simply, “response”)
- **outcome variable** (or simply, “outcome”)

These terms have a causal connotation in everyday English. These terms are often used in regression modeling with observational data, even if the model is not explicitly causal. On other term, common in introductory textbooks, is

- **dependent variable** – “dependent” in the sense that in a statistical model at least, the Y is a function of the X .

8.5 Modeling strategy

A “best practice” sequence of steps used throughout this text to analyze experimental data is

1. **examine the data** using **exploratory plots** to
 - examine individual points and identify outliers that are likely due to data transcription errors or measurement blunders
 - examine **outlier** points that are biologically plausible, but raise red flags about undue influence on fit models. This information is used to inform the researcher on the strategy to handle outliers in the statistical analysis, including algorithms for excluding data or implementation of **robust** methods.
 - provide useful information for initial model filtering (narrowing the list of potential models that are relevant to the question and data). Statistical modeling includes a diverse array of models, yet almost all methods used by researchers in biology, and all models in this book, are generalizations of the linear model specified in (8.6). For some experiments, there may be multiple models that are relevant to the question and data. Model checking (step 3) can help decide which model to ultimately use.
2. **fit the model**, in order to estimate the model parameters and the uncertainty in these estimates.
3. **check the model**, which means to use a series of diagnostic plots and computations of model output to check that the fit model reasonably approximates the data. If the diagnostic plots suggest a poor approximation, then choose a different model and go back to step 2.
4. **inference from the model**, which means to use the fit parameters to learn, with uncertainty, about the system, or to predict future observations, with uncertainty.
5. **plot the model**, which means to plot the data, which may be adjusted, and the estimated parameters (or other results derived from the estimates) with their uncertainty.

Note that step 1 (exploratory plots) is *not* data mining, or exploring the data for patterns to test.

8.6 Predictions from the model

For the linear model specified in Model (8.3), the fit model is

$$y_i = b_0 + b_1 x_i + e_i \quad (8.8)$$

where b_0 and b_1 are the coefficients of the fit model and the e_i are the residuals of the fit model. We can use the coefficients and residuals to recover the y_i , although this would rarely be done. More commonly, we could use the coefficients to calculate conditional means (the mean conditional on a specified value of X).

$$\hat{y}_i = b_0 + b_1 x_i \quad (8.9)$$

The conditional means are typically called **fitted values**, if the X are the X used to fit the model, or **predicted values**, if the X are new. “Predicted values” is often shortened to “the prediction”.

8.7 Inference from the model

If our goal is inference, we want to use the fit parameters to learn, with uncertainty, about the system. Using equation (8.8), the coefficients b_0 and b_1 are **point estimates** of the true, generating parameters β_0 and β_1 , the e_i are estimates of ε_i (the true, biological “noise”), and $\frac{\sum e_i^2}{N-2}$ is an estimate of the true, population variance σ^2 (this will be covered more in chapter xxx but you may recognize that $\frac{\sum e_i^2}{N-2}$ is the formula for a variance). And, using equation (8.9), \hat{y}_i is the point estimate of the parameter μ_i (the true mean conditional on $X = x_i$). Throughout this text, Greek letters refer to a theoretical parameter and Roman letters refer to point estimates.

Our uncertainty in the estimates of the parameters due to sampling is the standard error of the estimate. It is routine to report standard errors of means and coefficients of the model. While a standard error of the estimate of σ is available, this is effectively never reported, at least in the experimental biology literature, presumably because the variance is thought of as a nuisance parameter (noise) and not something worthy of study. This is a pity. Certainly treatments can effect the variance in addition to the mean.

Parametric inference assumes that the response is drawn from some probability distribution (Normal, or Poisson, or Bernouli, etc.). Throughout this text, I emphasize reporting and interpreting point estimates and **interval estimates** of the point estimate. A **confidence interval** is a type of interval estimate. A confidence interval of a parameter is a measure of the uncertainty in the

estimate. A 95% confidence interval has a 95% probability (in the sense of long-run frequency) of containing the parameter. This probability is a property of the population of intervals that could be computed using the same sampling and measuring procedure. It is not correct, without further assumptions, to state that there is a 95% probability that the parameter lies within the interval. Perhaps a more useful interpretation is that the interval is a **compatability interval** in that it contains the range of estimates that are compatible with the data, in the sense that a *t*-test would not reject the null hypothesis of a difference between the estimate and any value within the interval (this interpretation does not imply anything about the true value).

Another kind of inference is a **significance test**, which is the computation of the probability of “seeing the data” or something more extreme than the data, given a specified null hypothesis. This probability is the **p-value**, which can be reported with the point estimate and confidence interval. There are some reasonable arguments made by very influential statisticians that *p*-values are not useful and lead researchers into a quagmire of misconceptions that impede good science. Nevertheless, the current methodology in most fields of Biology have developed in a way to become completely dependent on *p*-values. I think at this point, a *p*-value can be a useful, if imperfect tool in inference, and will show how to compute *p*-values throughout this text.

Somewhat related to a significance test is a hypothesis test, or a **Null-Hypothesis Signficance Test** (NHST), in which the *p*-value from a significance test is compared to a pre-specified error rate called α . Hypothesis testing was developed as a formal means of decision making but this is rarely the use of NHST in experimental biology. For almost all applications of *p*-values that I see in the literature that I read in ecology, evolution, physiology, and wet-bench biology, comparing a *p*-value to α adds no value to the communication of the results.

8.7.1 Assumptions for inference with a statistical model

1. The data were generated by a process that is “linear in the parameters”, which means that the different components of the model are added together. This additive part of the model containing the parameters is the linear predictor in specifications (8.3) and (8.6) above. For example, a cubic polynomial model

$$\mathbb{E}(Y|X) = \beta_0 + \beta_1 X + \beta_2 X^2 + \beta_3 X^3 \quad (8.10)$$

is a linear model, even though the function is non-linear, because the different components are added. Because a linear predictor is additive, it can be compactly defined using matrix algebra

$$\mathbb{E}(Y|X) = \mathbf{X}\boldsymbol{\beta} \quad (8.11)$$

where \mathbf{X} is the **model matrix** and $\boldsymbol{\beta}$ is the vector of parameters. We discuss these more in chapter xxx.

A **Generalized Linear Model** (GLM) has the form $g(\mu_i) = \eta_i$ where η (the Greek letter “eta”) is the linear predictor

$$\eta = \mathbf{X}\boldsymbol{\beta} \quad (8.12)$$

GLMs are extensions of linear models. There are non-linear models that are not linear in the parameters, that is, the predictor is not a simple dot product of the model matrix and a vector of parameters. For example, the Michaelis-Menten model is a non-linear model

$$\mathbb{E}(Y|X) = \frac{\beta_1 X}{\beta_2 + X} \quad (8.13)$$

that is non-linear in the parameters because the parts are not added together. This text covers linear models and generalized linear models, but not non-linear models that are also non-linear in the parameters.

2. The draws from the probability distribution are **independent**. Independence implies **uncorrelated** Y conditional on the X , that is, for any two Y with the same value of X , we cannot predict the value of one given the value of the other. For example, in the ASK1 data above, “uncorrelated” implies that we cannot predict the glucose level of one mouse within a specific treatment combination given the glucose level of another mouse in that combination. For linear models, this assumption is often stated as “independent errors” (the ε in model (8.3)) instead of independent observations.

There are lots of reasons that conditional responses might be correlated. In the mouse example, correlation within treatment group could arise if subsets of mice in a treatment group are siblings or are housed in the same cage. More generally, if there are measures both within and among experimental units (field sites or humans or rats) then we’d expect the measures within the same unit to err from the model in the same direction. Multiple measures within experimental units (a site or individual) creates “clustered” observations. Lack of independence or clustered observations can be modeled using models with **random effects**. These models go by many names including linear mixed models (common in Ecology), hierarchical models, multilevel models, and random effects models. A linear mixed model is a variation of model (8.3). This text introduces linear mixed models in chapter xxx.

Measures that are taken from sites that are closer together or measures taken closer in time or measures from more closely related biological species will tend to have more similar values than measures taken from sites that are further apart or from times that are further apart or from species that are less closely related. Space and time and phylogeny create **spatial and temporal and phylogenetic autocorrelation**. Correlated error due to space or time or phylogeny can be modeled with **Generalized Least Squares** (GLS) models. A GLS model is a variation of model (8.3).

8.7.2 Specific assumptions for inference with a linear model

1. **Constant variance** or **homoskedasticity**. The most common way of thinking about this is the error term ε has constant variance, which is a short way of saying that random draws of ε in model (8.3) are all from the same (or **identical**) distribution. This is explicitly stated in the second line of model specification (8.3). If we were to think about this using model specification (8.6), then homoskedasticity means that σ in $N(\mu, \sigma)$ is constant for all observations (or that the *conditional* probability distributions are identical, where *conditional* would mean adjusted for μ)

Many biological processes generate data in which the error is a function of the mean. For example, measures of biological variables that grow, such as lengths of body parts or population size, have variances that “grow” with the mean. Or, measures of counts, such as the number of cells damaged by toxin, the number of eggs in a nest, or the number of mRNA transcripts per cell have variances that are a function of the mean. Heteroskedastic error can be modeled with **Generalized Least Squares**, a generalization of the linear model, and with **Generalized Linear Models** (GLM), which are “extensions” of the classical linear model.

2. Normal or **Gaussian** probability distribution. As above, the most common way of thinking about this is the error term ε is Normal. Using model specification (8.6), we’d say the conditional probability distribution of the response is normal. A normal probability distribution implies that 1) the response is continuous and 2) the conditional probability is symmetric around μ_i . If the conditional probability distribution has a long left or right tail it is **skewed** left or right. Counts (number of cells, number of eggs, number of mRNA transcripts) and binary responses (successful escape or successful infestation of host) are not continuous and often often have asymmetric probability distributions that are skewed to the right and while sometimes both can be reasonably modeled using a linear model they are more often modeled using generalized linear models, which, again, is an extension of the linear model in equation (8.6). A classical linear model is a specific case of a GLM.

A common misconception is that inference from a linear model assumes that the raw response variable is normally distributed. Both the error-draw and conditional-draw specifications of a linear model show precisely why this conception is wrong. Model (??) states explicitly that it is the error that has the normal distribution – the distribution of Y is a mix of the distribution of X and the error. Model (8.6) states that the conditional outcome has a normal distribution, that is, the distribution after adjusting for variation in X .

8.8 “linear model,”regression model“, or”statistical model”?

Statistical modeling terminology can be confusing. The X variables in a statistical model may be quantitative (continuous or integers) or categorical (names or qualitative amounts) or some mix of the two. Linear models with all quantitative independent variables are often called “regression models.” Linear models with all categorical independent variables are often called “ANOVA models.” Linear models with a mix of quantitative and categorical variables are often called “ANCOVA models” if the focus is on one of the categorical X or “regression models” if there tend to be many independent variables.

This confusion partly results from the history of the development of regression for the analysis of observational data and ANOVA for the analysis of experimental data. The math underneath classical regression (without categorical variables) is the linear model. The math underneath classical ANOVA is the computation of sums of squared deviations from a group mean, or “sums of squares”. The basic output from a regression is a table of coefficients with standard errors. The basic ouput from ANOVA is an ANOVA table, containing the sums of squares along with mean-squares, F -ratios, and p -values. Because of these historical differences in usage, underlying math, and output, many textbooks in biostatistics are organized around regression “vs.” ANOVA, presenting regression as if it is “for” observational studies and ANOVA as if it is “for” experiments.

It has been recognized for many decades that experiments can be analyzed using the technique of classical regression if the categorical variables are coded as numbers (again, this will be explained later) and that both regression and ANOVA are variations of a more general, linear model. Despite this, the “regression vs. ANOVA” way-of-thinking dominates the teaching of biostatistics.

To avoid misconceptions that arise from thinking of statistical analysis as “regression vs. ANOVA”, I will use the term “linear model” as the general, umbrella term to cover everything in this book. By linear model, I mean any model that is linear in the parameters, including classical regression models, marginal models, linear mixed models, and generalized linear models. To avoid repetition, I’ll also use “statistical model”.

Chapter 9

Linear models with a single, continuous X

1. observation warming on phenology
2. experiment warming on phenology
3. using regression to compare longitudinal (dietary methionine)

9.1 A linear model with a single, continuous X is classical “regression”

9.1.1 Analysis of “green-down” data

To introduce some principles of modeling with a single continuous X variable, I'll use a dataset from

Richardson, A.D., Hufkens, K., Milliman, T. et al. Ecosystem warming extends vegetation activity but heightens vulnerability to cold temperatures. *Nature* 560, 368–371 (2018).

Source data

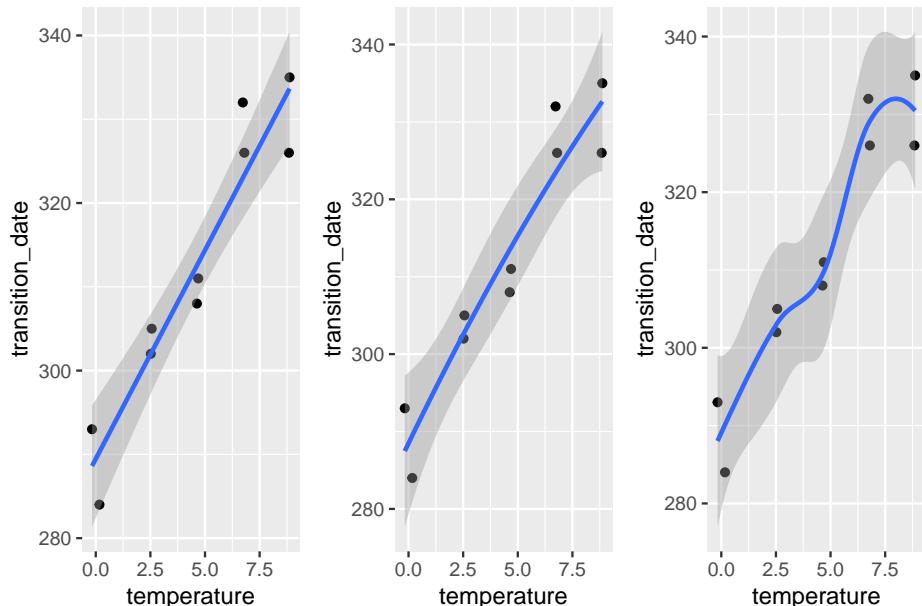
The data are from a long-term experiment on the effects of warming and CO₂ on a high-carbon northern temperate peatland and is the focal dataset of the study. The experiment involves 10 large, temperature and CO₂ controlled enclosures. CO₂ is set to 400 ppm in five enclosures and 900 ppm in five enclosures. Temperature of the five enclosures within each CO₂ level is set to 0, 2.25, 4.5, 6.75, or 9 °C above ambient temperature. The multiple temperature levels is a **regression design**, which allows a researcher to measure non-linear effects. Read more about the experimental design and the beautiful implementation.

The question pursued is in this study is, what is the causal effect of warming on the timing (or phenology) of the transition into photosynthetic activity (“green-up”) in the spring and of the transition out of photosynthetic activity (“green-down”) in the fall? The researchers measured these transition dates, or Day of Year (DOY), using foliage color. Here, we focus on the transition out of photosynthesis or “green-down” DOY.

Import the data

1. Examine the data

```
gg1 <- qplot(x = temperature,
              y = transition_date,
              data = fig2c) +
  geom_smooth(method = "lm")
gg2 <- qplot(x = temperature,
              y = transition_date,
              data = fig2c) +
  geom_smooth(method = "lm", formula = y ~ poly(x, 2))
gg3 <- qplot(x = temperature,
              y = transition_date,
              data = fig2c) +
  geom_smooth()
plot_grid(gg1, gg2, gg3, ncol=3)
```



No plot shows any obvious outlier that might be due to measurement blunders or curation error. The linear regression in the left-most plot clearly shows that

a linear response is sufficient to capture the effect of temperature on day of green-down.

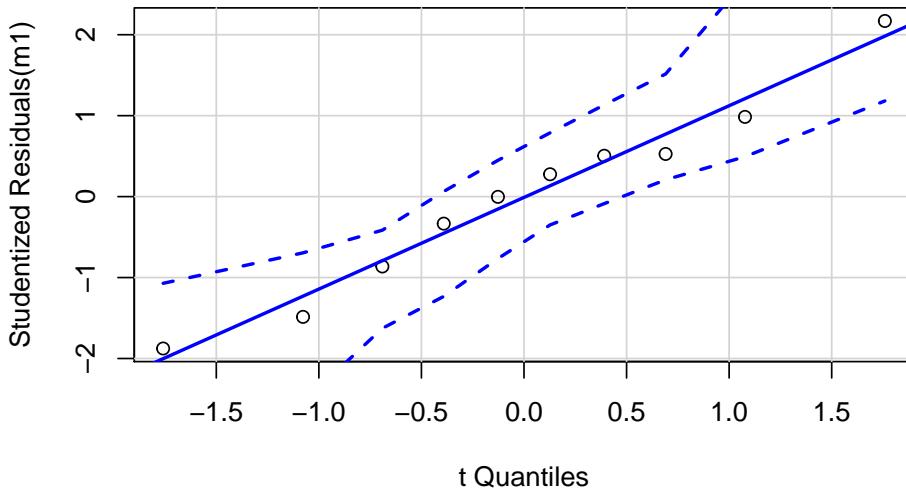
2. **choose a model.** Because the X variable (*temperature*) was experimentally set to five levels, the data could reasonably be modeled using either a linear model with a categorical X or a linear model with a continuous X . The advantage of modeling *temperature* as a continuous variable is that there is only one effect, the slope of the regression line. If modeled as a categorical factor with five levels, there are, at a minimum, four interesting effects (the difference in means between each non-reference level and reference (*temperature* = 0) level). Also, for inference, modeling *temperature* as a continuous variable increases power for hypothesis tests.

3. fit the model

```
# Step 1: fit the model
m1 <- lm(transition_date ~ temperature, data = fig2c)
```

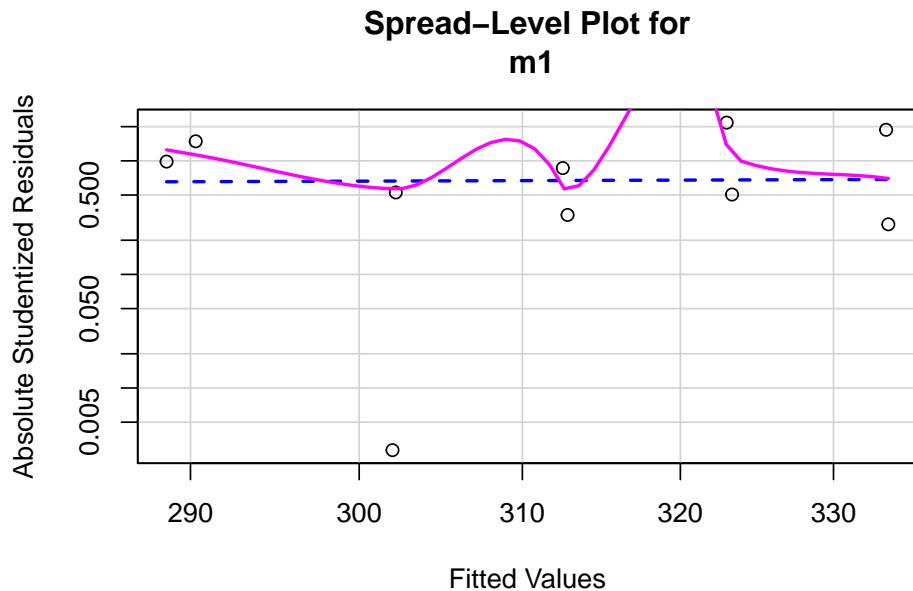
4. check the model

```
# check normality assumption
set.seed(1)
qqPlot(m1, id=FALSE)
```



The Q-Q plot indicates the distribution of residuals is well within that expected for a normal sample and there is no cause for concern with inference.

```
# check homogeneity assumption
spreadLevelPlot(m1, id=FALSE)
```



```
##
## Suggested power transformation: 0.6721303
```

The spread-location plot shows no conspicuous trend in how the spread changes with the conditional mean. There is no cause for concern with inference.

5. inference from the model

```
m1_coeff <- summary(m1) %>%
  coef()
m1_confint <- confint(m1)
m1_coeff <- cbind(m1_coeff, m1_confint)
m1_coeff

##           Estimate Std. Error   t value    Pr(>|t|)    2.5 %
## (Intercept) 289.458750  3.0593949 94.613071 1.738650e-13 282.403773
## temperature  4.982745  0.5541962  8.990941 1.866888e-05  3.704767
##             97.5 %
## (Intercept) 296.513728
## temperature  6.260724
```

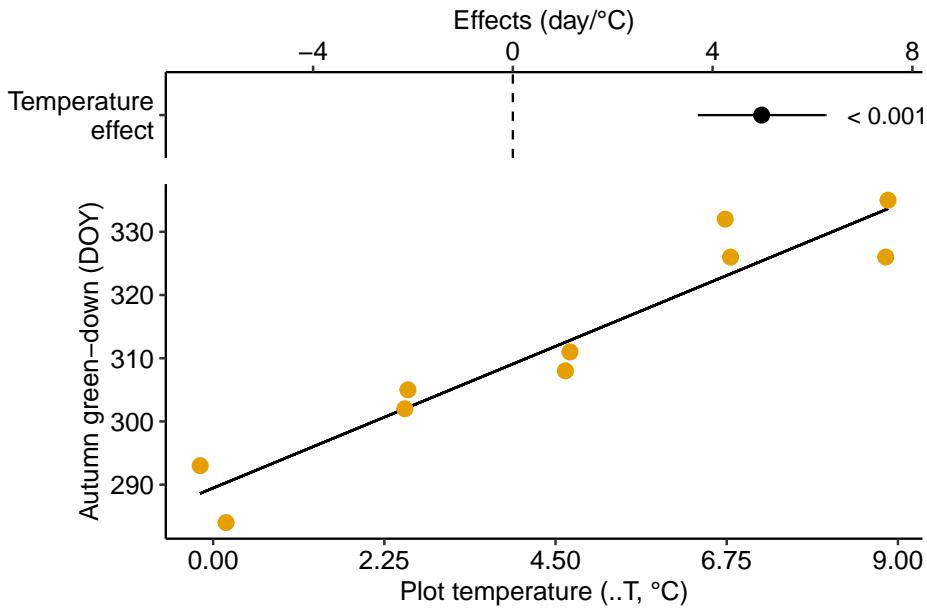


Figure 9.1: Modification of the published Figure 2c showing the experimental effect of warming on the date of autumn green-down (the transition to fall foliage color) in a mixed shrub community. The bottom panel is a scatterplot. The regression line shows the expected value of Y (transition day of year) given a value of X (added temperature). The slope of the regression line is the estimate of the effect. The estimate and 95% confidence interval of the estimate are given in the top panel.

The effect of added temperature on the day of green-down is 4.98 d per 1 °C (95% CI: 3.7, 6.3; $p < 0.001$).

6. plot the model

7. Report the results

The modeled effect of added temperature is Slope: 4.98 (3.7, 6.26) d per 1 °C (9.1).

9.1.2 Learning from the green-down example

Figure 9.1 is a **scatterplot** with the green-down DOY for the mixed-shrub community on the Y axis and added temperature on the X axis. The line

through the data is a regression line, which is the expected value of Y (green-down DOY) given a specific value of X (added temperature). The slope of the line is the **effect** of added temperature on timing of green-down. The intercept of the regression line is the value of the response (day of green-down) when X is equal to zero. Very often, this value is not of interest although the value should be reported to allow predictions from the model. Also very often, the value of the intercept is not meaningful because a value of $X = 0$ is far outside the range of measured X , or the value is absurd because it is impossible (for example, if investigating the effect of body weight on metabolic rate, the value $weight = 0$ is impossible).

The intercept and slope are the coefficients of the model fit to the data, which is

$$day_i = b_0 + b_1 \text{temperature}_i + e_i \quad (9.1)$$

where day is the day of green-down, temperature is the added temperature, and i refers (or “indexes”) the i th enclosure. This model completely reconstructs the day of green-down for all ten enclosures. For example, the day of green-down for enclosure 8 is

$$332 = 289.458750 + 4.982745 \times 6.73 + 9.00737 \quad (9.2)$$

The coefficients in the model are estimates of the parameters of the **generating model** fit to the data

$$day = \beta_0 + \beta_1 \text{temperature} + \varepsilon \quad (9.3)$$

$$\varepsilon \sim N(0, \sigma^2) \quad (9.4)$$

A generating model of the data is used to make inference, for example, a measure of uncertainty in a prediction in the timing of green-down with future warming, or a measure of uncertainty about the effect of temperature on green-down.

9.1.3 Using a regression model for “explanation” – causal models

In this text, “explanatory” means “causal” and a goal of explanatory modeling is the estimate of causal effects using a **causal interpretation** of the linear model (=regression) coefficients. “What what? I learned in my stats 101 course that we cannot interpret regression coefficients causally”.

Statisticians (and statistics textbooks) have been quite rigid that a regression coefficient has a descriptive (or “observational”, see below) interpretation but

not a causal interpretation. At the same time, statisticians (and statistics textbooks) do not seem to have any issue with interpreting the modeled effects from an experiment causally, since this is how they are interpreted. But if the modeled effects are simply coefficients from a linear (= regression) model, then this historical practice is muddled.

Part of this muddled history arises from the use of “regression” for models fit to observational data with one or more continuous X variables and the use of “ANOVA” for models fit to experimental data with one or more categorical X . This separation seems to have blinded statisticians from working on the formal probabilistic statements underlying causal interpretations of effect estimates in ANOVA and the synthesis of these statements with the probabilistic statements in regression modeling. Two major approaches to developing formal, probabilistic statements of causal modeling in statistics are the **Rubin causal model** and the **do-operator** of Pearl. Despite the gigantic progress in these approaches, little to none of this has found its way into biostatistics textbooks.

9.1.3.1 What a regression coefficient means

A linear (“regression”) model coefficient, such as the coefficient for temperature, β_1 , has two interpretations, an **observational** interpretation and a **causal interpretation**. To understand these interpretations, it’s useful to remember that a predicted value from a regression model is a **conditional mean**

$$E[day|temperature] = \beta_0 + \beta_1 temperature \quad (9.5)$$

In words “the expected value of day conditional on temperature is beta-knot plus beta-one times temperature”. An **expected value** is a long run average – if we had an infinite number of enclosures with $temperature = x$ (where x is a specific value of added temperature, say 2.5 °C), the average *day* of these enclosures is $\beta_0 + \beta_1 x$.

The parameter β_1 is a *difference* in conditional means.

$$\beta_1 = E[day|temperature = x + 1] - E[day|temperature = x] \quad (9.6)$$

In words, “beta-one is the expected value of day of green-down when the temperature equals $x + 1$ minus the expected value of day of green-down when the temperature equals x .” A very short way to state this is “beta-one is a difference in conditional means”.

tl;dr. Note that the “+ 1” in this definition is mere convenience. Since the slope of a line is $\frac{y_2 - y_1}{x_2 - x_1}$, where (x_1, y_1) and (x_2, y_2) are the coordinates of any two points on the line, it is convenient to choose two points that differ in X by one unit, which makes the fraction equal to the numerator only. The numerator

is a difference in conditional means. It is also why the units of a regression coefficient are "per unit of X even if defined as a difference in two Y values.

The difference between observational and causal interpretations of β_1 depends on the "event" conditioned on in $E[\text{day}|\text{temperature}]$. Let's start with the causal interpretation, which is how we should think about the regression coefficients in the green-down experiment.

9.1.3.2 Causal interpretation – conditional on "doing" $\mathbf{X} = \mathbf{x}$

In the causal interpretation of regression, $E[\text{day}|\text{temperature}]$ is conditioned on "doing" a real or hypothetical intervention in the system where we set the value of *temperature* to a specific value x ("temperature = x), while keeping everything else about the system the same. This can be stated explicitly as $E[\text{day}|\text{do temperature} = x]$. Using the do-operator, we can interpret β_1 as an **effect coefficient**.

$$\beta_1 = E[\text{day}|\text{do temperature} = x + 1] - E[\text{day}|\text{do temperature} = x] \quad (9.7)$$

In words, "beta-one is the expected value of day of green-down were we to set the temperature to $x + 1$, minus the expected value of day of green-down were we to set the temperature to x ."

tl;dr. In the green-down experiment, the researchers didn't set the temperature in the intervened enclosures to the ambient temperature + 1 but to ambient + 2.25, ambient + 4.5, ambient + 6.75, and ambient + 9.0. Again (see tl;dr above), the + 1 is mere convenience in the definition.

9.1.3.3 Observational interpretation – conditional on "seeing" $\mathbf{X} = \mathbf{x}$.

In the observational interpretation of regression, $E[\text{day}|\text{temperature}]$ is conditioned on sampling data and "seeing" a value of *temperature*. We can state this explicitly as $E[\text{day}|\text{see temperature}]$. From this, we can interpret β_1 as an **observational coefficient**

$$\beta_1 = E[\text{day}|\text{see temperature} = x + 1] - E[\text{day}|\text{see temperature} = x] \quad (9.8)$$

In words, "beta-one is the expected value of day of green-down when see that temperature equals $x + 1$ minus the expected value of day of green-down when we see that temperature equals x ." To understand what I mean by "observational", let's imagine that the green-down data do not come from an experiment

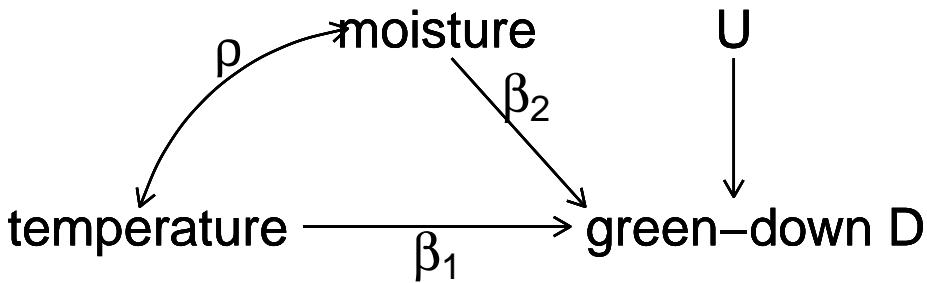


Figure 9.2: a Directed Acyclic (or causal) Graph of a hypothetical world where the day of green-down is caused by two, correlated environmental variables, temperature and moisture, and to a noise factor (U) that represents an unspecified set of additional variables that are not correlated to either temperature or moisture.

in which the researchers intervened and set the added temperature to a specific value but from ten sites that naturally vary in mean annual temperature. Or from a single site with 10 years of data, with some years warmer and some years colder. Data from this kind of study is **observational** – the researcher didn’t intervene and set the X values but merely observed the X values.

If we sample (or “see”) a site with a mean annual temperature that is 2.5 °C above a reference value, the expected day of green-down is $E[\text{day}|\text{temperature} = 2.5^\circ\text{C}]$. That is, values near $E[\text{day}|\text{temperature} = 2.5^\circ\text{C}]$ are more probable than values away from $E[\text{day}|\text{temperature} = 2.5^\circ\text{C}]$. Or, if the only information that we have about this site is a mean annual temperature that is 2.5 °C above some reference, then our best prediction of the day of green-down would be $E[\text{day}|\text{temperature} = 2.5^\circ\text{C}]$. We do not claim that the 4.98 day delay in green-down is caused by the warmer temperature, only that this is the expected delay relative to the reference having seen the data.

The seeing interpretation of a regression coefficient is **descriptive**– it is a description of a mathematical relationship. In this interpretation, the coefficient is not causal in the sense of what the expected **response** in Y would be were we to intervene in the system and change X from x to $x + 1$.

9.1.3.4 Omitted variable bias

What is the consequence of interpreting a regression coefficient causally instead of observationally?

Let’s expand the thought experiment where we have an observational data set of green down dates. In this thought experiment, only two variables systematically affect green-down DOY. The first is the temperature that the plants experience; the effect of *temperature* is β_1 . The second is the soil moisture that

the plants experience; the effect of *moisture* is β_2 . *temperature* and *moisture* are correlated with a value ρ . This causal model is graphically represented by the **causal graph** above.

Lets fit two linear models.

$$(\mathcal{M}_1) \text{ day} = b_0 + b_1 \text{ temperature} + b_2 \text{ moisture} + \varepsilon \quad (9.9)$$

$$(\mathcal{M}_2) \text{ day} = b_0 + b_1 \text{ temperature} + \varepsilon \quad (9.10)$$

\mathcal{M}_1 the linear model including both *temperature* and *moisture* as X variables and \mathcal{M}_2 the linear model including only *temperature* as an X variable. Given the true causal model above, b_1 is an **unbiased** estimate of the true causal effect of temperature β_1 in \mathcal{M}_1 because the expected value of b_1 is β_1 . By contrast, b_1 is a **biased** estimate of the true causal effect of temperature β_1 in \mathcal{M}_2 . The expected value of b_1 in \mathcal{M}_2 is the true, causal effect plus a bias term.

$$\mathbb{E}(b_1|\mathcal{M}_1) = \beta + \rho \alpha \frac{\sigma_{\text{moisture}}}{\sigma_{\text{temperature}}} \quad (9.11)$$

This bias ($\rho \alpha \frac{\sigma_{\text{moisture}}}{\sigma_{\text{temperature}}}$) is **omitted variable bias**. The omitted variable *moisture* is an unmeasured, **confounding variable**. A variable X_2 is a confounder for variable X_1 if X_2 has both a correlation with X_1 and a path to the response Y that is not through X_1 . With omitted variable bias, as we sample more and more data, our estimate of the effect doesn't converge on the truth but the truth plus some bias.

9.1.3.5 Causal modeling with experimental versus observational data

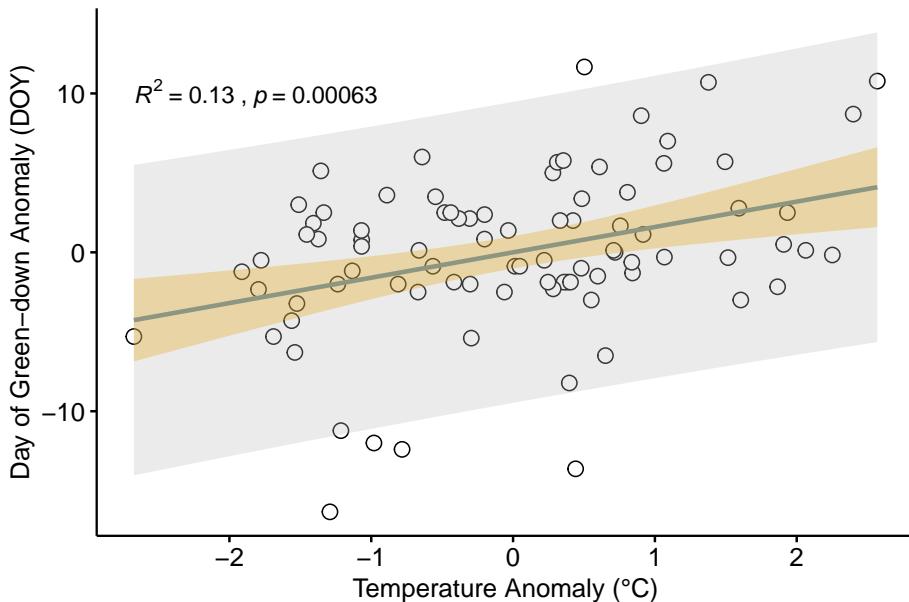
Causal interpretation requires conditioning on “doing $X=x$ ”. For the green-down data, “doing $X=x$ ” is achieved by the random treatment assignment of the enclosures. How does random treatment assignment achieve this? Look again at Figure 9.2. If the values of *treatment* are randomly assigned, then, by definition, the expected value of the correlation between *treatment* and *moisture* is zero. In fact, the expected value of the correlation between *treatment* and any measurable variable at the study site is zero. Given, this, the expected value of the regression coefficient for *temperature* (b_1) is β because $\rho = 0$. That is, the estimate of the true effect is unbiased. It doesn't matter if *moisture*, or any other variable, is excluded from the model. (That said, we may want to include *moisture* or other variables in the model to increase precision of the causal effect. This is addressed in the chapter “Models with Covariates”). This is what an experiment does and why experiments are used to answer causal questions.

Can we use observational data for causal modeling? Yes, but the methods for this are outside of the scope of this text. The mathematical foundation for this

is known as **path analysis**, which was developed by geneticist and evolutionary biologist Sewell Wright (I include this because this work has inspired much of how I think about biology and because he is both my academic grandfather and great-grandfather). Causal analysis of observational data in biology is rare outside of ecology and epidemiology. Good starting points for the modern development of causal analysis are Hernán MA and Robins JM (2020) and Burgess et al. (2018). A gentle introduction is Pearl and Mackenzie (2018).

9.1.4 Using a regression model for prediction – prediction models

The researchers in the green-down study also presented estimates of the effect of temperature on green-down using two observational datasets. Let’s use the one in Extended Data Figure 3d to explore using a regression model for prediction. The data are taken from measurements of the day of green-down over an 86 year period at a single site. The response variable is *green_down_anomaly* (the difference between the day of green down for the year and the mean of these days over the 86 years). The predictor variable is *autumn_temp_anomaly* (the difference between the mean temperature over the year and the mean of these means).



The plot in figure ?? shows the regression line of a linear model fit to the data. Two statistics are given in the plot.

1. The *p*-value of the slope (the coefficient b_1 of *autumn_temp_anomaly*)

2. The R^2 of the model fit.

In addition, two intervals are shown.

1. 95% **confidence interval** of the expected values in blue. The width of this band is a measure of the precision of the estimates of expected values.
2. 95% **prediction interval** of the predictions in gray. The width of this band is a measure of how well the model predicts.

It is important that researchers know what each of these statistics and bands are, when to compute them and when to ignore them, and what to say about each when communicating the results.

9.1.4.1 95% CI and p -value

Both the 95% confidence interval of the expected values and the p -value are a function of the standard error of the slope coefficient b_1 and so are complementary statistics. The p -value is the probability of sampling the null that results in a t -value as or more extreme than the observed t for the coefficient b_1 . This null includes the hypothesis $\beta_1 = 0$. The 95% confidence interval of the expected values is the band containing expected values (mean *green_down_anomaly* conditional on *autumn_temp_anomaly*) that are compatible with the data. There are a couple of useful ways of thinking about this band.

1. The band captures an infinite number of regression lines that are compatible with the data. Some are more steep and predict smaller expected values at the low end of *autumn_temp_anomaly* and higher expected values at the high end of *autumn_temp_anomaly*. Others are less steep and predict higher expected values at the low end of *autumn_temp_anomaly* and lower expected values at the high end of *autumn_temp_anomaly*.
2. The band captures the 95% CI of the conditional mean at every value of X . Consider the 95% CI of the conditional mean at the mean value of X . As we move away from this mean value (lower to higher X), the 95% CI of the conditional mean increases.

A 95% CI and p -value are useful statistics to report if the purpose is *causal modeling*, as in the example above using the experimental green-down data (where the 95% CI was not presented as a confidence band of the expected values but a CI of the estimate of the effect of added temperature). A 95% CI and p -value are also useful statistics to report if the purpose is *descriptive modeling*, simply wanting to know how the conditional mean of the response is related to an X variable.

9.1.4.2 95% prediction interval and R^2

Both the R^2 and the 95% prediction interval are a function of the *population* variability of *green_down_anomaly* conditional on *autumn_temp_anomaly* (the spread of points along a vertical axis about the regression line) and are complimentary statistics. Briefly,

1. the R^2 is a measure of the fraction of the variance in the response that is accounted by the model (some sources say “explained by the model” but this is an odd use of “explain”). It is a number between 0 and 1 and is a measure of “predictability” if the goal is prediction modeling.
2. The 95% prediction interval will contain a new observation 95% of the time. It provides bounds on a **prediction** – given a new observation, there is a 95% probability that the interval at x_{new} will contain y_{new} .

To understand R^2 and the 95% prediction interval a bit better, let’s back up.

$$green_down_anomaly_i = b_0 + b_1 autumn_temp_anomaly_i + e_i \quad (\#eq : doy_fit) \quad (9.12)$$

Model @ref(eq:doymodel) recovers the measured value of *green_down_anomaly* for any year, given the *autumn_temp_anomaly* for the year. The equation includes the linear predictor ($b_0 + b_1 autumn_temp_anomaly_i$) and the **residual** from the predictor (e_i). The predictor part of @ref(eq:doy_fit) is used to compute \hat{y} (“y hat”).

$$\hat{y}_i = b_0 + b_1 autumn_temp_anomaly_i \quad (\#eq : doyhat) \quad (9.13)$$

The \hat{y} are **fitted values**, if the values are computed from the data that were used for the fit, or **predicted values** (or simply the **prediction**), if the values are computed from values of the predictor variables outside the set used to fit the model. For the purpose of plotting, generally, with models with categorical factors as X , I prefer either **modeled values** or **conditional means** to fitted values.

9.1.4.3 R^2

A good model accounts for a sizable fraction of the total variance in the response. This fraction is the R^2 value, which is given in `summary(m1)` and accessed directly with

```
summary(m1)$r.squared
```

```
## [1] 0.1305754
```

R^2 is the fraction of the total variance of Y that is generated by the linear predictor.

$$R^2 = \frac{\text{VAR}(\hat{y})}{\text{VAR}(y)} \quad (9.14)$$

```
yhat <- fitted(m1)
y <- efig_3d[, green_down_anomaly]
var(yhat)/var(y)
```

```
## [1] 0.1305754
```

R^2 will vary from zero (the model accounts for nothing) to one (the model accounts for everything). R^2 is often described as the fraction of total variation *explained* by the model but the usage of “explain” is observational and not causal. Because of the ambiguous usage of “explain” in statistics, I prefer to avoid the word.

It can be useful sometimes to think of R^2 in terms of **residual error**, which is the variance of the residuals from the model. The larger the residual error, the smaller the R^2 , or

$$R^2 = 1 - \frac{\text{VAR}(e)}{\text{VAR}(y)} \quad (9.15)$$

```
e <- residuals(m1)
y <- efig_3d[, green_down_anomaly]
1 - var(e)/var(y)
```

```
## [1] 0.1305754
```

The smaller the residuals, the higher the R^2 and the closer the predicted values are to the measured values. The sum of the model variance and residual variance equals the total variance and, consequently, R^2 is a signal to signal + noise ratio. The noise in R^2 is the sampling variance of the individual measures. The noise in a t -value is the sampling variance of the parameter (for m1, this is the sampling variance of b_1). This is an important distinction because it means that t and R^2 are not related 1:1. In a simple model with only a single X , one might expect R^2 to be big if the p -value for the slope is tiny, but this isn’t necessarily true because of the different meaning of noise in each. A study with a very large sample size n can have a tiny p -value and a small R^2 . A p -value is not a good indicator of predictability. R^2 is. This is explored more below.

9.1.4.4 Prediction interval

A good prediction model has high predictability, meaning the range of predicted values is narrow. A 95% CI is a reasonable range to communicate. For any decision making using prediction, it is better to look at numbers than a band on a plot.

Autumn Temp Anomaly ($^{\circ}\text{C}$)

Expected (days)

2.5% (days)

97.5% (days)

0.5

0.8

-8.7

10.3

1.0

1.6

-7.9

11.1

1.5

2.4

-7.2

12.0

2.0

3.2

-6.4

12.8

2.5

4.0

-5.7

13.7

Given these data and the fit model, if we see a $2 ^{\circ}\text{C}$ increase in mean fall temperature, we expect the autumn green-down to extend 3.2 days. This expectation is an average. We'd expect 95% of actual values to range between -6.4 days

and 12.8 days. This is a lot of variability. Any prediction has a reasonable probability of being over a week early or late. This may seem surprising given the p -value of the fit, which is 0.0006. But the p -value is not a reliable indicator of predictability. It is a statistic related to the blue band, not the gray band.

To understand this prediction interval a bit more, and why a p -value is not a good indicator of predictability, it's useful to understand what makes a prediction interval “wide”. The width of the prediction interval is a function of two kinds of variability

1. The variance of the expected value at a specific value of X . This is the square of the standard error of b_1 . The blue band is communicating the CI based on this variance. The p -value is related to the width of this band.
2. The variance of Y at a specific value of X . This variance is σ^2 . It is useful for learning to think about the equation for the estimate of this variance.

$$\hat{\sigma}^2 = \frac{\sum e_i^2}{N - 2} \quad (9.16)$$

Again, e_i is the residual for the i th case. The denominator ($N - 2$) is the degrees of freedom of the model. Computing $\hat{\sigma}^2$ manually helps to insure that we understand what is going on.

```
summary(m1)$sigma # R's calculation of sigma hat

## [1] 4.736643

df <- summary(m1)$df[2] # R's calculation of df. Check that this is n-2!
sqrt(sum(e^2)/df)

## [1] 4.736643
```

Remember that an assumption of the linear models that we are working with at this point is, this variance is constant for all values of X , so we have a single σ . Later, we will cover linear models that model heterogeneity in this variance. σ is a function of variability in the population – it is the population standard deviation conditional on X .

Importantly, predictability is a function of both these components of variability. As a consequence, it is R^2 , and not the p -value, that is the indicator of predictability. In the observational green-down data, even if we had thousands of years of data, we would still have a pretty low R^2 because of the population variability of *green_down_anomaly* at a given *autumn_temp_anomaly*.

9.1.4.5 Median Absolute Error and Root Mean Square Error are absolute measures of predictability

A p -value is not *at all* a good guide to predictability. R^2 is proportional to predictability but is not really useful in any absolute sense. If we want to predict the effect of warming on the day of green-down, I would like to have a measure of predictability in the units of green-down, which is days. The prediction interval gives this for any value of X . But what about an overall measure of predictability?

Three overall measures of predictability are

1. $\hat{\sigma}$, the estimate of σ . This is the standard deviation of the sample conditional on X .
2. $RMSE$, the root mean squared error. This is

$$SSE = \sum (y_i - \hat{y}_i)^2 \quad (9.17)$$

$$MSE = \frac{SSE}{N} \quad (9.18)$$

$$RMSE = \sqrt{MSE} \quad (9.19)$$

SSE (“sum of squared error”) is the sum of the squared residuals $(y_i - \hat{y}_i)$ of the model. MSE (“mean squared error”) is the mean of the squared errors. $RMSE$ is the square root of the mean squared error. $RMSE$ is almost equal to $\hat{\sigma}$. The difference is the denominator, which is N in the computation of $RMSE$ and df (the degrees of freedom of the model, which is N minus the number of fit parameters) in the computation of $\hat{\sigma}$.

3. MAE , the mean absolute error. This is

$$MAE = \frac{1}{N} \sum |y_i - \hat{y}_i| \quad (9.20)$$

sigma

RMSE

MAE

4.74

4.68

3.58

If the goal of an analysis is prediction, one of these statistics should be reported. For the model fit to the observational green-down data in Extended Figure 3d, these three statistics are given in Table ?? above (to two decimal places simply to show numerical difference between $\hat{\sigma}$ and $RMSE$). All of these are measures of an “average” prediction error in the units of the response. The average error is either 4.7 or 3.6 days, depending on which statistic we report. Why the difference? Both $\hat{\sigma}$ and $RMSE$ are functions of squared error and so big differences between predicted and actual value are emphasized. If an error of 6 days is more than twice as bad than an error of 3 days, report $RMSE$. Why $RMSE$ and not $\hat{\sigma}$? Simply because researchers using prediction models are more familiar with $RMSE$. If an error of 6 days is *not* more than twice as bad than an error of 3 days, report MAE .

9.1.4.6 Prediction modeling is more sophisticated than presented here

For data where the response is a non-linear function of the predictor, or for data with many predictor variables, researchers will often build a model using a **model selection** method. Stepwise regression is a classical model selection method that is commonly taught in intro biostatistics and commonly used by researchers. Stepwise regression as a method of model selection has many well-known problems and should be avoided.

Some *excellent* books that are useful for building models and model selection are

1. The Elements of Statistical Learning
2. Regression and Other Stories
3. Regression Modeling Strategies

9.1.5 Using a regression model for creating a new response variable – comparing slopes of longitudinal data

In the study in this example, the researchers compared the growth rate of tumors in mice fed normal chow versus mice with a methionine restricted diet. Growth rate wasn’t actually compared. Instead, the researchers used a t -test to compare the size of the tumor measured at six different days. A problem with multiple t -tests for this dataset is that the errors (residuals from the model) on one day are correlated with the errors from another day because of the repeated measures on each mouse. This correlation will inflate Type I error rate.

Instead of six t -tests, a better strategy for these data is to use a regression to calculate a tumor growth rate for each mouse. There are sixteen mice so this is sixteen fit models. Here I use a “for loop” to fit the model to the data from

a single mouse and use the slope (b_1) as the estimate of the tumor growth rate for that mouse.

Use a for-loop to estimate growth rate for each mouse. In each pass through the loop

1. the subset of fig1f (the data in long format) belonging to mouse i is created
2. the linear model `volume ~ day` is fit to the subset
3. the coefficient of `day` (the slope, b_1) is inserted in mouse i 's row in the column “growth” in the data.table “fig1f_wide”.

At the end of the loop, we have the data.table `fig1f_wide` which has one row for each mouse, a column for the treatment factor (diet) and a column called “growth” containing each mouse’s growth rate. There are also columns of tumor volume for each mouse on each day but these are ignored.

```
N <- nrow(fig1f_wide)
id_list <- fig1f_wide[, id]
for(i in 1:N){
  mouse_i_data <- fig1f[id == id_list[i]] # subset
  fit <- lm(volume ~ day, data = mouse_i_data)
  fig1f_wide[id == id_list[i], growth := coef(fit)[2]]
}
# View(fig1f_wide)
# qplot(x = treatment, y = growth, data = fig1f_wide)
# qplot(x = day, y = volume, color = treatment, data = fig1f) + geom_smooth(aes(group = id), method = "loess")
```

Step 3. fit the model

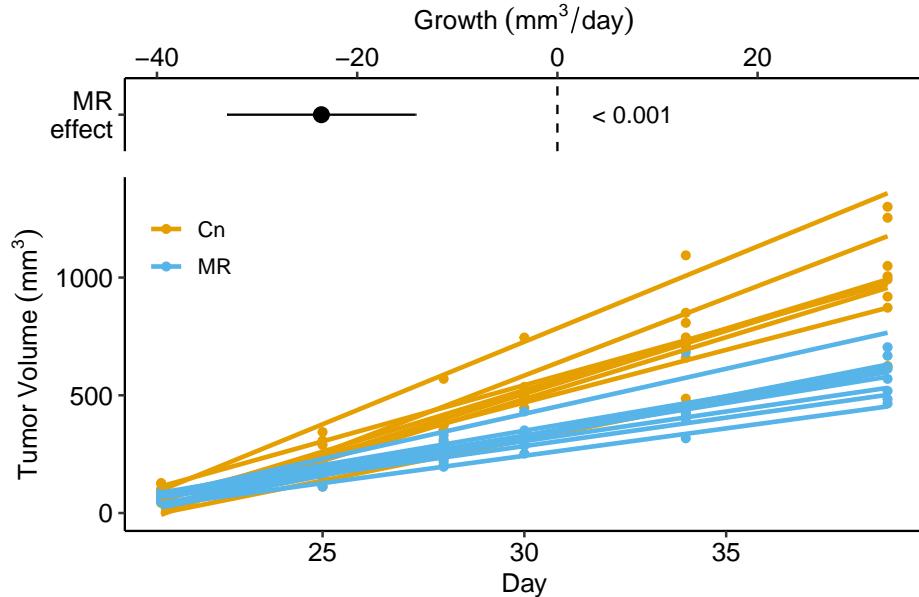
```
m1 <- lm(growth ~ treatment, data = fig1f_wide)
```

Step 5. inference from the model

```
m1_coef <- summary(m1) %>%
  coef
m1_ci <- confint(m1)
(m1_coef_table <- cbind(m1_coef, m1_ci))

##             Estimate Std. Error   t value   Pr(>|t|)    2.5 %
## (Intercept) 52.63406   3.102096 16.967258 9.865155e-11 45.98072
## treatmentMR -23.57616   4.387026 -5.374065 9.809457e-05 -32.98540
##                      97.5 %
## (Intercept) 59.28739
## treatmentMR -14.16693
```

Step 6. plot the model



9.1.6 Using a regression model for calibration

9.2 Working in R

9.2.1 Fitting the linear model

A linear model is fit with the `lm` function, which is very flexible and will be a workhorse in much of this text.

```
m1 <- lm(transition_date ~ temperature, data = fig2c)
```

`m1` is an `lm` model object that contains many different kinds information, such as the model coefficients.

```
coef(m1)
```

```
## (Intercept) temperature
## 289.458750    4.982745
```

We'll return to others, but first, let's explore some of the flexibility of the `lm` function. Two arguments were sent to the function

1. the model formula `transition_date ~ temperature` with the form `Y ~ X`, where Y and X are names of columns in the data.
- The model formula itself be assinged to a variable, which is useful when building functions. An example

```
coef_table <- function(x, y, data){
  m1_form <- formula(paste(y, "~", x))
  m1 <- lm(m1_form, data = data)
  return(coef(summary(m1)))
}

coef_table(x = "temperature",
            y = "transition_date",
            data = fig2c)

##           Estimate Std. Error   t value   Pr(>|t|)
## (Intercept) 289.458750  3.0593949 94.613071 1.738650e-13
## temperature    4.982745  0.5541962  8.990941 1.866888e-05
```

- Both Y and X can also be column names embedded within a function, for example

```
m2 <- lm(log(transition_date) ~ temperature, data = fig2c)
coef(m2)

## (Intercept) temperature
##      5.6690276     0.0160509
```

or

```
m3 <- lm(scale(transition_date) ~ scale(temperature), data = fig2c)
coef(m3)

## (Intercept) scale(temperature)
##      6.484929e-16      9.539117e-01
```

2. The data frame (remember that a `data.table` is a data frame) containing the columns with the variable names in the model formula. A `data` argument is not necessary but it is usually the better way (an exception is when a researcher wants to create a matrix of Y variables or to construct their own model matrix)

type `?lm` into the console to see other arguments of the `lm` function.

```
x <- fig2[, temperature]
y <- fig2[, transition_date]
m4 <- lm(y ~ x)
coef(m4)
```

```
## (Intercept)          x
## 204.8866185   0.4324755
```

9.2.2 Getting to know the linear model: the `summary` function

The `lm` function returns an `lm` object, which we've assigned to the name `m1`. `m1` contains lots of information about our fit of the linear model to the data. Most of the information that we want for most purposes can be retrieved with the `summary` function, which is a general-purpose R command that works with many R objects.

```
summary(m1)

##
## Call:
## lm(formula = transition_date ~ temperature, data = fig2c)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -7.5062 -3.8536  0.6645  2.7537  9.0074
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 289.4588     3.0594  94.613 1.74e-13 ***
## temperature  4.9827     0.5542   8.991 1.87e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.443 on 8 degrees of freedom
## Multiple R-squared:  0.9099, Adjusted R-squared:  0.8987
## F-statistic: 80.84 on 1 and 8 DF,  p-value: 1.867e-05
```

What is here:

Call. This is the model that was fit

Residuals. This is a summary of the distribution of the residuals. From this one can get a sense of the distribution (for inference, the model assumes a normal

distribution with mean zero). More useful ways to examine this distribution will be introduced later in this chapter.

Coefficients table. This contains the linear model coefficients and their standard error and associated t and p values.

1. The column of values under “Estimate” are the coefficients of the fitted model (equation (9.1)). Here, 289.4587503 is the intercept (b_0) and 4.9827453 is the effect of *temperature* (b_1).
2. The column of values under “Std. Error” are the standard errors of the coefficients.
3. The column of values under “t value” are the *t-statistics* for each coefficient. A *t*-value is a **signal to noise ratio**. The coefficient b_1 is the “signal” and the SE is the noise. Get used to thinking about this ratio. A *t*-value greater than about 3 indicates a “pretty good” signal relative to noise, while one much below than 2 is not.
4. The column of values under “Pr(>|t|)” is the *p*-value, which is the *t*-test of the estimate. What is the *p*-value a test of? The *p*-value tests the hypothesis “how probable are the data, or more extreme than than the data, if the true parameter is zero?”. Formally $p = \text{freq}(t' \geq t | H_0)$, where t' is the hypothetical *t*-value, t is the observed *t*-value, and H_0 is the null hypothesis. We will return to *p*-values in Chapter xxx.

Signif. codes. Significance codes are extremely common in the wet bench experimental biology literature but do not have much to recommend. I’ll return to these in the *p*-values chapter.

Beneath the Signif. codes are some model statistics which are useful

Residual standard error This is $\sqrt{\sum e_i^2 / (n - 2)}$, where e_i are the residuals in the fitted model. “degrees of freedom” is the number of e_i that are “allowed to vary” after fitting the parameters, so is the total sample size (n) minus the number of parameters in the model. The fit model has two fit parameters (b_0 and b_1) so the df is $n - 2$. Note that this is the denominator in the residual standard error equation.

Multiple R-squared. This is an important but imperfect summary measure of the whole model that effectively measures how much of the total variance in the response variable “is explained by” the model. Its value lies between zero and 1. **It’s a good measure to report in a manuscript**, especially for observational data.

F-statistic and p-value. These are statistics for the whole model (not the individual coefficients) and I just don’t find these very useful.

9.2.3 Inference – the coefficient table and Confidence intervals

To get the coefficient table without the other statistics from `summary`, use `summary(m1) %>% coef()`.

```
m1_coeff <- summary(m1) %>%
  coef()
m1_confint <- confint(m1)
m1_coeff <- cbind(m1_coeff, m1_confint)
m1_coeff
```

	Estimate	Std. Error	t value	Pr(> t)	2.5 %
## (Intercept)	289.458750	3.0593949	94.613071	1.738650e-13	282.403773
## temperature	4.982745	0.5541962	8.990941	1.866888e-05	3.704767
##	97.5 %				
## (Intercept)	296.513728				
## temperature	6.260724				

Note that the *p*-value for the coefficient for *temperature* is very small and we could conclude that the data are not compatible with a model of no temperature effect on day of green-down. But did we need a formal hypothesis test for this? We haven't learned much if we have only learned that the slope is "not likely to be exactly zero" (Temperature effects everything in biology). A far more important question is not *if* there is a relationship between *temperature* and *day* of green-down, but what is the sign and magnitude of the effect and what is the uncertainty in our estimate of this effect. For this, we want the coefficient and its SE or confidence interval, both of which are in this table. Remember, our working definition of a confidence interval:

A confidence interval contains the range of parameter values that are compatible with the data, in the sense that a *t*-test would not reject the null hypothesis of a difference between the estimate and any value within the interval

A more textbook way of defining a CI is: A 95% CI of a parameter has a 95% probability of including the true value of the parameter. It does not mean that there is a 95% probability that the true value lies in the interval. This is a subtle but important difference. Here is a way of thinking about the proper meaning of the textbook definition: we don't know the true value of β_1 but we can 1) repeat the experiment or sampling, 2) re-estimate β_1 , and 3) re-compute a 95% CI. If we do 1-3 many times, 95% of the CIs will include β_1 within the interval.

Confidence intervals are often interpreted like *p*-values. That is, the researcher looks to see if the CI overlaps with zero and if it does, concludes there is "no effect". First, this conclusion is not correct – **the inability to find sufficient**

evidence for an effect does not mean there is no effect, it simply means there is insufficient evidence to conclude there is an effect!

Second, what we want to use the CI for is to guide us about how big or small the effect might reasonably be, given the data. Again, A CI is a measure of parameter values that are “compatible” with the data. If our biological interpretations at the small-end and at the big-end of the interval’s range radically differ, then we don’t have enough *precision* in our analysis to reach an unambiguous conclusion.

9.2.4 How good is our model? – Model checking

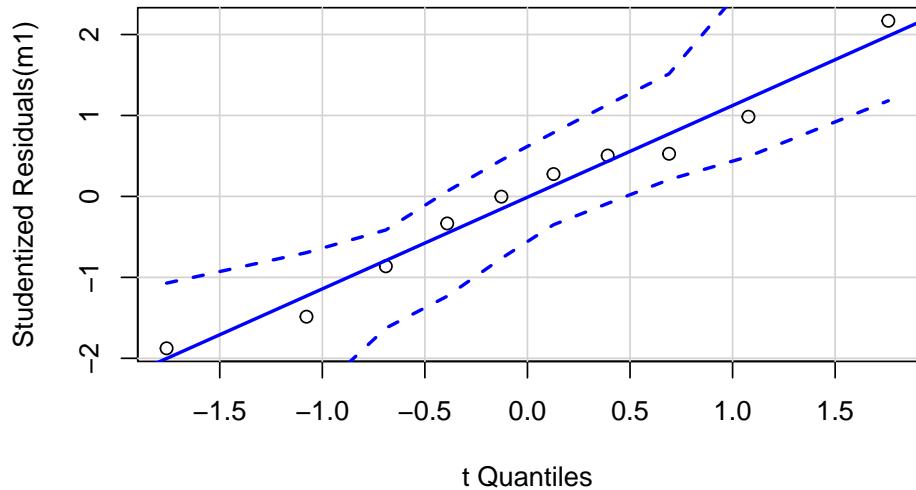
There are two, quite different senses of what is meant by a good model.

1. How good is the model at predicting? Or, how much of the variance in the response (the stuff to be “explained”) is accounted for by the model? This was described above.
2. How well do the data look like a sample from the modeled distribution? If not well, we should consider alternative models. This is **model checking**.

For inference, a good model generates data that look like the real data. If this is true, our fit model will have well-behaved residuals. There are several aspects of “well-behaved” and each is checked with a diagnostic plot. This **model checking** is covered in more detail in chapter xxx.

Inference from model (9.4) assumes the data were sampled from a normal distribution. To check this, use a quantile-quantile or **Q-Q plot**. The `qqPlot` function from the `car` package generates a more useful plot than that from Base R.

```
set.seed(1)
qqPlot(m1, id=FALSE)
```

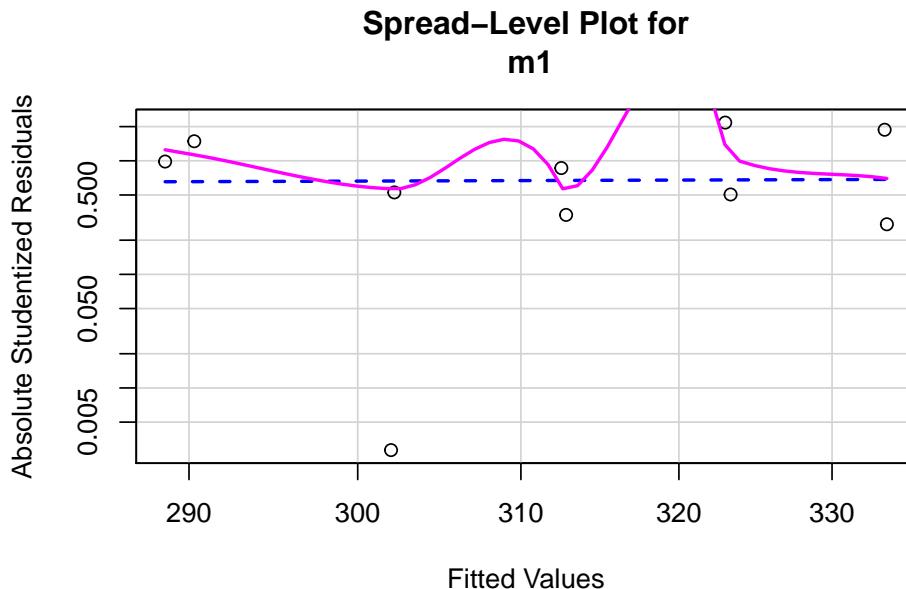


Approximately normal residuals will track the solid line and stay largely within the boundaries marked by the dashed lines. The residuals from $m1$ fit to the green-down data track the solid line and remain within the dashed lines, indicating adequate model fit.

Note that a formal test of normality is often recommended. Formal tests do not add value above a diagnostic check. Robustness of inference (for example, a p -value) is a function of the type and degree of “non-normalness”, not of a p -value. For a small sample, there is not much power to test for normality, so samples from non-normal distributions pass the test ($p > 0.05$) and are deemed “normal”. For large samples, samples from distributions that deviate slightly from normal fail the test ($p < 0.05$) and are deemed “not normal”. Inference with many non-normal samples with large n are very robust (meaning inference is not likely to fool you with randomness).

Inference from model (9.4) assumes homogeneity of the response conditional on X . For a continuous X , this means the residuals should have approximately equal variance at low, mid, and high values of X (and everywhere in between). One can visually inspect the spread of points in the Y direction across the groups for categorical X or along the X -axis for continuous X . A useful method for checking how residual variance changes (usually increases) with the conditional mean of Y is a **spread-location plot**. The `spreadLevelPlot(m1)` function from the car package is useful.

```
spreadLevelPlot(m1)
```



```
##  
## Suggested power transformation: 0.6721303
```

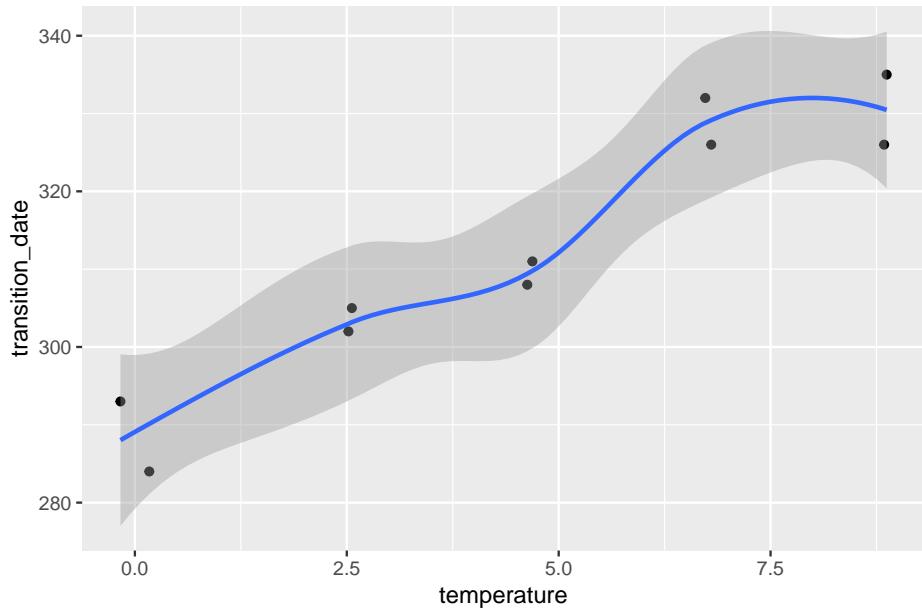
The dashed blue line shows linear trends while the magenta line shows non-linear trends. For the green-down data, the dashed line is very flat while the magenta-line shows what looks like random fluctuations. Taken together, the two lines indicate adequate model fit.

9.2.5 Plotting models with continuous X

9.2.5.1 Quick plot

`qplot` from the `ggplot` package is useful for initial examinations of the data

```
qplot(x = temperature, y = transition_date, data = fig2c) + geom_smooth()
```



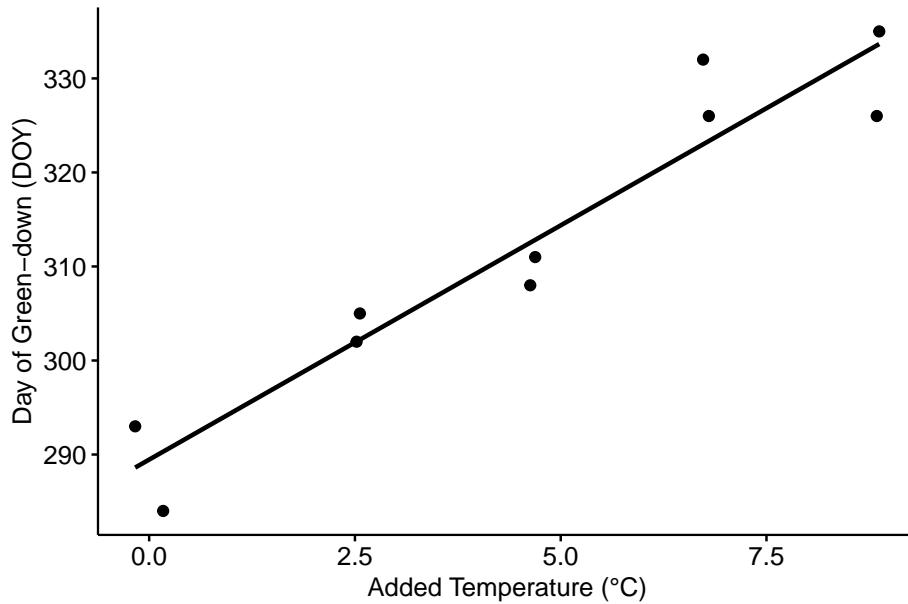
Some variations of this quick plot to explore include

1. use `geom_smooth(method = "lm")`

9.2.5.2 ggpublisher plots

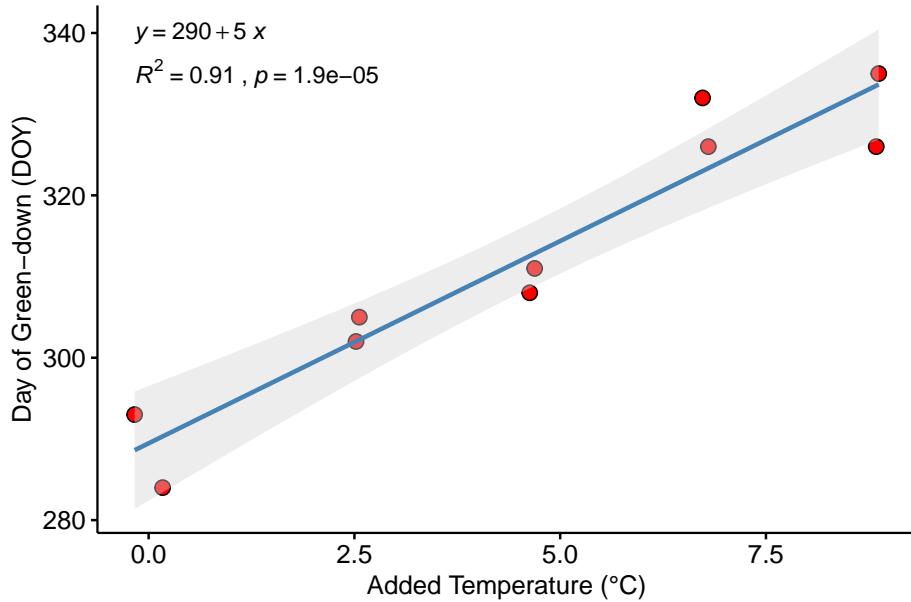
ggpubr is a package with functions that automates the construction of publication-ready ggplots. `ggscatter` can generate a publishable plot with little effort.

```
ggscatter(data = fig2c,
          x = "temperature",
          y = "transition_date",
          add = "reg.line",
          xlab = "Added Temperature (°C)",
          ylab = "Day of Green-down (DOY)")
```



It take only a little more effort to add useful modifications.

```
ggsscatter(data = fig2c,
  x = "temperature",
  y = "transition_date",
  color = "black",
  fill = "red",
  shape = 21,
  size = 3,
  add = "reg.line",
  add.params = list(color = "steelblue",
    fill = "lightgray"),
  conf.int = TRUE,
  xlab = "Added Temperature (°C)",
  ylab = "Day of Green-down (DOY)") +
  stat_regleine_equation(size = 4,
    label.y = 340) +
  stat_cor(aes(label = paste(..rr.label.., ..p.label.., sep = "~~`~~")),
    size = 4,
    label.y = 335) +
  NULL
```



Notes

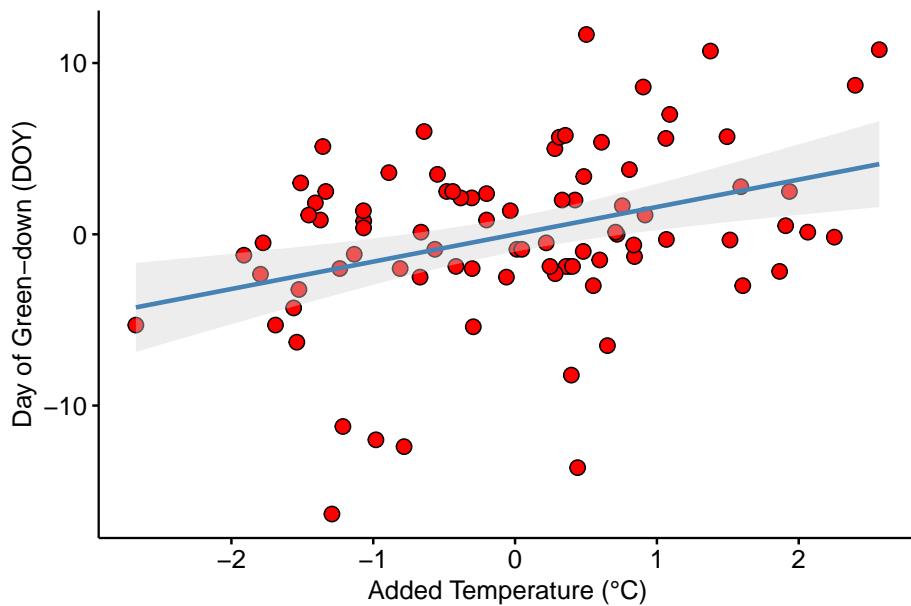
1. The interval shown is the 95% confidence interval of the expected values, which is what we want to communicate with the experimental green-down data. Were we to want to use this as a prediction model, we would want the 95% prediction interval. I'm not sure that ggpibr can plot a prediction interval. To see how I plotted the prediction interval in Figure ?? above, see the Hidden Code section below.
2. The arguments of the ggscatter function are typed in explicitly (`x = "temperarture"` and not just `"temperature"`). Each argument starts on a new line to increase readability.
3. The `+` after the ggscatter function adds additional layers to the plot. Each additional component is started on a new line to increase readability.
4. The first line of the stat_cor function (everything within `aes()`) plots the R^2 instead of the correlation coefficient r . Copy and pasting this whole line just works.
5. Comment out the line of the ggplot script starting with `stat_cor` and re-run (comment out by inserting a `#` at the front of the line. A consistent way to do this is to triple-click the line to highlight the line and then type command-shift-c on Mac OS). The script runs without error because `NULL` has been added as the final plot component. Adding “`NUL`” is a useful trick.

```
ggscatter(data = efig_3d,
          x = "autumn_temp_anomaly",
          y = "green_down_anomaly",
          aes(x = autumn_temp_anomaly,
              y = green_down_anomaly,
              color = "red"),
          add = "reg.line",
          reg.line = list(label = "y = 290 + 5 x",
                         R2 = TRUE),
          cor.coef = TRUE,
          cor.method = "pearson",
          cor.size = 100,
          cor.color = "black",
          cor.x = 0.0,
          cor.y = 340)
```

```

color = "black",
fill = "red",
shape = 21,
size = 3,
add = "reg.line",
add.params = list(color = "steelblue",
                   fill = "lightgray"),
conf.int = TRUE,
xlab = "Added Temperature (°C)",
ylab = "Day of Green-down (DOY)")

```



9.2.5.3 ggplots

Use `ggplot` from the `ggplot2` package for full control of plots. See the Hidden Code below for how I generated the plots in Figure 9.1 above.

9.2.6 Creating a table of predicted values and 95% prediction intervals

`efig_3d` is the `data.table` created from importing the data in Extended Data Figure 3d above. The fit model is

```
efig3d_m1 <- lm(green_down_anomaly ~ autumn_temp_anomaly, data = efig_3d)
```

The `predict(efig3d_m1)` function is used to compute either fitted or predicted values, and either the confidence or prediction interval of these values.

Unless specified by `newdata`, the default x-values used to generate the y in `predict(efig3d_m1)` are the x-values in the data used to fit the model. The returned values are the expected value for each x_i in the data. The argument `newdata` passes a `data.frame` (remember a `data.table` is a `data.frame!`) with new x-values. Since these x-values were not in the data used to fit the model, the returned y_{hat} are predicted values.

The range of x-values in the data is

```
range(efig_3d[,autumn_temp_anomaly])
```

```
## [1] -2.673793 2.568045
```

Let's get predicted values for a value of $autumn_temp_anomaly = 2$.

```
predict(efig3d_m1, newdata = data.table(autumn_temp_anomaly = 2))
```

```
##          1
## 3.192646
```

And predicted values across the range of measured values

```
new_dt <- data.table(autumn_temp_anomaly = c(-2, -1, 1, 2))
predict(efig3d_m1, newdata = new_dt)
```

```
##          1          2          3          4
## -3.192646 -1.596323  1.596323  3.192646
```

Add 95% confidence intervals (this could be used to create the band for plotting)

```
predict(efig3d_m1,
        newdata = new_dt,
        interval = "confidence",
        se.fit = TRUE)$fit

##          fit      lwr      upr
## 1 -3.192646 -5.2485713 -1.1367215
## 2 -1.596323 -2.9492686 -0.2433778
## 3  1.596323  0.2433778  2.9492686
## 4  3.192646  1.1367215  5.2485713
```

Change to the 95% prediction intervals

```
predict(efig3d_m1,
        newdata = new_dt,
        interval = "prediction",
        se.fit = TRUE)$fit

##          fit      lwr      upr
## 1 -3.192646 -12.833739  6.448446
## 2 -1.596323 -11.112325  7.919679
## 3  1.596323 -7.919679 11.112325
## 4  3.192646 -6.448446 12.833739
```

Put this all together in a pretty table. I've used knitr's `kable` function but there are table packages in R that allow extensive control of the output.

```
efig3d_m1 <- lm(green_down_anomaly ~ autumn_temp_anomaly, data = efig_3d)
new_dt <- data.table(autumn_temp_anomaly = c(-2, -1, 1, 2))
prediction_table <- predict(efig3d_m1,
                            newdata = new_dt,
                            interval = "prediction",
                            se.fit = TRUE)$fit
prediction_table <- data.table(new_dt, prediction_table)
pretty_names <- c("Autumn Temp Anomaly", "Estimate", "2.5%", "97.5%")
setnames(prediction_table, old = colnames(prediction_table), new = pretty_names)
knitr::kable(prediction_table, digits = c(1, 1, 1, 1))
```

Autumn Temp Anomaly

Estimate

2.5%

97.5%

-2

-3.2

-12.8

6.4

-1

-1.6

-11.1

7.9

```

1
1.6
-7.9
11.1
2
3.2
-6.4
12.8

```

9.3 Hidden code

9.3.1 Import and plot of fig2c (ecosystem warming experimental) data

Import

```

data_from <- "Ecosystem warming extends vegetation activity but heightens vulnerability"
file_name <- "41586_2018_399_MOESM3_ESM.xlsx"
file_path <- here(data_folder, data_from, file_name)
fig2 <- read_excel(file_path) %>% # import
  clean_names() %>% # clean the column names
  data.table() # convert to data.table
# View(fig2)

fig2c <- fig2[panel == "2c",]
# View(fig2c)

```

Creating the response plot (the bottom component)

```

m1_b <- coef(m1)
m1_ci <- confint(m1)
m1_b0_text <- paste0("Intercept: ",
                      round(m1_b[1],1),
                      " (",
                      round(m1_ci[1,1],1),
                      ", ",
                      round(m1_ci[1,2],1),
                      ") d")
m1_b1_text <- paste0("Slope: ",
                      round(m1_b[2],2),

```

```

    " (",
    round(m1_ci[2,1],2),
    ", ",
    round(m1_ci[2,2],2),
    ") d per 1 °C")

# regression line
m1_x <- min(fig2c[, temperature])
m1_xend <- max(fig2c[, temperature])
m1_y <- m1_b[1] + m1_b[2]*m1_x
m1_yend <- m1_b[1] + m1_b[2]*m1_xend

fig2c_gg_response <- ggplot(data = fig2c,
  aes(x = temperature,
      y = transition_date)) +

# regression line first, to not block point
geom_segment(x = m1_x,
  y = m1_y,
  xend = m1_xend,
  yend = m1_yend) +
# create black edge to points
# geom_point(size = 4,
#            color = "black") +
geom_point(size = 3,
  color = pal_okabe_ito[1]) +
scale_x_continuous(breaks = c(0, 2.25, 4.5, 6.75, 9)) +
xlab("Plot temperature ( $\Delta T$ , °C)") +
ylab("Autumn green-down (DOY)") +
theme_pubr() +
NULL

# fig2c_gg_response

```

Creating the effects plot (the top component)

```

m1_coeff_dt <- data.table(term = row.names(m1_coeff),
                           data.table(m1_coeff))[2,] %>%
  clean_names()
m1_coeff_dt[, p.pretty := pvalString(pr_t)]

min_bound <- min(m1_coeff_dt[, x2_5_percent])
max_bound <- min(m1_coeff_dt[, x97_5_percent])

```

```

y_lo <- min(min_bound+min_bound*0.2,
             -max_bound)
y_hi <- max(max_bound + max_bound*0.2,
            -min_bound)
y_lims <- c(y_lo, y_hi)

fig2c_gg_effect <- ggplot(data=m1_coeff_dt,
                           aes(x = term,
                               y = estimate)) +
  # confidence level of effect
  geom_errorbar(aes(ymax=x2_5_percent,
                     ymax=x97_5_percent),
                 width=0,
                 color="black") +
  # estimate of effect
  geom_point(size = 3) +
  # zero effect
  geom_hline(yintercept=0, linetype = 2) +
  # p-value
  annotate(geom = "text",
           label = m1_coeff_dt$p.pretty,
           x = 1,
           y = 7.5) +
  # aesthetics
  scale_y_continuous(position="right") +
  scale_x_discrete(labels = "Temperature\nneffect") +
  ylab("Effects (day/°C)") +
  coord_flip(ylim = y_lims) +
  theme_pubr() +
  theme(axis.title.y = element_blank()) +
NULL

#fig2c_gg_effect

```

Combining the response and effects plots into single plot

```

fig3d_fig <- plot_grid(fig2c_gg_effect,
                        fig2c_gg_response,
                        nrow=2,
                        align = "v",
                        axis = "lr",

```

```
    rel_heights = c(0.4,1))
fig3d_fig
```

9.3.2 Import and plot efig_3d (Ecosysem warming observational) data

Import

```
data_from <- "Ecosystem warming extends vegetation activity but heightens vulnerability to cold t
file_name <- "41586_2018_399_MOESM6_ESM.xlsx"
file_path <- here(data_folder, data_from, file_name)
efig_3d <- read_excel(file_path,
                      range = "J2:K87",
                      col_names = FALSE) %>% # header causing issues
data.table() # convert to data.table
setnames(efig_3d, old = colnames(efig_3d), new = c("autumn_temp_anomaly", "green_down_anomaly"))

# View(efig_3d)
```

Plot

```
m1 <- lm(green_down_anomaly ~ autumn_temp_anomaly, data = efig_3d)

# get x for drawing slope
minx <- min(efig_3d[,autumn_temp_anomaly])
maxx <- max(efig_3d[,autumn_temp_anomaly])
new_x <- seq(minx, maxx, length.out = 20)
new_data <- data.table(autumn_temp_anomaly = new_x)
new_data[, yhat := predict(m1, newdata = new_data)]
new_data[, conf_lwr := predict(m1,
                               se.fit = TRUE,
                               interval = "confidence",
                               newdata = new_data)$fit[, "lwr"]]
new_data[, conf_upr := predict(m1,
                               se.fit = TRUE,
                               interval = "confidence",
                               newdata = new_data)$fit[, "upr"]]
new_data[, pred_lwr := predict(m1,
                               se.fit = TRUE,
                               interval = "prediction",
                               newdata = new_data)$fit[, "lwr"]]
new_data[, pred_upr := predict(m1,
                               se.fit = TRUE,
```

```

            interval = "prediction",
            newdata = new_data)$fit[, "upr"]]

gg <- ggscatter(data = efig_3d,
                 x = "autumn_temp_anomaly",
                 y = "green_down_anomaly",
                 color = "black",
                 shape = 21,
                 size = 3,
                 add = "reg.line",
                 add.params = list(color = "steelblue",
                                   fill = "lightgray"),
                 xlab = "Temperature Anomaly (°C)",
                 ylab = "Day of Green-down Anomaly (DOY)") +
                 geom_ribbon(data = new_data,
                             aes(ymin = pred_lwr,
                                 ymax = pred_upr,
                                 y = yhat,
                                 fill = "band"),
                             fill = "gray",
                             alpha = 0.3) +
                 geom_ribbon(data = new_data,
                             aes(ymin = conf_lwr,
                                 ymax = conf_upr,
                                 y = yhat,
                                 fill = "band"), alpha = 0.3) +
                 stat_cor(aes(label = paste(..rr.label.., ..p.label.., sep = "~~")),
                           size = 4,
                           label.y = 10) +
                 scale_fill_manual(values = pal_okabe_ito) +
                 theme(legend.position="none") +
                 NULL

gg

```

9.3.3 Import and plot of fig1f (methionine restriction) data

Import

```
data_from <- "Dietary methionine influences therapy in mouse cancer models and alters human metab
file_name <- "41586_2019_1437_MOESM2_ESM.xlsx"
file_path <- here(data_folder, data_from, file_name)

fig1f_wide <- read_excel(file_path,
                           sheet = "f",
                           range = "B7:R12",
                           col_names = FALSE) %>%
  data.table() # convert to data.table

setnames(fig1f_wide,
         old = colnames(fig1f_wide),
         new = c("day",
                paste0("Cn_", 1:8),
                paste0("MR_", 1:8)))

fig1f_wide <- transpose(fig1f_wide, make.names = 1, keep.names = "id")
fig1f_wide[, treatment := factor(substr(id, 1, 2))]

days <- c(21, 25, 28, 30, 34, 39)
fig1f <- melt(fig1f_wide,
               id.vars <- c("treatment", "id"),
               measure.vars <- as.character(days),
               variable.name = "day",
               value.name = "volume")
fig1f[, day := as.numeric(as.character(day))]
# View(fig1f)
# qplot(x = day, y = volume, color = treatment, data = fig1f) + geom_line(aes(group = id))
```

Creating the response plot (the bottom component)

```
fig1f_gg_response <- ggplot(data = fig1f,
                               aes(x = day, y = volume, color = treatment)) +
  geom_point() +
  geom_smooth(aes(group = id), method = "lm", se = FALSE) +
  xlab("Day") +
  ylab(expression(Tumor~Volume^(mm^3))) +
  scale_color_manual(values = pal_okabe_ito) +
  theme_pubr() +
  theme(
```

```

legend.position = c(.15, .98),
legend.justification = c("right", "top"),
legend.box.just = "right",
legend.margin = margin(6, 6, 6, 6),
legend.title = element_blank()
) +
NULL

# fig1f_gg_response

```

Creating the effects plot (the top component)

```

m1_coeff_dt <- data.table(term = row.names(m1_coef_table),
                           data.table(m1_coef_table))[2,] %>%
  clean_names()
m1_coeff_dt[, p.pretty := pvalString(pr_t)]

min_bound <- min(m1_coeff_dt[, x2_5_percent])
max_bound <- min(m1_coeff_dt[, x97_5_percent])

y_lo <- min(min_bound+min_bound*0.2,
             -max_bound)
y_hi <- max(max_bound + max_bound*0.2,
            -min_bound)
y_lims <- c(y_lo, y_hi)

fig1f_gg_effect <- ggplot(data=m1_coeff_dt,
                            aes(x = term,
                                y = estimate)) +
  # confidence level of effect
  geom_errorbar(aes(ymin=x2_5_percent,
                     ymax=x97_5_percent),
                width=0,
                color="black") +
  # estimate of effect
  geom_point(size = 3) +
  # zero effect
  geom_hline(yintercept=0, linetype = 2) +
  # p-value
  annotate(geom = "text",
           label = m1_coeff_dt$p.pretty,
           x = 1,

```

```

y = 7.5) +
  # aesthetics
  scale_y_continuous(position="right") +
  scale_x_discrete(labels = "MR\neffect") +
  ylab(expression(Growth~(mm^3/day))) +
  coord_flip(ylim = y_lims) +
  theme_pubr() +
  theme(axis.title.y = element_blank()) +
  NULL
#fig1f_gg_effect

```

Combining the response and effects plots into single plot

```

fig1f_gg <- plot_grid(fig1f_gg_effect,
                      fig1f_gg_response,
                      nrow=2,
                      align = "v",
                      axis = "lr",
                      rel_heights = c(0.4,1))
fig1f_gg

```

9.4 Try it

9.4.1 A prediction model from the literature

The data come from the top, middle plot of Figure 1e of

Parker, B.L., Calkin, A.C., Seldin, M.M., Keating, M.F., Tarling, E.J., Yang, P., Moody, S.C., Liu, Y., Zerenturk, E.J., Needham, E.J. and Miller, M.L., 2019. An integrative systems genetic analysis of mammalian lipid metabolism. Nature, 567(7747), pp.187-193.

Public source

Source data

The researchers built prediction models from a hybrid mouse diversity panel (HMDP) to predict liver lipid levels from measured plasma lipid levels in mice *and* in humans. The value of the predictor (X) variable for an individual is not a measured value of a single plasma lipid but the predicted value, or **score**, from the prediction model based on an entire panel of lipid measurements in that individual. The Y variable for the individual is the total measured level

across a family of lipids (ceramides, triacylglycerols, diacylglycerols) in the liver of the individual. The question is, how well does the prediction score predict the actual liver level?

1. Use the data for TG (triacylglycerols) (the top, middle plot of Figure 1e). The column D is the “score” from the prediction model using plasma lipid levels. This is the X variable (the column header is “fitted.total.Cer”). The column E is the total measured liver TG, so is the Y variable.
2. Fit the linear model $y \sim x$.
3. Create a publication-quality plot of liver TG (Y -axis) against score (X -axis) – what the researchers labeled “fitted.total.Cer”. Include R^2 on the plot.
4. Advanced – add a 95% prediction interval to the plot (the template code for this is in the Hidden Code section for efig3d)
5. Create a table of expected liver TG and 95% prediction interval of liver TG of the score values (13.5, 14, 14.5, 15, 15.5, 16).
6. Comment on the predictability of liver TG using the plasma scores.

9.5 Intuition pumps

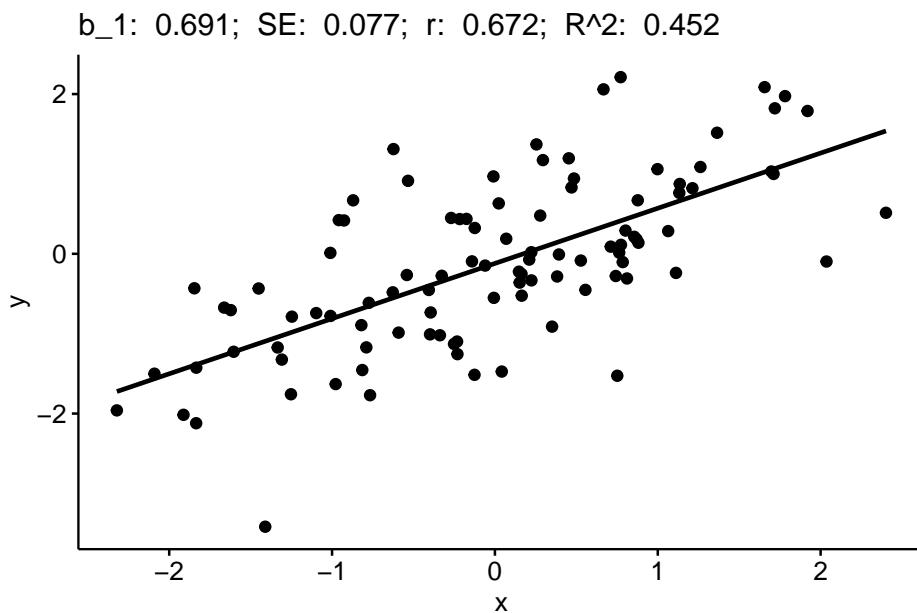
9.5.1 Correlation and R^2

In the code below

1. change the value of `n` to explore effect on the variability of the estimate. Look at this variability, and the magnitude of the SE, and the magnitude of the p-value.
2. change the value of `beta_1` to explore effect on what different slopes and correlations look like. Notice that if the variance is fixed (as in this simulation) the expected slope and expected correlation are equal. (Because I’ve fixed the variance in this simple simulation, this code will fail if $\text{abs}(\text{beta_1}) \geq 1$).

```
n <- 100 # choose between 3 and 10^5
beta_1 <- 0.6 # choose a value between -0.99 and 0.99
x <- rnorm(n)
y <- beta_1*x + sqrt(1-beta_1^2)*rnorm(n)
m1 <- lm(y ~ x)
slope <- paste("b_1: ", round(coef(m1)[2], 3))
se <- paste("SE: ", round(coef(summary(m1))[2,2], 3))
r <- paste("r: ", round(cor(x,y), 3))
```

```
r2 <- paste("R^2: ", round(summary(m1)$r.squared, 3))
ggscatter(data = data.table(x=x, y=y), x = "x", y = "y",
           add = "reg.line") +
  ggtitle(label = paste(slope, se, r, r2, sep="; "))
```



Chapter 10

Linear models with a single, categorical X

10.1 A linear model with a single, categorical X variable estimates the effects of the levels of X on the response.

To introduce a linear model with a single, categorical X variable, I'll use data from a set of experiments designed to measure the effect of the lipid 12,13-diHOME on brown adipose tissue (BAT) thermoregulation and the mechanism of this effect.

Lynes, M.D., Leiria, L.O., Lundh, M., Bartelt, A., Shamsi, F., Huang, T.L., Takahashi, H., Hirshman, M.F., Schlein, C., Lee, A. and Baer, L.A., 2017. The cold-induced lipokine 12, 13-diHOME promotes fatty acid transport into brown adipose tissue. *Nature medicine*, 23(5), pp.631-637.

Public source

Data source

Download the source data files and move to a new folder named “The cold-induced lipokine 12,13-diHOME promotes fatty acid transport into brown adipose tissue”.

Cold temperature and the neurotransmitter/hormone norepinephrine are known to stimulate increased thermogenesis in BAT cells. In this project, the researchers probed the question “what is the pathway that mediates the effect of cold-exposure on BAT thermogenesis?”. In the “discovery” component of this project, the researchers measured plasma levels of 88 lipids with known signalling properties in humans exposed to one hour of both normal (20 °C) and cold

temperature (14°C) temperature. Of the 88 lipids, 12,13-diHOME had the largest response to the cold treatment. The researchers followed this up with experiments on mice.

10.1.1 Example 1 – two treatment levels (“groups”)

Let’s start with the experiment in Figure 3d, which was designed to measure the effect of 12,13-diHOME on plasma triglyceride level. If 12,13-diHOME stimulates BAT activity, then levels in the 12,13-diHOME mice should be less than levels in the control mice.

response variable: *serum_tg*, a continuous variable.

treatment variable: *treatment*, with levels: “Vehicle”, “12,13-diHOME” (the control or “Vehicle” mice were injected with saline). Coded as a factor.

design: single, categorical X

10.1.1.1 Step 1 – import

The first step in any analysis is to open the data and, if necessary, wrangle into an analyzable format. The script to import these data is in the section Hidden code below.

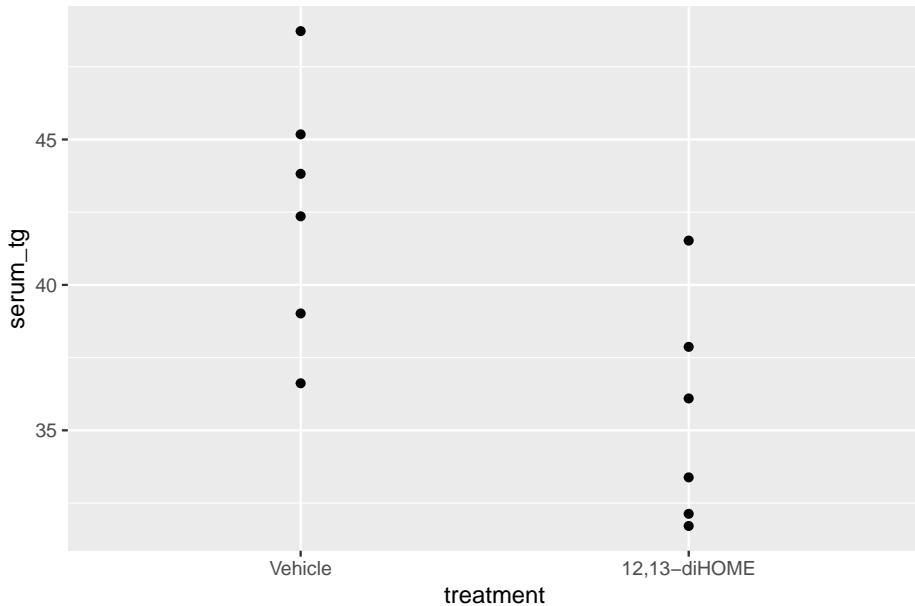
10.1.1.2 Step 2 – examine the data

The second step is to examine the data to

6. get a sense of sample size and balance
7. check for biologically implausible outliers that suggest measurement failure, or transcription error (from a notebook, not in a cell)
8. assess outliers for outlier strategy or robust analysis
9. assess reasonable distributions and models for analysis.

```
qplot(x = treatment, y = serum_tg, data = fig_3d)
```

10.1. A LINEAR MODEL WITH A SINGLE, CATEGORICAL X VARIABLE ESTIMATES THE EFFECTS OF THE TREATMENT



```
# ggdotplot(x = "treatment", y = "serum_tg", data = fig_3d)
```

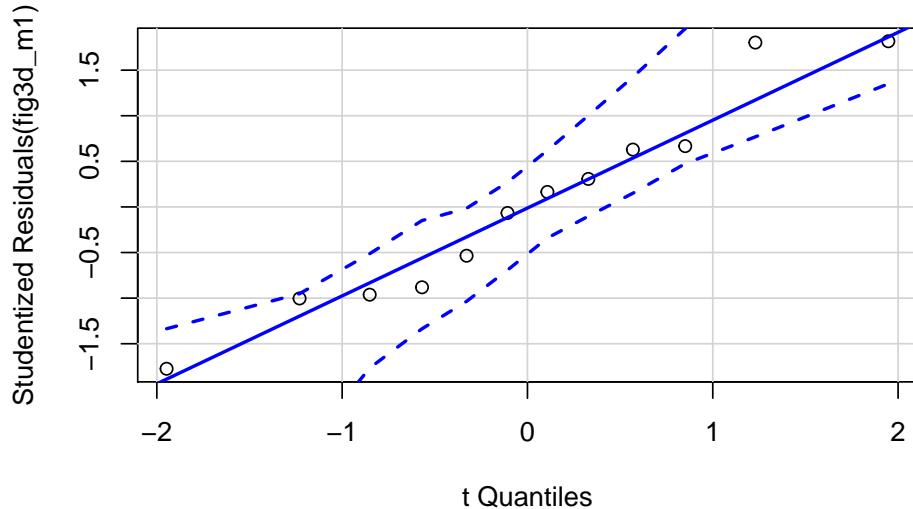
There are no obviously implausible data points. A normal distribution is a good, reasonable start. This can be checked more thoroughly after fitting the model.

10.1.1.3 Step 3 – fit the model

```
fig3d_m1 <- lm(serum_tg ~ treatment, data = fig_3d)
```

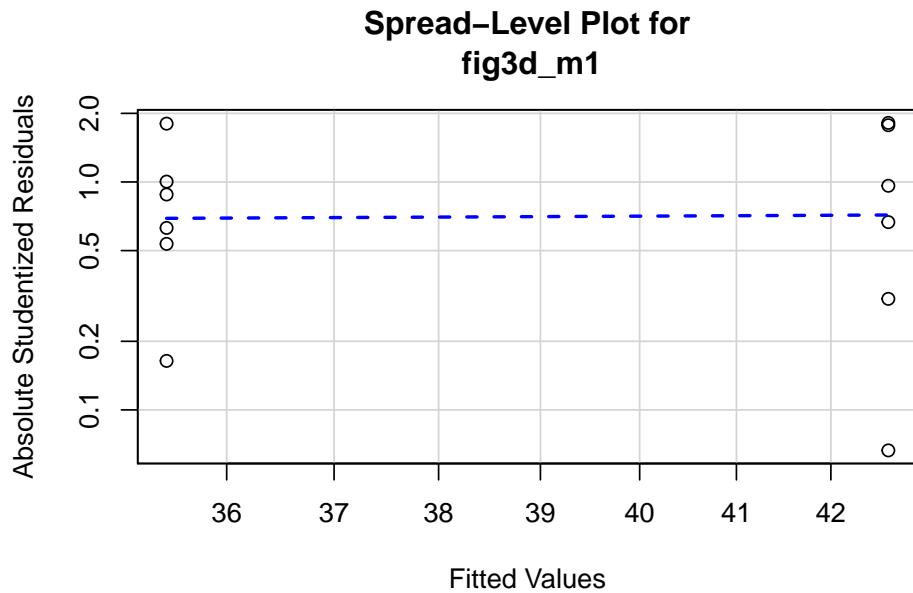
10.1.1.4 Step 4 – check the model

```
set.seed(1)
qqPlot(fig3d_m1, id=FALSE)
```



The Q-Q plot indicates the distribution of residuals is well within that expected for a normal sample and there is no cause for concern with inference.

```
spreadLevelPlot(fig3d_m1, id=FALSE)
```



```
##  
## Suggested power transformation: 0.8167264
```

The spread-location plot shows no conspicuous trend in how the spread changes with the conditional mean. There is no cause for concern with inference.

10.1. A LINEAR MODEL WITH A SINGLE, CATEGORICAL X VARIABLE ESTIMATES THE EFFECTS OF THE

10.1.1.5 Step 5 – inference

10.1.1.5.1 coefficient table

```
fig3d_m1_coef <- cbind(coef(summary(fig3d_m1)),
                         confint(fig3d_m1))
fig3d_m1_coef

##                               Estimate Std. Error   t value   Pr(>|t|)
## (Intercept)           42.620042   1.667226 25.563447 1.926081e-10
## treatment12,13-diHOME -7.167711   2.357813 -3.039982 1.246296e-02
##                               2.5 %    97.5 %
## (Intercept)           38.90523  46.334853
## treatment12,13-diHOME -12.42125 -1.914175

#knitr::kable(fig3d_m1_coef, digits = c(1,2,1,4,1,1))
```

10.1.1.5.2 Marginal means table

```
(fig3d_m1_emm <- emmeans(fig3d_m1, specs = "treatment"))

##  treatment     emmean    SE df lower.CL upper.CL
##  Vehicle       42.6  1.67 10     38.9     46.3
##  12,13-diHOME 35.5  1.67 10     31.7     39.2
##
##  Confidence level used: 0.95
```

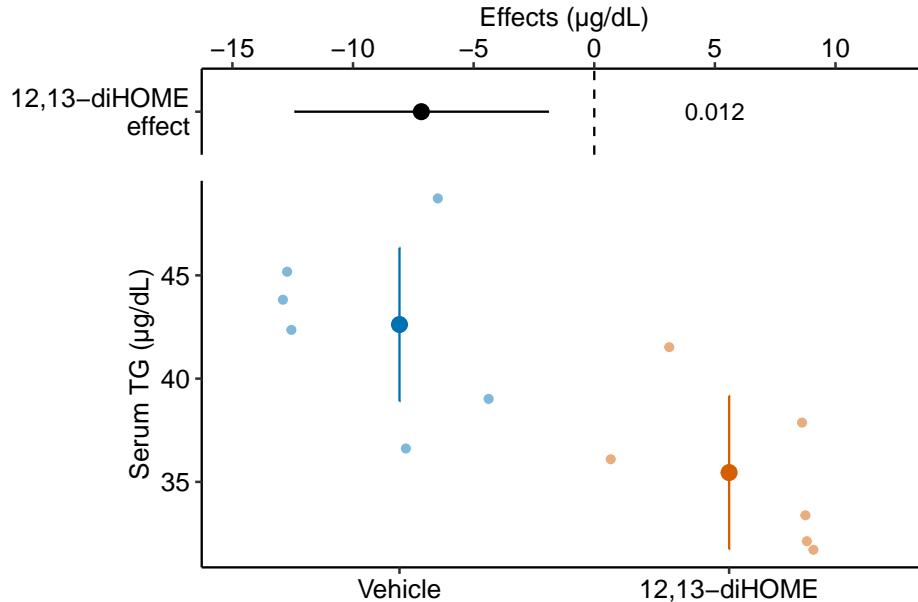
10.1.1.5.3 Contrasts table

```
(fig3d_m1_pairs <- contrast(fig3d_m1_emm,
                             method = "revpairwise") %>%
  summary(infer = TRUE))

##  contrast             estimate    SE df lower.CL upper.CL t.ratio p.value
##  12,13-diHOME - Vehicle -7.17  2.36 10    -12.4    -1.91 -3.040  0.0125
##
##  Confidence level used: 0.95
```

10.1.1.6 Step 6 – plot the model

The script for plotting the model in the section Hidden code below.



10.1.1.7 Step 7 – report the model results

Mean serum TG in mice with 12,13-diHOME ($35.5 \mu\text{g}/\text{dL}$, 95% CI: 31.7, 39.2) was $7.17 \mu\text{g}/\text{dL}$ less (95% CI: -12.4, -1.9, $p = 0.012$) than mean serum TG in control mice ($42.6 \mu\text{g}/\text{dL}$, 95% CI: 38.9, 46.3).

10.1.2 Understanding the analysis with two treatment levels

The variable *treatment* in the Figure 3d mouse experiment, is a single, categorical *X* variable. In a linear model, categorical variables are called **factors**. *treatment* can take two different values, “Vehicle” and “12,13-diHOME”. The different values in a factor are the **factor levels** (or just “levels”). “Levels” is a strange usage of this word; a less formal name for levels is “groups”. In a **Nominal** categorical factor, the levels have no units and are unordered, even if the variable is based on a numeric measurement. For example, I might design an experiment in which mice are randomly assigned to one of three treatments: one hour at 14°C , one hour at 18°C , or one hour at 26°C . If I model this treatment as a nominal categorical factor, then I simply have three levels. While I would certainly choose to arrange these levels in a meaningful way in a plot,

10.1. A LINEAR MODEL WITH A SINGLE, CATEGORICAL X VARIABLE ESTIMATES THE EFFECTS OF THE TREATMENT

for the analysis itself, these levels have no units and there is no order. **Ordinal** categorical factors have levels that are ordered but there is no information on relative distance. The treatment at 18 °C is not more similar to 14 °C than to 26 °C. Nominal categorical factors is the default in R and how all factors are analyzed in this text.

10.1.2.1 Linear models are regression models

The linear model fit to the serum TG data is

$$serum_tg = treatment + \varepsilon \quad (10.1)$$

$$\varepsilon \sim N(0, \sigma^2) \quad (10.2)$$

This specification is potentially confusing because the variable *treatment* is a factor containing the words “Vehicle” and “12,13-diHOME” and not numbers. A linear model with a single factor containing two levels can be specified using notation for a regression model.

$$Y = \beta_0 + \beta_1 X_1 + \varepsilon \quad (10.3)$$

$$\varepsilon \sim N(0, \sigma^2) \quad (10.4)$$

Model (10.4) is a regression model where X_1 is not the variable *treatment*, containing the words “Vehicle” or “12,13-diHOME” but a numeric variable that indicates group membership, containing the number 1 if the element belongs to the first non-reference level (if there are only two levels, then there is only a single, non-reference level) and the number 0 if the element doesn’t belong to the first non-reference level.

For the serum TG data, “Vehicle” is the reference, so we can write the linear model fit to the serum TG data using regression model notation.

$$serum_tg = \beta_0 + \beta_1 treatment_{12,13-diHOME} + \varepsilon \quad (10.5)$$

$$\varepsilon \sim N(0, \sigma^2) \quad (10.6)$$

Model (10.6) is a regression model where $treatment_{12,13-diHOME}$ is not the variable *treatment*, containing the words “Vehicle” or “12,13-diHOME” but a numeric variable that indicates membership in the level “12,13-diHOME”. This variable contains the number 1 if the element belongs to “12,13-diHOME” and the number 0 if the element doesn’t belong to “12,13-diHOME”. More generally, model (10.4) is a regression model where X_1 contains the number 1 if the element

270CHAPTER 10. LINEAR MODELS WITH A SINGLE, CATEGORICAL X

belongs to the first non-reference level (if there are only two levels, then there is only a single, non-reference level) and the number 0 if the element doesn't belong to the first non-reference level. If there were a third level within *treatment* (say, a 12,13-diHOME inhibitor), there would be a second *X* variable added to the model (X_2), which would contain the number 1 if the element belongs to the second non-reference level (12,13-diHOME inhibitor) and the number 0, otherwise.

The *X* variables in the regression model notation that indicate group membership are called **indicator variables**. There are several ways of coding indicator variables and the way described here is called **dummy** or treatment coding. This text will typically call dummy-coded indicator variables **dummy variables**. The `lm` function creates these dummy variables under the table, in something called the **model matrix**. You won't see these columns in your data but if you did, they would look like this

```
treatment
serum_tg
treatment12,13-diHOME
Vehicle
42.35908
0
Vehicle
43.82046
0
Vehicle
39.01879
0
Vehicle
48.72651
0
Vehicle
45.17745
0
Vehicle
36.61795
0
```

10.1. A LINEAR MODEL WITH A SINGLE, CATEGORICAL X VARIABLE ESTIMATES THE EFFECTS OF THE

12,13-diHOME

36.09603

1

12,13-diHOME

32.12944

1

12,13-diHOME

33.38205

1

12,13-diHOME

41.52401

1

12,13-diHOME

31.71190

1

12,13-diHOME

37.87056

1

R names dummy variables by combining the names of the factor and the name of the level within the factor. So the X variable that R creates in the model matrix for the fit linear model in model (10.4) is $treatment12,13 - diHOME$. You can see these names as terms in the coefficient table of the fit model.

Analysis fail. There are alternatives to dummy coding for creating indicator variables. Dummy coding is the default in R and it makes sense when thinking about experimental data with an obvious control level. I also like the interpretation of a “interaction effect” using Dummy coding. The classical coding for ANOVA is deviation effect coding, which creates coefficients that are deviations from the grand mean. In contrast to R, Deviation coding is the default in many statistical software packages including SAS, SPSS, and JMP. The method of coding can make a difference in an ANOVA table. Watch out for this – I’ve come across numerous published papers where the researchers used the default dummy coding but interpreted the ANOVA table as if they had used deviation coding. This is both getting ahead of ourselves and somewhat moot, because I don’t advocate publishing ANOVA tables.

10.1.2.2 The “Estimates” in the coefficient table are estimates of the parameters of the linear model fit to the data.

```
##                               Estimate Std. Error   t value   Pr(>|t|) 
## (Intercept)            42.620042  1.667226 25.563447 1.926081e-10
## treatment12,13-diHOME -7.167711  2.357813 -3.039982 1.246296e-02
##                               2.5 %    97.5 %
## (Intercept)            38.90523 46.334853
## treatment12,13-diHOME -12.42125 -1.914175
```

The linear model (10.6) fit to the serum TG data has three parameters, including two in the regression equation. The “estimates” in the coefficient table are the estimates of the regression parameters β_0 and β_1 . These estimates are the coefficients of the fit model

$$\text{serum_tg} = b_0 + b_1 \beta_1 \text{treatment}_{12,13-\text{diHOME}} + e \quad (10.7)$$

The coefficients b_0 and b_1 are the two values in the column “Estimate” of the table of model coefficients (or “coefficient table”). In addition to the estimates, the table includes the standard error, 95% confidence interval, and t and p -values of each estimate.

10.1.2.3 The parameters of a linear model using dummy coding have an important interpretation

It is important to understand the interpretation of the coefficients of the fit linear model @??eq:fit-serum-tg). The “coefficient” b_0 is the first value in the “Estimate” column of the coefficient table (in the row “(intercept)”). This is the conditional mean of the response for the reference level, which is “Vehicle”. Remember that a conditional mean is the mean of a group that all have the same value for one or more X variables. The coefficient b_1 is the second value in the “Estimate” column (in the row “treatment12,13-diHOME”). b_1 is the difference between the conditional means of the 12,13-diHOME level and the reference (Vehicle) level. The *direction* of this difference is important; it is $\bar{Y}_{12,13-\text{diHOME}} - \bar{Y}_{\text{Vehicle}}$, that is, the non-reference level minus the reference level. The estimate for treatment12,13-diHOME is the **effect** that we are interested in. Specifically, it is the effect of 12,13-diHOME on serum TG. When we inject 12,13-diHOME, we find the mean serum TG decreases by -7.2 µg/dL relative to the mean serum TG in the mice that were injected with saline. Importantly, the reference level is not a property of an experiment but is set by whomever is analyzing the data. Since the non-reference estimates are differences in means, it often makes sense to set the “control” treatment level as the reference level.

10.1. A LINEAR MODEL WITH A SINGLE, CATEGORICAL X VARIABLE ESTIMATES THE EFFECTS OF THE

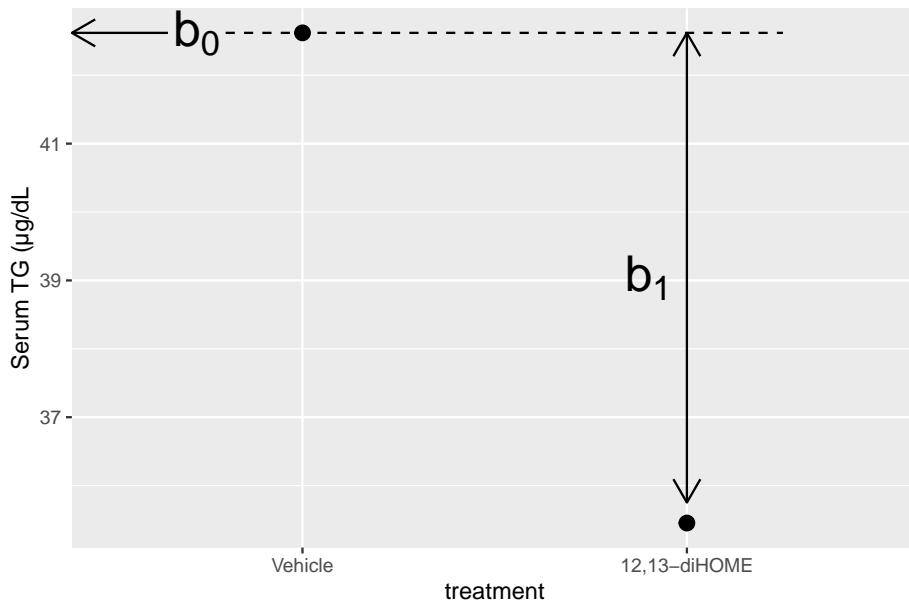


Figure 10.1: What the coefficients of a linear model with a single categorical X mean. The means of the two treatment levels for the serum TG data are shown with the filled circles. The intercept (b_0) is the mean of the reference treatment level. The coefficient b_1 is the difference between the treatment level's mean and the reference mean. As with a linear model with a continuous X, the coefficients are effects.

The intercept estimates the true, mean serum TG in a hypothetical population of mice that have been given saline but not 12,13-diHOME. The treatment 12,13-diHOME value estimates the true, difference in means between a hypothetical population of mice that have been given 12,13-diHOME and a population that has been given only saline.

tl;dr. What is a population? In the experimental biology examples in this text, we might consider the population as a very idealized, infinitely large set of mice, or fish, or fruit flies, or communities from which our sample is a reasonably representative subset. For the experiments in the 12,13-diHOME study, the population might be conceived of as the hypothetical, infinitely large set of 12-week-old, male, C57BL/6J mice, raised in the mouse facility at Joslin Diabetes Center. An even more abstract way to think about what the population could be is the infinitely large set of values that could be generated by the linear model.

Let's put this all together. b_0 is the conditional mean of the reference level ("Vehicle") and is an estimate of β_0 , the true, conditional mean of the population. b_1 is the difference in the conditional means of the first non-reference level ("12,13-diHOME") and the reference level ("Vehicle") and is an estimate of β_1 ,

the true difference in the conditional means of the population with and without the treatment 12,13-diHOME.

10.1.2.4 The table of marginal means is a table of modeled means and inferential statistics, not a table of raw means and inferential statistics

The table of marginal means for the model fit to the Figure 3d serum TG data is (shown to five decimal places for a later comparison)

treatment

emmmean

SE

df

lower.CL

upper.CL

Vehicle

42.62004

1.66723

10

38.90523

46.33485

12,13-diHOME

35.45233

1.66723

10

31.73752

39.16714

A **marginal mean** is the mean of a set of conditional means and is, consequently, a **modeled mean** (it comes from a model). The table of marginal means (“marginal means table”) outputs the specified marginal means and the standard error and 95% confidence interval of each mean. There is no test-statistic with a p -value because there is no significance test. The specified marginal means table of the Figure 3d data is not too exciting because it simply contains the conditional means – the values are not marginalized over any X . In several sections of this text, the marginal means table will contain values that average conditional means over one or more factors. The marginal means table

10.1. A LINEAR MODEL WITH A SINGLE, CATEGORICAL X VARIABLE ESTIMATES THE EFFECTS OF THE

also computes these means as the expected value (mean) at the average value of a continuous covariate, if any covariates are in the linear model. Because the marginal means table contains different sorts of means (conditional, marginal, adjusted), this text will generally refer to the means in this table as "modeled means".

If the design is **balanced**, meaning the sample size for each conditional mean is the same, then a marginal mean will simply equal the average of the individual values. But, this is not the case for unbalanced designs. For example, if we unbalance the Figure 3d data by throwing out the first row of and refit the model, the raw mean of *serum_tg* is 38.734 and the marginal mean, marginalized over *treatment*, is 39.062.

Like the modeled means, the standard errors in the marginal means table are modeled and not raw values. Recall that the standard error of a mean is $\frac{s}{\sqrt{n}}$, where s is the sample standard deviation. In the marginal means table, s is not the raw standard deviation of the group but the estimate of σ , the square root of the true variance. As with the raw standard error of the mean, the denominator is the sample size n for the group. Since the numerator of the modeled SE is the same for all groups, the modeled SE will be the same in all groups that have the same sample size, as seen in the marginal means table for the model fit to the Figure 3d data. This may seem odd. It is not. Remember that an assumption of the linear model is homogeneity of variances – that the e_i for each group are drawn from $N(0, \sigma^2)$ regardless of group. s^2 , which is computed as a variance of the model residuals, is an estimate this true variance (σ^2). It is also useful to think of the raw variances computed separately for each group (level of *treatment*) as estimates of σ^2 . The separately computed estimates are averaged to create a single estimate, which is equal to s^2 computed from the model residuals.

Unlike the modeled means, the modeled standard error and confidence interval will, effectively, never equal the raw values.

10.1.2.5 Report the modeled means and inferential statistics from the marginal means table, not the raw means and inferential statistics

This text advocates the best practice of reporting, including plotting, the modeled means and inferential statistics (SEM or confidence interval) and not the raw means and summary statistics, because only the modeled means and statistics are consistent with the modeled statistical analysis. Raw means and summary statistics can both mask the effects that we want to communicate and give misleading interpretation of the statistics, including the conditional distribution of the data.

The raw, group means and standard errors of each mean of the Figure 3d serum TG are

```
fig_3d[, .(mean = mean(serum_tg),
           SE = sd(serum_tg)/sqrt(.N)),
       by = treatment]

##      treatment     mean        SE
## 1:    Vehicle 42.62004 1.773251
## 2: 12,13-diHOME 35.45233 1.553984
```

The raw means are equal to the modeled means but the SE differs.

10.1.2.6 Estimates of the effects are in the contrasts table

```
fig3d_m1_pairs
```

```
##   contrast           estimate     SE df lower.CL upper.CL t.ratio p.value
## 12,13-diHOME - Vehicle -7.17 2.36 10   -12.4   -1.91 -3.040 0.0125
##
## Confidence level used: 0.95
```

This table is important for reporting treatment effects and CIs and for plotting the model. A contrast is a difference in means. With only two treatment levels, the table of contrasts doesn't give any more information than the coefficient table – the single contrast is the coefficient b_1 in the coefficient table. Nevertheless, I advocate computing this table to stay consistent and because the script to plot the model uses this table and not the coefficient table.

The values in the column “estimate” is the simple difference of groups given in the contrast column (what you would compute if you simply computed the difference for these groups). This is true for this model, but is not generally true.

The values in the “SE” column are standard errors of a difference (SED), specifically the difference in the estimate column. These SEs are from the fit model using the pooled estimate of σ .

The values in the “lower.CL” and “upper.CL” columns are the bounds of the 95% confidence interval of the estimate. This confidence level applies to the procedure and not the estimate. Think of this interval as containing potential values of the true parameter (the true difference in means between the two groups) that are reasonably compatible with the data. More formally, it is correct to interpret this CI as “95% of the CIs computed using this procedure include the true value given the model conditions.”

The columns “t.ratio” and “p.value” contains the t and p values of the significance (not hypothesis!) test of the estimate. The t -statistic is the ratio of the

10.1. A LINEAR MODEL WITH A SINGLE, CATEGORICAL X VARIABLE ESTIMATES THE EFFECTS OF THE

estimate to the SE of the estimate (use the console to confirm this given the values in the table). It is a signal (the estimate) to noise (SE of the estimate) ratio. The p -value is the probability of sampling the same distribution, fitting the linear model, and observing a t -value as or more extreme than the observed t . A very small p -value is consistent with the experiment “not sampling from distributions with the same mean” – meaning that adding a treatment affects the mean of the distribution. This is the logic used to infer a treatment effect. Unfortunately, it is also consistent with the experiment not approximating other conditions of the model, including non-random assignment, non-independence, non-normal conditional responses, and variance heterogeneity. It is up to the rigorous researcher to be sure that these other model conditions are approximated or “good enough” to use the p -value to infer a treatment effect on the mean.

10.1.2.7 t and p from the contrasts table – when there are only two levels in X – are the same as t and p from a t -test

Compare

```
m1 <- lm(serum_tg ~ treatment, data = fig_3d)
m1_pairs <- emmeans(m1, specs = "treatment") %>%
  contrast(method = "revpairwise") %>%
  summary(infer = TRUE)
m1_pairs

## contrast             estimate    SE df lower.CL upper.CL t.ratio p.value
## 12,13-diHOME - Vehicle     -7.17 2.36 10    -12.4    -1.91 -3.040  0.0125
##
## Confidence level used: 0.95

m2 <- t.test(serum_tg ~ treatment,
             data = fig_3d,
             var.equal = TRUE)
glance(m2) # glance is from the broom package

## # A tibble: 1 x 10
##   estimate estimate1 estimate2 statistic p.value parameter conf.low
##       <dbl>      <dbl>      <dbl>      <dbl>     <dbl>      <dbl>
## 1     7.17      42.6      35.5      3.04  0.0125      10     1.91
## # ... with 3 more variables: conf.high <dbl>, method <chr>,
## #   alternative <chr>
```

Notes

1. The “statistic” in the output contains the t -value of the t -test. It is equal in magnitude but opposite in sign to that in the contrast table from the linear model. This is because the `method = "revpairwise"` argument tells the `contrast` function to use the difference “non-reference mean minus reference mean”, which is the direction I prefer (I like to compare to the reference, which is usually a control)
2. The “ p .value” in the output contains the p -value of the t -test. It is equal to that in the contrast table from the linear model.
3. The `t.test` function doesn’t output the estimate of the difference in means, only the estimates of each mean.
4. The `t.test` function does give the 95% confidence intervals of the difference in means. Notice that these have the same magnitude but opposite sign of those in the contrast table, for the same reason given in note 1 above.

The t and p values for the t -test are the same as those for the linear model, because the t -test is a specific case of the linear model. Reasons to abandon classic t -tests and learn the linear modeling strategy include

1. A linear modeling strategy encourages researchers to think about the effect and uncertainty in the effect and not just a p -value.
2. The linear model is nearly infinitely flexible and expandible while the t -test has extremely limited flexibility.

10.1.3 Example 2 – three treatment levels (“groups”)

The data come from the experiment reported in Figure 2a of the 12,13-diHOME article described above. This experiment was designed to probe the hypothesis that 12,13-diHOME is a mediator of known stimulators of increased BAT activity (exposure to cold temperature and sympathetic nervous system activation). Mice were assigned to control (30 °C), one-hour exposure to 4 °C, or 30 minute norepinephrine (NE) treatment level (NE is the neurotransmitter of the sympathetic neurons targeting peripheral tissues).

response variable: *diHOME*, the serum concentration of 12,13-diHOME. a continuous variable.

treatment variable: *treatment*, with levels: “Control”, “1 hour cold”, ”30 min NE. Coded as a factor.

design: single, categorical X

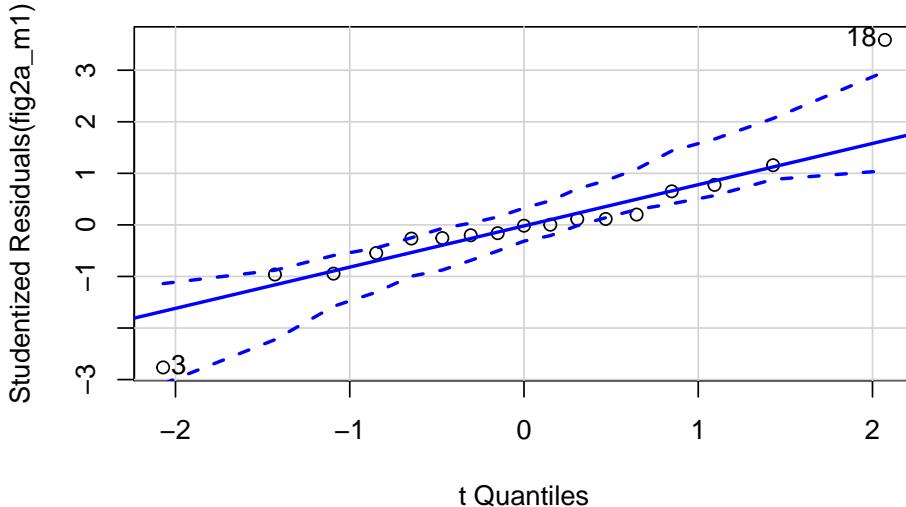
10.1.3.1 fit the model

10.1. A LINEAR MODEL WITH A SINGLE, CATEGORICAL X VARIABLE ESTIMATES THE EFFECTS OF THE

```
fig2a_m1 <- lm(diHOME ~ treatment, data = fig2a)
```

10.1.3.2 check the model

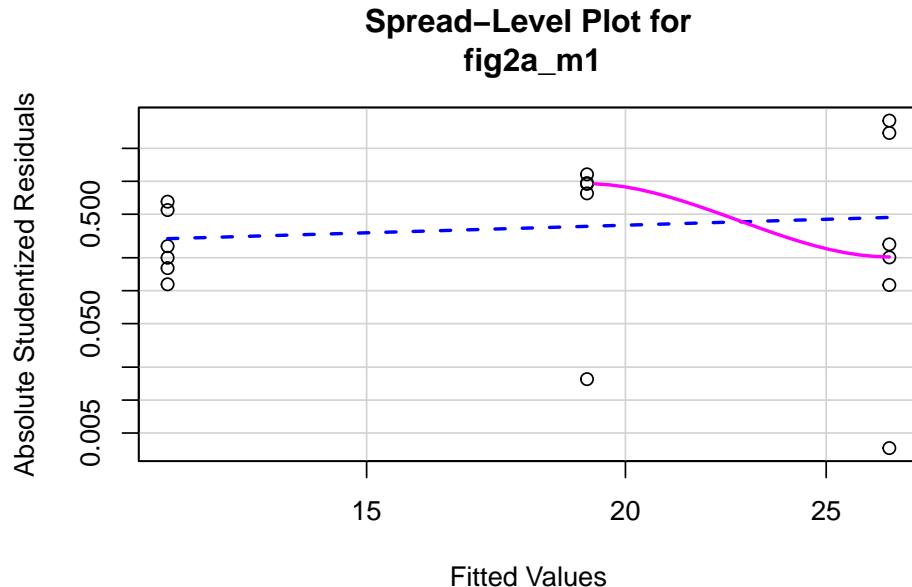
```
set.seed(1)
qqPlot(fig2a_m1)
```



```
## [1] 3 18
```

The Q-Q plot indicates potential issues at the extreme quantiles, what is called “heavy tails”. The two values are the extreme values in the “30 min NE” group. This could be the result of a small sample from a response with a larger variance.

```
spreadLevelPlot(fig2a_m1, id=FALSE)
```



```
##  
## Suggested power transformation: 0.4430799
```

The combination of the raw residuals and the spread-level plot suggests heterogeneity but low confidence in anything given the small sample size.

10.1.3.3 Inference from the model

10.1.3.3.1 Coefficient table

```
fig2a_m1_coef <- cbind(coef(summary(fig2a_m1)),  
                         confint(fig2a_m1))  
fig2a_m1_coef
```

	Estimate	Std. Error	t value	Pr(> t)	2.5 %
## (Intercept)	12.023075	3.081337	3.901902	0.001595771	5.414264
## treatment1 hour cold	7.140386	4.570362	1.562324	0.140527829	-2.662066
## treatment30 min NE	14.794354	4.357669	3.395015	0.004355868	5.448083
##	97.5 %				
## (Intercept)	18.63189				
## treatment1 hour cold	16.94284				
## treatment30 min NE	24.14063				

10.1. A LINEAR MODEL WITH A SINGLE, CATEGORICAL X VARIABLE ESTIMATES THE EFFECTS OF THE

10.1.3.3.2 Marginal means table

```
(fig2a_m1_emm <- emmeans(fig2a_m1, specs = "treatment"))
```

```
## treatment    emmean    SE df lower.CL upper.CL
## Control      12.0 3.08 14     5.41    18.6
## 1 hour cold  19.2 3.38 14    11.92    26.4
## 30 min NE    26.8 3.08 14    20.21    33.4
##
## Confidence level used: 0.95
```

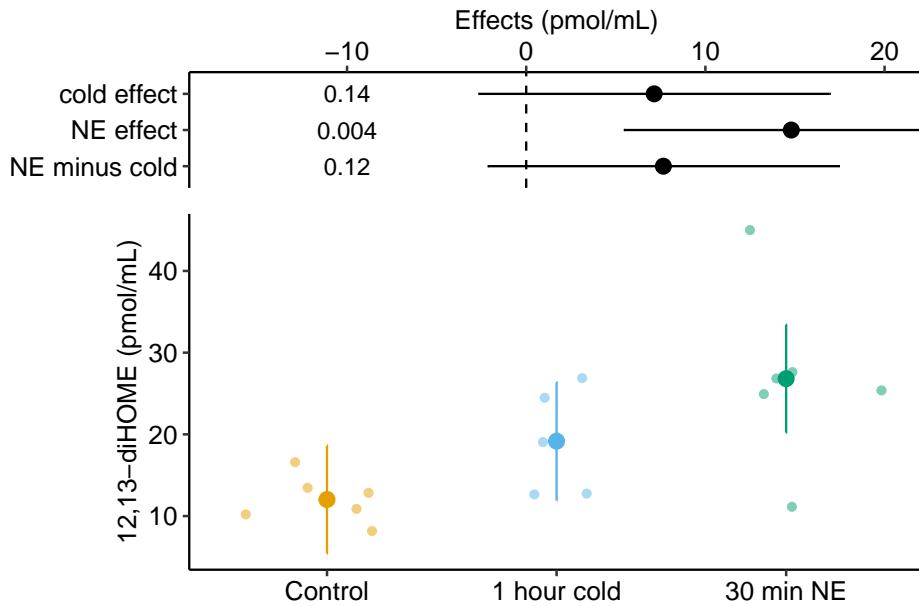
10.1.3.3.3 Contrasts table

```
(fig2a_m1_pairs <- contrast(fig2a_m1_emm,
                           method = "revpairwise",
                           adjust = "none") %>%
  summary(infer = TRUE))
```

```
## contrast          estimate    SE df lower.CL upper.CL t.ratio
## 1 hour cold - Control 7.14 4.57 14   -2.66    16.9 1.562
## 30 min NE - Control 14.79 4.36 14    5.45    24.1 3.395
## 30 min NE - 1 hour cold 7.65 4.57 14   -2.15    17.5 1.675
##
## p.value
## 0.1405
## 0.0044
## 0.1162
##
## Confidence level used: 0.95
```

10.1.3.4 plot the model

The script for plotting the model in the section Hidden code below.



10.1.3.5 Report the model results

Compared to control mice (12.0 pmol/mL, 95% CI: 5.4, 18.6), mean serum 12,13-diHOME in mice exposed to one-hour cold (19.2 pmol/mL, 95% CI: 11.9, 26.4) was 7.1 pmol/mL higher (95% CI: -2.7, 16.9, $p = 0.14$) while mean Serum 12,13-diHOME in mice exposed to 30 minutes NE (26.8 pmol/mL, 95% CI: 20.2, 33.3) was 14.8 pmol/mL higher (95% CI: 5.4, 24.1 $p = 0.004$).

10.1.4 Understanding the analysis with three (or more) treatment levels

10.1.4.1 The coefficient table

```

##                               Estimate Std. Error t value Pr(>|t|) 2.5 %
## (Intercept)           12.023075  3.081337 3.901902 0.001595771 5.414264
## treatment1 hour cold  7.140386  4.570362 1.562324 0.140527829 -2.662066
## treatment30 min NE   14.794354  4.357669 3.395015 0.004355868 5.448083
##                               97.5 %
## (Intercept)           18.63189
## treatment1 hour cold 16.94284
## treatment30 min NE   24.14063

```

To understand row names in the first column, it's useful to recall the order of the factor levels of *treatment*, which is

10.1. A LINEAR MODEL WITH A SINGLE, CATEGORICAL X VARIABLE ESTIMATES THE EFFECTS OF THE TREATMENT

```
levels(fig2a$treatment)

## [1] "Control"      "1 hour cold"   "30 min NE"
```

“Control” is the reference level, so the Estimate of the intercept is the mean $\bar{y}_{13\text{diHome}}$ for the “Control” mice. The 2nd row of the coefficient table contains the estimate and statistics for the “1 hour cold” level. The estimate (in the column “Estimate”) is the difference in means $\bar{y}_{1\text{hour_cold}} - \bar{y}_{\text{Control}}$. The 3rd row contains the estimate and statistics for the “30 min NE” level. The estimate is the difference in means $\bar{y}_{30\text{min_NE}} - \bar{y}_{\text{Control}}$.

Let’s put this in the context of the linear model fit to the data.

$$diHOME_i = b_0 + b_1 treatment_{1\text{hour_cold},i} + b_2 treatment_{30\text{min_NE},i} + e_i \quad (10.8)$$

The value in the column “Estimate” for the “(Intercept)” row is b_0 , the estimate of β_0 , the “population” mean of “control” mice. The value in the column “Estimate” for the “treatment1 hour cold” row is b_1 , the estimate of β_1 , the effect of the cold treatment on the response (relative to the control). The value in the column “Estimate” for the “treatment30 min NE” row is b_2 , the estimate of β_2 , the effect of the NE treatment on the response (relative to the control).

$treatment_{1\text{hour_cold},i}$ is a dummy-coded indicator variable, containing the number 1, if i is in the “1 hour cold” group, or the number 0, otherwise. $treatment_{30\text{min_NE},i}$ is a dummy-coded indicator variable, containing the number 1, if i is in the “30 min NE” group, or the number 0, otherwise. Importantly, the function `lm` creates these indicator variables under the hood. You don’t create these but understanding how these are made gives you phenomenal cosmic power (because there are models where you have to construct these manually).

This generalizes to any number of levels of the factor variable. If there are k levels of the factor, there are $k - 1$ indicator variables, each with its own coefficient (b_1 through b_{k-1}) that estimates the effect of that treatment level relative to the control (if using dummy coding).

As in the example with only two treatment levels above, both b_1 and b_2 are “slopes”. Don’t visualize this as a single line from the control mean through both non-control means but as two lines, each with their own slopes. The numerator of each slope is the difference between that group’s mean and the control mean. The denominator of each slope is 1 (because each has the value 1 when the row is assigned to that group).

The model formula (`dihome ~ treatment`) used in the `lm` function is the same, regardless of the number of levels in the factor *treatment*. This model formula

is the verbal form of the fit linear model and is useful for communication, but disconnects the formula from the actual model. For example, it might lead to the confusion on how the model `dihome ~ treatment` can output an intercept and *two* coefficients when there is only *one* X variable. The answer is, this question confuses the verbal formula with the quantitative formula. The model is fit using the quantitative formula which has $3 - 1 = 2$ indicator variables resulting in two (non-intercept) coefficients.

10.1.4.2 The estimated marginal means table

```
fig2a_m1_emm
```

```
##   treatment    emmean    SE df lower.CL upper.CL
## Control      12.0 3.08 14     5.41    18.6
## 1 hour cold  19.2 3.38 14    11.92    26.4
## 30 min NE    26.8 3.08 14    20.21    33.4
##
## Confidence level used: 0.95
```

This table is important for reporting means and CIs and for plotting the model. As in example 1, the values in the column “`emmean`” are the simple means of each group (what you would compute if you simply computed the mean for that group). Again, this is true for this model, but is not generally true. Despite the column label standing for “estimated marginal mean”, these are conditional means – the mean conditional on treatment level.

Also as in example 1, the SE for each mean is *not* the sample SE but the modeled SE – it is based on a pooled estimate of σ . *These are the SEs that you should report* because it is these SEs that are used to compute the p -value and CI that you report, that is, they tell the same “story”. The SE for the “1 hour cold” group is a bit higher because the sample size n for this group is smaller by 1.

10.1.4.3 The contrasts table

```
fig2a_m1_pairs
```

```
##   contrast           estimate    SE df lower.CL upper.CL t.ratio
## 1 hour cold - Control    7.14 4.57 14    -2.66    16.9 1.562
## 30 min NE - Control     14.79 4.36 14     5.45    24.1 3.395
## 30 min NE - 1 hour cold  7.65 4.57 14    -2.15    17.5 1.675
## p.value
```

10.1. A LINEAR MODEL WITH A SINGLE, CATEGORICAL X VARIABLE ESTIMATES THE EFFECTS OF THE

```
## 0.1405
## 0.0044
## 0.1162
##
## Confidence level used: 0.95
```

If a factor variable has more than two levels, there are multiple kinds of contrasts that a researcher can compare. The simplest are **pairwise contrasts**, which are differences between group means. These are typically called “post-hoc” tests in hypothesis testing but I avoid that because the focus in this text is estimation.

This table is important for reporting treatment effects and CIs and for plotting the model. As in example 1, the values in the column “estimate” are the simple differences between the means of the groups given in the contrast column (what you would compute if you simply computed the difference for these groups). Again, this is true for this model, but is not generally true.

The values in the “SE” column are standard errors of a difference (SED), specifically the difference in the estimate column. These SEs are from the fit model using the pooled estimate of σ and not the SED one would compute if one simply used the two groups in the contrast column. *These are the SEs that you should report* because it is these SEs that are used to compute the p -value and CI that you report, that is, they tell the same “story”.

The values in the “lower.CL” and “upper.CL” columns are the bounds of the 95% confidence interval of the estimate. Again, this confidence level applies to the procedure and not the estimate. Think of this interval as containing potential values of the true parameter (the true difference in means between the two groups) that are reasonably compatible with the data. More formally, it is correct to interpret this CI as “95% of the CIs computed using this procedure include the true value given the model conditions.”

The columns “t.ratio” and “p.value” contains the t and p values of the significance (not hypothesis!) test of the estimate. The t -statistic is the ratio of the estimate to the SE of the estimate (use the console to confirm this given the values in the table). It is a signal (the estimate) to noise (SE of the estimate) ratio. The p -value is the probability of sampling the same distribution, fitting the linear model, and observing a t -value as or more extreme than the observed t . A very small p -value is consistent with the experiment “not sampling from distributions with the same mean” – meaning that adding a treatment affects the mean of the distribution. This is the logic used to infer a treatment effect. Unfortunately, it is also consistent with the experiment not approximating other conditions of the model, including non-random assignment, non-independence, non-normal conditional responses, and variance heterogeneity. It is up to the rigorous researcher to be sure that these other model conditions are approximated or “good enough” to use the p -value to infer a treatment effect on the mean.

10.1.4.4 t and p from the contrasts table – when there are more than two levels in X – are not the same as those from pairwise t -tests among pairs of groups

The contrasts, CIs, and significance test statistics in the contrasts table come from a single model fit to the data. Researchers commonly fit separate t -tests for each pair of treatment levels instead of a single linear model.

```
# classic t-test
test1 <- t.test(fig2a[treatment == "1 hour cold", diHOME],
                 fig2a[treatment == "Control", diHOME],
                 var.equal=TRUE)

test2 <- t.test(fig2a[treatment == "30 min NE", diHOME],
                 fig2a[treatment == "Control", diHOME],
                 var.equal=TRUE)

test3 <- t.test(fig2a[treatment == "30 min NE", diHOME],
                 fig2a[treatment == "1 hour cold", diHOME],
                 var.equal=TRUE)

ttests <- data.frame(t = c(test1$statistic, test2$statistic, test3$statistic),
                      p = c(test1$p.value, test2$p.value, test3$p.value))
row.names(ttests) <- c("1 hour cold - Control",
                      "30 min NE - Control",
                      "30 min NE - 1 hour cold")
ttests

##          t      p
## 1 hour cold - Control  2.415122 0.038920739
## 30 min NE - Control   3.238158 0.008897132
## 30 min NE - 1 hour cold 1.380666 0.200700587
```

The t and p -values computed from three separate tests differ from the t and p -values computed from the single linear model shown in the contrasts table above. The values differ because the SE in the denominators used to compute the t -values differ. The t -value computed from the linear model use variation in all three groups to estimate σ^2 the variance of the response conditional on treatment level, and this estimate is common to all three t -tests (the SE differs slightly because of sample size differences among levels). The t -value computed from the separate tests each use variation from only the two groups in that test to estimate σ^2 . Consequently, σ^2 , and the SE of the estimate, differs for each test.

Using the linear model is a better practice than the pairwise t -tests. The reason is that the t -tests assume homogeneity of variance, that is, the σ^2 equal in all

three groups. And, since a t -test estimates σ^2 , it is more precise to estimate this using three groups (the linear model) than two groups (the pairwise t -test). And, because the three pairwise t -test computes three estimates of σ^2 , it is inconsistent to use one estimate in one test and a different estimate in a second test (why would we think the σ^2 for the Control group is two different values?).

10.1.4.5 The contrasts table – when there are more than two levels in X – has multiple p-values. How to handle this “multiple testing” is highly controversial

Multiple testing is the practice of adjusting p -values (and less commonly confidence intervals) to account for the expected increase in the frequency of Type I error in a batch, or *family*, of tests. An example of a batch of tests is the three tests in the contrast table for the analysis of the fig2a data. Multiple testing is a concept that exists because of Neyman-Pearson hypothesis testing strategy. My own belief is that the major problem of multiple testing is less the inflation of Type I error and more the idea that we “discover by p -value”. If we focus on effects and uncertainty in experiments where we have good prior knowledge to have reasonable expectations of these effects for the different treatment levels, we shouldn’t be concerned about inflated Type I error. If we use experiments to “see what happens” in tens, hundreds, thousands, or millions of response variables after we perturb the system, then multiple testing is more problematic. Issues surrounding multiple testing are fleshed out in more detail in Chapter xxx “Best Practices”. Computing adjusted values is covered below in the “Working in R” section.

10.2 Working in R

10.2.1 Specifying the contrasts

10.2.2 Adjustment for multiple comparisons

If the head of your group or a reviewer demands that you adjust p -values in data from an experiment like that for the fig2a data, then the `adjust` argument in `emmeans::contrast()` controls the method for p -value adjustment. The default is “tukey”.

1. “none” – no adjustment, in general my preference.
2. “tukey” – Tukey’s HSD
3. “bonferroni” – the standard bonferroni, which is conservative
4. “fdr” – the false discovery rate
5. “mvt” – based on the multivariate t distribution and using covariance structure of the variables

Here I use the Tukey HSD adjustment, which is a common choice for the “posthoc comparison of means” in experimental data like this. The Tukey adjustment is the default of `emmeans::contrast()` if the method of comparison is `revpairwise`, which means that the `adjust` argument does not need to be specified. That said, explicitly specify the adjustment as this makes the analysis more transparent.

```
(fig2a_m1_pairs_tukey <- contrast(fig2a_m1_emm,
                                    method = "revpairwise",
                                    adjust = "tukey") %>%
  summary(infer = TRUE))

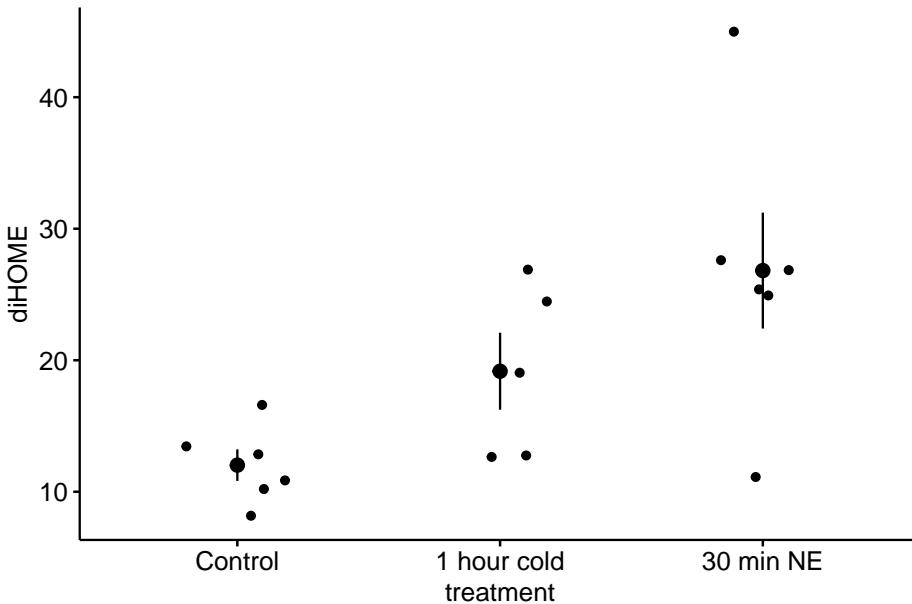
## #> #> contrast           estimate   SE df lower.CL upper.CL t.ratio
## #> 1 hour cold - Control    7.14 4.57 14   -4.82    19.1 1.562
## #> 30 min NE - Control     14.79 4.36 14    3.39    26.2 3.395
## #> 30 min NE - 1 hour cold   7.65 4.57 14   -4.31    19.6 1.675
## #> p.value
## #> 0.2936
## #> 0.0114
## #> 0.2490
## #>
## #> ## Confidence level used: 0.95
## #> ## Conf-level adjustment: tukey method for comparing a family of 3 estimates
## #> ## P value adjustment: tukey method for comparing a family of 3 estimates
```

10.2.3 Plotting models with a single, categorical X

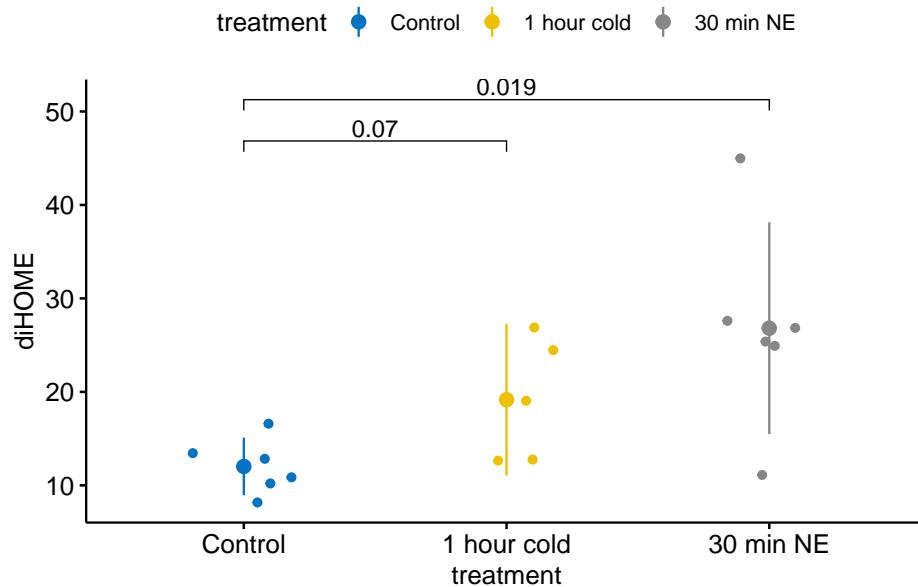
10.2.3.1 Response plot – Sample means and error bars using `ggpubr`

The package `ggpubr` makes it very easy to create a publishable plot.

```
gg_good <- ggstripchart(data = fig2a,
                         x = "treatment",
                         y = "diHOME",
                         add = "mean_se"
)
gg_good
```



With a little work, we can improve this. Note that I've changed the error interval from 1 SE to a 95% CI.



10.2.3.2 Response plot – Modeled CIs and custom p-values using ggpubr

The CI's and p -values computed using ggpubr are sample statistics and multiple, independent Welch t -tests and not from the linear model `lm(fig2a ~ treatment)`. Since the p -values are from Welch t -tests, the CIs and p -values are consistent in that they are using the same models to compute them.

ggpubr and `stat_compare_means()` have very limited flexibility in the means, CI's and p values that can be reported in a plot. This raises issues with best practices in this text, which advocates reporting modeled means, CIs and p -values. This includes adjusted p -values, which are not handled by `stat_compare_means()`. ggpubr has the function `stat_pvalue_manual()`, which is useful for reporting p -values from tests other than the limited number of tests available in `stat_compare_means()`. For the model used with fig2a, we can use ggpubr to construct the base plot of means and raw values. But with more complex models, we have to skip ggpubr altogether and use ggplot2 directly.

Step 1 – convert the “emm” (estimated marginal means) and “pairs” (contrast) tables to data.table. The “emm” table is ready after conversion to a data.table but the “pairs” table needs additional columns

1. create a column of pretty p -values that are rounded or converted to “ < 0.001 ” if small.
2. create the group columns containing the pair of groups that are compared. adding p -values to the graph requires a bracket to show which groups are

being compared. The column “group1” is added to list the “x value” of the first group. The column “group2” is added to list the “x value” of the second group. The “x value” of a group is the index of the group returned by the `levels()` function.

```
levels(fig2a$treatment)

## [1] "Control"      "1 hour cold"   "30 min NE"
```

As an example, the contrast in the first row of the contrast table is “1 hour cold - Control”. This is 2 - 1. The first element of Group1 is “2” and the first element of Group2 is “1”. ggpibr will use these columns to construct the brackets.

```
fig2a_m1_emm_dt <- summary(fig2a_m1_emm) %>%
  data.table
fig2a_m1_pairs_dt <- data.table(fig2a_m1_pairs)

# pvalString is from package lazyWeave
fig2a_m1_pairs_dt[, p.pretty := pvalString(p.value)]
# also create a column with "p-val: "
fig2a_m1_pairs_dt[, pval.pretty := paste("p-val:", p.pretty)]

# create group columns
fig2a_m1_pairs_dt[, group1 := c(2, 3, 3)]
fig2a_m1_pairs_dt[, group2 := c(1, 1, 2)]
```

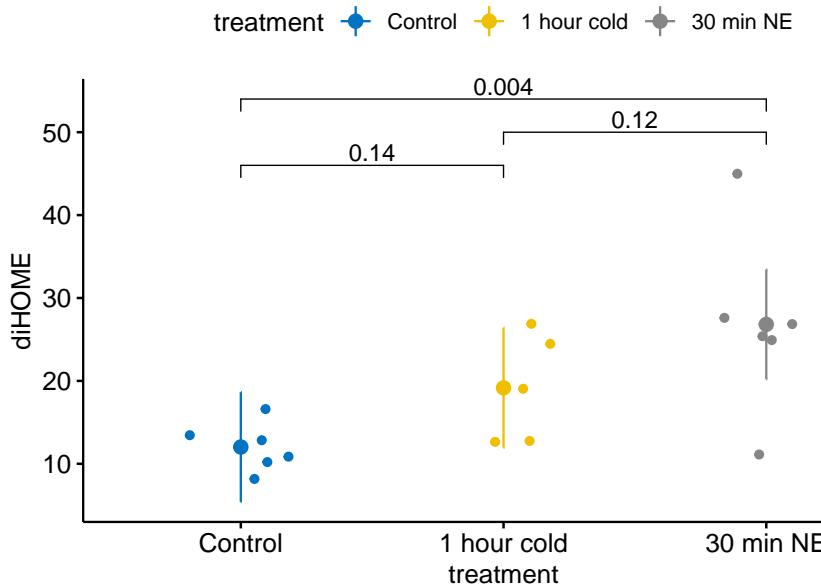
Step 2 – Add the CIs and *p*-value to a ggpibr plot

```
gg_almost_best <- ggstripchart(data = fig2a,
  x = "treatment",
  y = "diHOME",
  add = "mean",
  color = "treatment",
  palette = "jco"
) +
  geom_errorbar(data = fig2a_m1_emm_dt,
    aes(y = emmean,
        ymin = lower.CL,
        ymax = upper.CL,
        color = treatment),
    width = 0) +
  # only plotting 1st two p-values
```

```

stat_pvalue_manual(fig2a_m1_pairs_dt,
                   label = "p.pretty",
                   y.position = c(46, 54, 50)) +
  NULL # add to ease exploring plot components
gg_almost_best

```



Notes

1. The `y.position` argument in `stat_pvalue_manual()` contains the position on the y-axis for the p-value brackets. I typically choose these values “by eye”. Essentially, I look at the maximum y-value on the plot and then choose a value just above this for the first bracket.
2. All three p -values are shown. In general, all p -values (and means and CIs) should be reported but these can be in a supplemental table. To limit the p -values that are shown, use the row index of a data.table. For example, to show only the p -values with the control, use the following script (note that `y.position` was also changed to only show the y position of two brackets).

```

stat_pvalue_manual(fig2a_m1_pairs_dt[1:2,],
                   label = "p.pretty",
                   y.position = c(46, 50))

```

```
## mapping: xmin = ~xmin, xmax = ~xmax, label = ~label, y.position = ~y.position, vjust = ~vjust
```

```
## geom_bracket: type = text, na.rm = FALSE, coord.flip = FALSE
## stat_bracket: tip.length = 0.03, na.rm = FALSE
## position_identity
```

10.2.3.3 Response plot – Modeled means and CIs using ggplot2

The only difference between this and the “gg_almost_best” plot constructed above is that I am using the `ggplot` function to create the base plot (the axes) and `geom_sina` to plot the points in place of `ggstripchart` from the `ggpubr` package.

Step 1 - Create the data tables “fig2a_m1_emm_dt” and “fig2a_m1_pairs_dt” as above.

Step 2 - plot using the color-blind friendly Okabe-Ito paletter instead of “jco”.

```
gg_response <- ggplot(data = fig2a,
                       aes(x = treatment,
                            y = diHOME,
                            color = treatment)) +
  # points
  geom_sina(alpha = 0.5) + # ggforce package

  # plot means and CI
  geom_errorbar(data = fig2a_m1_emm_dt,
                aes(y = emmean,
                    ymin = lower.CL,
                    ymax = upper.CL,
                    color = treatment),
                width = 0
  ) +

  geom_point(data = fig2a_m1_emm_dt,
             aes(y = emmean,
                 color = treatment),
             size = 3
  ) +

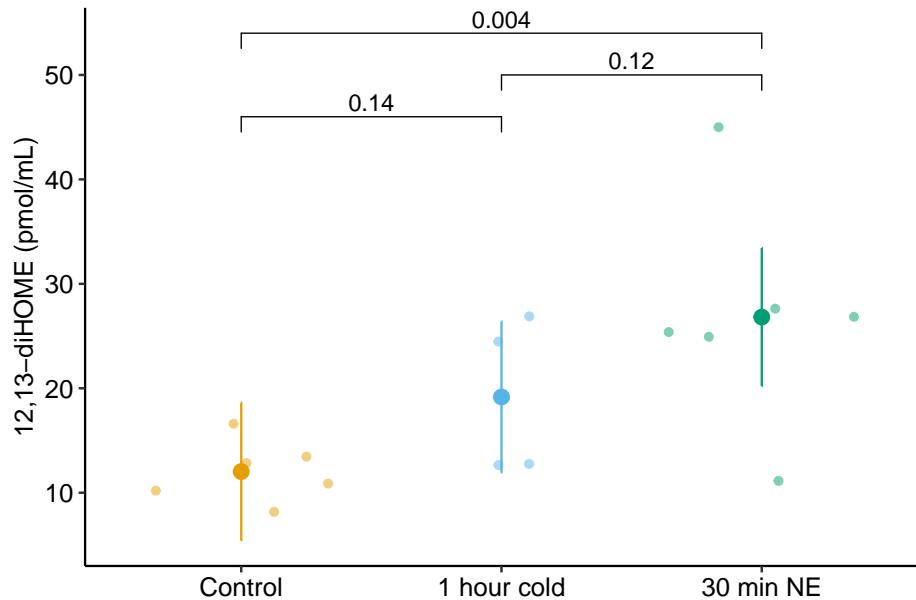
  # aesthetics
  ylab("12,13-diHOME (pmol/mL)") +
  scale_color_manual(values=pal_okabe_ito,
                     name = NULL) +
  theme_pubr() +
  theme(legend.position="none") +
  theme(axis.title.x=element_blank()) +
```

```
NULL

# create a version with the p-values. We want the p-valueless version for the combined

gg_response_p <- gg_response +
  # only plotting 1st two p-values
  stat_pvalue_manual(fig2a_m1_pairs_dt,
                      label = "p.pretty",
                      y.position = c(46, 54, 50))

gg_response_p
```



10.2.3.4 Effects plot

Effects plots commonly use the y-axis for the categorical variable (the contrast pairs) and the x-axis for the continuous variable (the effects).

```
gg_effect <- ggplot(data=fig2a_m1_pairs_dt,
                      aes(x = estimate,
                           y = contrast)) +
  # confidence level of effect
  geom_errorbar(aes(xmin=lower.CL,
                     xmax=upper.CL),
                width=0,
```

```
        color="black") +
# estimate of effect
geom_point(size = 3) +

# draw a line at effect = 0
geom_vline(xintercept=0, linetype = 2) +

# p-value. The y coordinates are set by eye
annotate(geom = "text",
         label = fig2a_m1_pairs_dt$p_pretty,
         y = 1:3,
         x = 28) +
annotate(geom = "text",
         label = "p-value",
         y = 3.25,
         x = 28) +

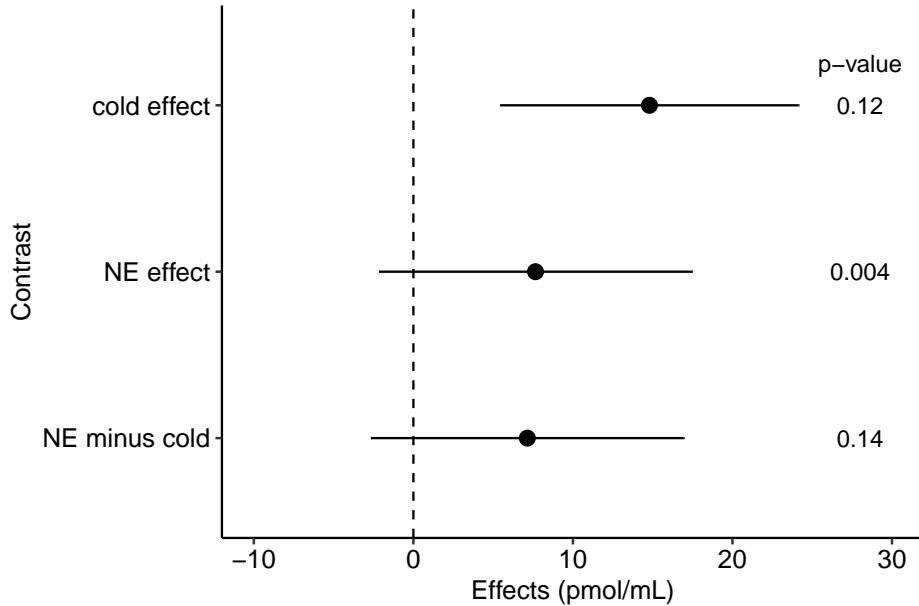
# contrast labels
scale_y_discrete(labels = c("NE minus cold",
                            "NE effect",
                            "cold effect"
)) +

# x-axis label and aesthetics
xlab("Effects (pmol/mL)") +
ylab("Contrast") +
coord_cartesian(xlim = c(-10,30)) +

# use ggpubr theme
theme_pubr() +

NULL

gg_effect
```



10.2.3.5 Combined effects-response plot

`plot_grid` from the `cowplot` package is used to create a combined effects-response plot by combining the `gg_effect` and `gg_response` plots. This looks best if the labels of the x-axis are set to “top” instead of “bottom”. Also we don’t need p-values on both components. The placement of “p-value” in the `gg_effect` plot above does not look good in the combined plot and I cannot remove this (easily) so I’m rebuilding the `gg_effect` component from scratch.

```
gg_effect <- ggplot(data=fig2a_m1_pairs_dt,
                      aes(x = estimate,
                           y = contrast)) +
  # confidence level of effect
  geom_errorbar(aes(xmin=lower.CL,
                     xmax=upper.CL),
                width=0,
                color="black") +
  # estimate of effect
  geom_point(size = 3) +
  # draw a line at effect = 0
  geom_vline(xintercept=0, linetype = 2) +
  # p-value. The y coordinates are set by eye
  annotate(geom = "text",
```

```

label = fig2a_m1_pairs_dt$p.pretty,
y = 1:3,
x = 28) +
# comment this out
# annotate(geom = "text",
#           label = "p-value",
#           y = 3.25,
#           x = 28) +
#
# contrast labels

scale_y_discrete(labels = c("NE minus cold",
                            "NE effect",
                            "cold effect"
)) +
# x-axis label and aesthetics
xlab("Effects (pmol/mL)") +
ylab("Contrast") +
coord_cartesian(xlim = c(-10,30)) +
# the new code
scale_x_continuous(breaks = c(-10, 0, 10, 20, 28),
                   labels = c("-10", "0", "10", "20", "p-value"),
                   position = "top") + # move to top

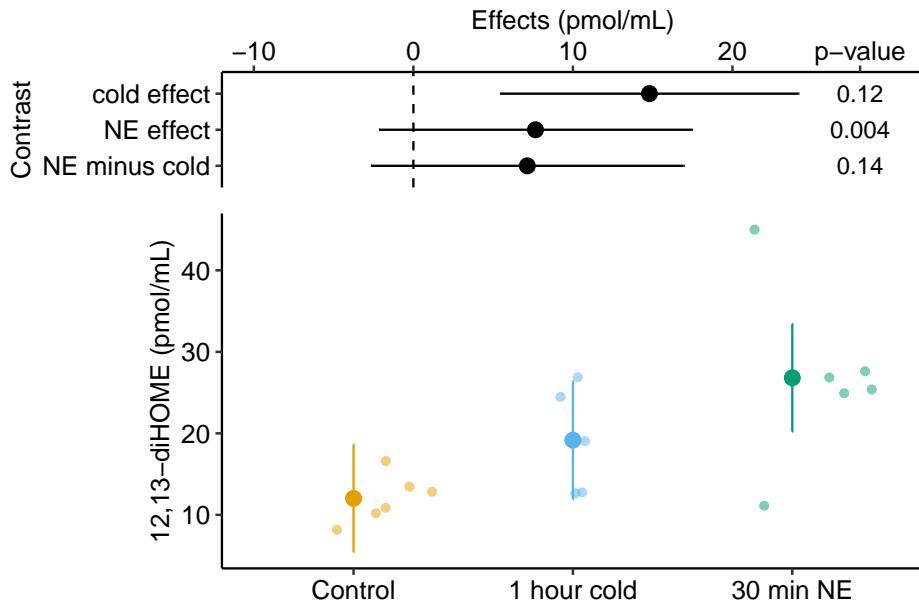
# use ggpubr theme
theme_pubr() +
NULL

```

```

plot_grid(gg_effect,
          gg_response,
          nrow=2,
          align = "v",
          axis = "lr",
          rel_heights = c(0.5,1))

```



10.3 Issues in inference in models with a single, categorical X

10.3.1 Lack of independence

The data from the experiment for Figure 1b of the 12,13-diHOME article outlined above are the plasma concentrations of 12,13-diHOME in humans in response to either saline or one-hour cold challenge. The response variable (*diHOME*) is not independent because

```
# fit the model
m1 <- lmer(diHOME ~ treatment + (1|id), data = fig1b)

# estimated marginal means table
m1_emm <- emmeans(m1, specs = "treatment")

# contrasts table
(m1_pairs <- contrast(m1_emm,
                        method = "revpairwise") %>%
  summary(infer = TRUE))

##  contrast      estimate       SE df lower.CL upper.CL t.ratio p.value
##  cold - saline    0.234  0.0549   8     0.108    0.361  4.272  0.0027
```

```

## Degrees-of-freedom method: kenward-roger
## Confidence level used: 0.95

t.test(x = fig1b[treatment == "cold", diHOME],
       y = fig1b[treatment == "saline", diHOME],
       paired = TRUE)

## Paired t-test
##
## data: fig1b[treatment == "cold", diHOME] and fig1b[treatment == "saline", diHOME]
## t = 4.2722, df = 8, p-value = 0.002716
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
## 0.1078778 0.3609173
## sample estimates:
## mean of the differences
## 0.2343975

```

10.3.2 Heterogeneity of variances

Textbooks that use a “which test?” strategy often point to a Welch’s t -test in place of Student’s t -test if there is heterogeneity of variances between treatment groups. A Welch t -test is infrequent in the experimental biology literature, perhaps because

1. it is poorly known and it doesn’t occur to researchers to use a test that models heterogeneity of variances.
2. many experiments have more than two levels, or are factorial, and these are often analyzed with ANOVA instead of multiple t -tests.
3. heterogeneity often arises in right-skewed data, which is often analyzed with a non-parametric test like the Mann-Whitney U test.

The Welch t -test is a special case of a linear model that explicitly models the within-group variance using **generalized least squares** (GLS). The 95% CI of a mean differences and p -values from the fit gls linear model and from Welch’s t -test are the same. Advantages of using a linear modeling strategy is that a researcher uses the model to estimate effects (difference in means) and measures of uncertainty in the effects (standard errors or confidence intervals of the difference). Advantages of specifically using the gls extension of the linear model is that it is it can be easily expanded to analyze more complex designs including 1) more than two treatment groups, 2) more than one factor, and 3) additional covariates.

300CHAPTER 10. LINEAR MODELS WITH A SINGLE, CATEGORICAL X

Modeling variance heterogeneity is the focus of chapter xxx so the account here is brief. Heterogeneity can be modeled using a generalized least squares linear model with the `gls` function. The `weights` argument is used to model the variances using each group's sample variance. In this example, I use the data from the Figure 1b experiment, which can be compared to the analysis of the same data in Example 2 above.

```
subdata <- fig2a[is.na(diHOME) == FALSE,] # omit rows with missing data
fig2a_m2 <- gls(diHOME ~ treatment,
                 data = subdata,
                 weights = varIdent(form = ~ 1 | treatment))
```

The model `fig2a_m2` uses variance computed in each group separately as the estimate of σ^2 for that group. The coefficient table of the GLS model is

```
summary(fig2a_m2) %>%
  coef()

##                               Value Std.Error   t-value   p-value
## (Intercept)           12.023075  1.200139 10.018072 9.134757e-08
## treatment1 hour cold  7.140386  3.163467  2.257139 4.050473e-02
## treatment30 min NE   14.794354  4.568757  3.238157 5.951274e-03
```

Notes

1. **Important** for reporting p -values. Unlike the linear model modeling homogenous variance, the p -values for the coefficients of *treatment1 hour cold* and *treatment30 min NE* are *not* the same as the p -values of these equivalent contrasts in the contrasts table (see below). The reason is, the computation of the p -value in the two tables use two different degrees of freedom. Report the p -values from the contrast table using the Satterthwaite df.

The modeled means and contrasts are computed as above for the `lm` object

```
fig2a_m2_emm <- emmeans(fig2a_m2, specs="treatment")
fig2a_m2_emm

##   treatment   emmean    SE df lower.CL upper.CL
##   Control      12.0 1.20  5     8.94    15.1
##   1 hour cold  19.2 2.93  4    11.04    27.3
##   30 min NE    26.8 4.41  5    15.49    38.1
##
## Degrees-of-freedom method: satterthwaite
## Confidence level used: 0.95
```

10.3. ISSUES IN INFERENCE IN MODELS WITH A SINGLE, CATEGORICAL X 301

Notes

1. The SE of the means in this table are modeled SEs but are equal to the sample SE of the means, because this was specified in the GLS model.

```
fig2a_m2_pairs <- contrast(fig2a_m2_emm,
                           method = "revpairwise",
                           adjust = "none") %>%
  summary(infer = TRUE)
fig2a_m2_pairs

##   contrast           estimate    SE   df lower.CL upper.CL t.ratio
## 1 hour cold - Control     7.14 3.16 5.34   -0.839    15.1 2.257
## 30 min NE - Control      14.79 4.57 5.74    3.490    26.1 3.238
## 30 min NE - 1 hour cold    7.65 5.29 8.35   -4.460    19.8 1.446
##   p.value
## 0.0703
## 0.0189
## 0.1845
##
## Degrees-of-freedom method: satterthwaite
## Confidence level used: 0.95
```

Compare this table to the three Welch t -tests of all pairs of treatment levels in the fig2a experiment.

```
test1 <- t.test(fig2a[treatment == "1 hour cold", diHOME],
                 fig2a[treatment == "Control", diHOME],
                 var.equal=FALSE)

test2 <- t.test(fig2a[treatment == "30 min NE", diHOME],
                 fig2a[treatment == "Control", diHOME],
                 var.equal=FALSE)

test3 <- t.test(fig2a[treatment == "30 min NE", diHOME],
                 fig2a[treatment == "1 hour cold", diHOME],
                 var.equal=FALSE)

welch_tests <- data.frame(t = c(test1$statistic, test2$statistic, test3$statistic),
                           p = c(test1$p.value, test2$p.value, test3$p.value))
row.names(welch_tests) <- c("1 hour cold - Control",
                           "30 min NE - Control",
                           "30 min NE - 1 hour cold")
welch_tests
```

302CHAPTER 10. LINEAR MODELS WITH A SINGLE, CATEGORICAL X

```
##          t      p
## 1 hour cold - Control 2.257139 0.07026220
## 30 min NE - Control   3.238158 0.01889041
## 30 min NE - 1 hour cold 1.446454 0.18451549
```

The t and p -values computed from the GLS linear model and from the three, pairwise Welch t -tests are the same (to about the 6th decimal place). They are the same because each is estimating σ^2 separately for each group and not as the pooled (among two groups for t -test or three groups for the linear model) estimate and because they use the same degrees of freedom to compute the p -value.

Let's summarize these comparisons

1. Inference from a linear model using homogenous variance (the `lm` function) and from a Student's t -test are the same if there are only two levels in the treatment variable.
2. Inference from a linear model using homogenous variance (the `lm` function) and from the series of pairwise, Student's t -tests differ when there are more than two levels in the treatment variable.
3. Inference from a GLS linear model using heterogenous variance (the `gls` function) and from a Welch t -test are the same regardless of the number of levels in the treatment variable.

Even though the linear model that models heterogeneity and the Welch t -test produce the same results, researchers should use the linear model because

1. A linear modeling strategy encourages researchers to think about the effect and uncertainty in the effect and not just a p -value.
2. The linear model is nearly infinitely flexible and expandible while the t -test has extremely limited flexibility (The Welch t -test is one way to expand the classical, Student's t -test).

10.3.3 The conditional response isn't Normal

10.3.4 Pre-post designs

10.3.5 Longitudinal designs

10.3.6 Comparing responses normalized to a standard

10.3.7 Comparing responses that are ratios

10.3.8 Researcher degrees of freedom

Conspicuously, the p -value for the “1 hour cold - Control” contrast is 0.039, which is “significant” using the conventional

10.4 Hidden Code

10.4.1 fig2a data

The script for plotting the model in the section Hidden code below.

```
fig2a_m1_emm_dt <- summary(fig2a_m1_emm) %>%
  data.table
fig2a_m1_pairs_dt <- data.table(fig2a_m1_pairs)
fig2a_m1_pairs_dt[ , p.pretty := pvalString(p.value)]
```



```
fig2a_gg_response <- ggplot(data = fig2a,
                           aes(x = treatment,
                                 y = diHOME,
                                 color = treatment)) +
```



```
# points
geom_sina(alpha = 0.5) +
```



```
# plot means and CI
geom_errorbar(data = fig2a_m1_emm_dt,
               aes(y = emmean,
                     ymin = lower.CL,
                     ymax = upper.CL,
                     color = treatment),
               width = 0
) +
```

```

geom_point(data = fig2a_m1_emm_dt,
            aes(y = emmean,
                color = treatment),
            size = 3
) +
# aesthetics
ylab("12,13-diHOME (pmol/mL)") +
scale_color_manual(values=pal_okabe_ito,
                    name = NULL) +
theme_pubr() +
theme(legend.position="none") +
theme(axis.title.x=element_blank()) +
NULL

#fig2a_gg_response

# reverse order of rows so flipped plot has original order
fig2a_m1_pairs_dt <- fig2a_m1_pairs_dt[rev(1:N)]

# order "contrast" factor as in table!
contrast_order <- fig2a_m1_pairs_dt[, contrast]
fig2a_m1_pairs_dt[, contrast := factor(contrast, contrast_order)]

min_bound <- min(fig2a_m1_pairs_dt[, lower.CL])
max_bound <- min(fig2a_m1_pairs_dt[, upper.CL])
y_lo <- min(min_bound+min_bound*0.2,
             -max_bound)
y_hi <- max(max_bound + max_bound*0.2,
             -min_bound)
y_lims <- c(y_lo, y_hi)

fig2a_gg_effect <- ggplot(data=fig2a_m1_pairs_dt,
                            aes(x = contrast,
                                y = estimate)) +
# confidence level of effect
geom_errorbar(aes(ymin=lower.CL,
                   ymax=upper.CL),
              width=0,
              color="black") +
# estimate of effect
geom_point(size = 3) +

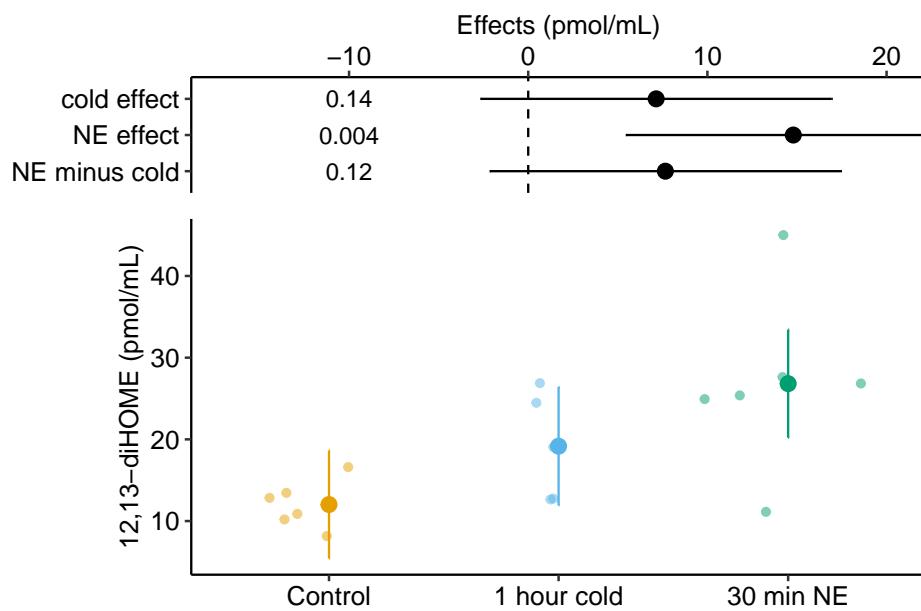
```

```
# zero effect
geom_hline(yintercept=0, linetype = 2) +
# p-value
annotate(geom = "text",
         label = fig2a_m1_pairs_dt$p.pretty,
         x = 1:3,
         y = -10) +
# aesthetics
scale_y_continuous(position="right") +
scale_x_discrete(labels = c("NE minus cold",
                            "NE effect",
                            "cold effect"))
)) +
coord_flip(ylim = y_lims) +
theme_pubr() +
ylab("Effects (pmol/mL)") +
theme(axis.title.y = element_blank()) +
NULL

# fig2a_gg_effect
```

```
fig2a_fig <- plot_grid(fig2a_gg_effect,  
                      fig2a_gg_response,  
                      nrow=2,  
                      align = "v",  
                      axis = "lr",  
                      rel_heights = c(0.5,1))  
  
fig2a_fig
```

306 CHAPTER 10. LINEAR MODELS WITH A SINGLE, CATEGORICAL X



Chapter 11

Model Checking

11.1 All statistical analyses should be followed by model checking

We us a linear model to infer effects or predict future outcomes. Our inference is uncertain. Given the model assumptions, we can quantify this uncertainty with standard errors, and from these standard errors we can compute confidence intervals and p -values. It is good practice to use a series of **diagnostic plots**, diagnostic statistics, and simulation to check how well the data approximate the fit model and model assumptions. **Model checking** is used to both check our subjective confidence in the modeled estimates and uncertainty and to provide empirical evidence for subjective decision making in the analysis workflow.

NHST blues – Researchers are often encouraged by textbooks, colleagues, or the literature to test the assumptions of a t -test or ANOVA with formal hypothesis tests of distributions such as a Shapiro-Wilks test of normality or a Levine test of homogeneity. In this strategy, an alternative to the t -test/ANOVA is used if the distribution test's p -value is less than some cut-off (such as 0.05). Common alternatives include 1) transformations of the response to either make it more normal or the variances more homogenous, 2) implementation of alternative tests such as a Mann-Whitney-Wilcoxon (MWW) test for non-normal data or a Welch t -test/ANOVA for heterogenous variances. The logic of a test of normality or homogeneity before a t -test/ANOVA isn't consistent with frequentist thinking because the failure to reject a null hypothesis does not mean the null hypothesis is true. We shouldn't conclude that a sample is "normal" or that the variances are "homogenous" because a distributional test's p -value > 0.05 . But, maybe we should of the distributional pre-test as an "objective" model check? The logic of this objective decision rule suffers from several issues. **First**, the subsequent p -value of the t test/ANOVA test is not valid because this p -value is the long-run frequency of a test-statistic as large or larger than the

observed statistic conditional on the null – not conditional on the subset of nulls with $p > 0.05$ in the distribution test. **Second**, real data are only approximately normal; with small n , it will be hard to reject the null of a normal distribution because of low power, but, as n increases, a normality test will reject any real dataset. **Third**, and most importantly, our analysis should follow the logic of our goals. If our goal is the estimation of effects, we cannot get meaningful estimates from a non-parametric test (with a few exceptions) or a transformed response, as these methods are entirely about computing a “correct” p -value. Good alternatives to classic non-parametric tests and transformations are bootstrap estimates of confidence limits, permutation tests, and generalized linear models.

11.2 Linear model assumptions

To facilitate explanation of assumptions of the linear model and extensions of the linear model, I will use both the error-draw and conditional-draw specifications of a linear model with a single X variable.

error draw:

$$y = \beta_0 + \beta_1 x_i + \varepsilon_i \quad (11.1)$$

$$\varepsilon_i \sim N(0, \sigma^2) \quad (11.2)$$

conditional draw:

$$y_i \sim N(\mu_i, \sigma^2) \quad (11.3)$$

$$E(Y|X = x_i) = \mu_i \quad (11.4)$$

$$\mu_i = \beta_0 + \beta_1 x_i \quad (11.5)$$

This model generates random data using the set of rules specified in the model equations. To quantify uncertainty in our estimated parameters, including standard errors, confidence intervals, and p -values, we make the assumption that the data from the experiment is a random sample generated using these rules.

The two rules specified in the model above (Model (??)) are

1. The systematic component of data generation is $\beta_0 + \beta_1 X$.
- More generally, all linear models in this text specify systematic components that are *linear in the parameters*. Perhaps a better name for this is “additive in the parameters”. Additive (or linear) simply means that we can add up the products of a parameter and an X variable to get the conditional expectation $E(Y|X)$.

- For observational inference, the rule $E(Y|see\ X = x_i) = \mu_i$ is sufficient. For causal inference with data from an experiment, or with observational data and a well defined causal diagram, we would need to modify this to $E(Y|do\ X = x_i) = \mu_i$.
2. The stochastic component of data generation is “IID Normal”, where IID is **independent and identically distributed** and Normal refers to the Normal (or Gaussian) distribution. The IID assumption is common to all linear models. Again, for the purpose of this text, I define a “linear model” very broadly as a model that is linear in the parameters. This includes many extensions of the classical linear model including generalized least squares linear models, linear mixed models, generalized additive models, and generalized linear models. **Parametric** inference form all linear models requires the specification of a distribution family to sample. Families that we will use in this book include Normal, gamma, binomial, poisson, and negative binomial. This text will also cover **distribution free** methods of quantifying uncertainty using the bootstrap and permutation, which do not specify a sampling distribution family.

11.2.1 A bit about IID

1. **Independent** means that the random draw for one case cannot be predicted from the random draw of any other case. A lack of independence creates *correlated error*. There are lots or reasons that errors might be correlated. If individuals are measured both within and among cages, or tanks, or plots, or field sites, then we'd expect the measures within the same unit (cage, tank, plot, site) to err from the model in the same direction because of environmental features shared by individuals within the unit but not by individuals in other units. Multiple measures within experimental units create “clusters” of error. Lack of independence or clustered error can be modeled using **generalized least squares (GLS)** models that directly model the structure of the error and with **random effects** models. Random effects models go by many names including linear mixed models (common in Ecology), hierarchical models, and multilevel models. Both GLS and random effects models are variations of linear models.

tl;dr – Measures taken within the same individual over time (*repeated measures*) are correlated and are common in all areas of biology. In ecology and evolutionary studies, measures that are taken from sites that are closer together or measures taken closer in time or measures from more closely related biological species will tend to have more similar error than measures taken from sites that are further apart or from times that are further apart or from species that are less closely related. Space and time and phylogeny create **spatial, temporal, and phylogenetic autocorrelation**.

2. **Identical** means that all random draws at a given value of X are from the same distribution. Using the error-draw specification of the model above, this can be restated as, the error-draw (ε_i) for every i is from the same distribution $N(0, \sigma^2)$. Using the conditional-draw specification, this can be restated as, the random-draw y_i for every i with the same expected value $\mu = \mu_i$ is from the same distribution $N(\mu_i, \sigma^2)$. Understand the importance of this. Parametric inference using this model assumes that the sampling variance of μ at a single value of X is the same for all values of X . If X is continuous, this means the spread of the points around the regression line is the same at all values of X in the data. If X is categorical, this means the spread of the points around the mean of a group is the same for all groups. A consequence of “identical”, then, for all classical linear models, is the assumption of homogeneity (or **homoskedasticity**) of variance. If the sampling variance differs among the X , then the variances are heterogenous or **heteroskedastic**. Experimental treatments can affect the variance of the response in addition to the mean of the response. Heterogenous variance can be modeled using **Generalized Least Squares (GLS)** linear models. Many natural biological processes generate data in which the error is a function of the mean. For example, measures of biological variables that grow, such as size of body parts, have variances that “grow” with the mean. Or, measures of counts, such as the number of cells damaged by toxin, the number of eggs in a nest, or the number of mRNA transcripts per cell have variances that are a function of the mean. Both growth and count measures can sometimes be reasonably modeled using a linear model but more often, they are better modeled using a **Generalized Linear Model (GLM)**.

11.3 Diagnostic plots use the residuals from the model fit

11.3.1 Residuals

A residual of a statistical model is $y_i - \hat{y}_i$. Remember that \hat{y}_i is the predicted value of Y when X has the value x_i (compactly written as $X = x_i$). And remember that \hat{y}_i is the estimate of μ_i . For linear models (but not generalized linear models), the residuals of the fit model are estimates of the ε in equation (11.2). This *is not* true for generalized linear models because GLMs are not specified using (11.2).

Alert A common misconception is that inference from a linear model assumes that the *response* (the measured Y) is IID Normal. This is wrong. Either specification of the linear model shows precisely why this conception is wrong. Model (11.2) explicitly shows that it is the error that has the normal distribution – the distribution of Y is a mix of the distribution of X and that of the error. A more

11.3. DIAGNOSTIC PLOTS USE THE RESIDUALS FROM THE MODEL FIT311

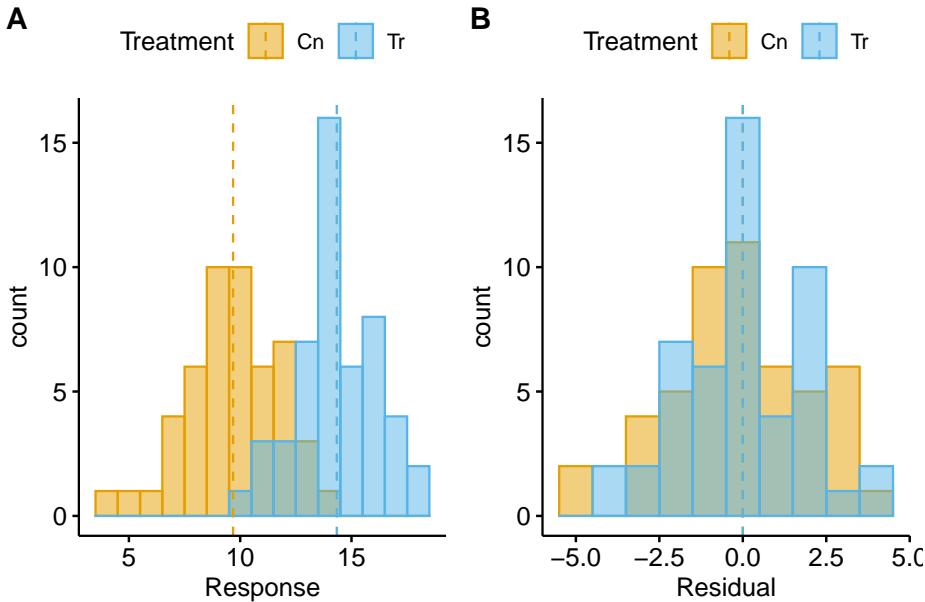


Figure 11.1: (#fig:model-check-histogram, model-check-residuals1) Histogram of the (A) response, showing modes near the true means of each group and (B) residuals, with a mode for both groups at zero.

general way of thinking about the assumed distribution uses the specification in model (11.5), which shows that it is the *conditional* response that is assumed to be IID normal. Remember, a conditional response (y_i) is a random draw from the infinite set of responses at a given value of X .

Let's look at the distribution of residuals versus the distribution of responses for a hypothetical experiment with a single, categorical X variable (the experimental factor) with two levels ("Cn" for control and "Tr" for treatment). The true parameters are $\beta_0 = 10$ (the true mean for the control group, or μ_0), $\beta_1 = 4$ (the difference between the true mean for the treatment minus the true mean for the control, or $\mu_1 - \mu_0$), and $\sigma = 2$ (the error standard deviation).

The plot above shows a histogram of the response (A) and residuals (B). In the plot of the response, the mode (the highest bar, or bin with the most cases) includes true mean for each group. And, as expected given $\beta_1 = 4$, the modes of the two groups are 4 units apart. It should be easy to see from this plot that the response does not have a normal distribution. Instead, it is distinctly bimodal. But the distribution of the response *within* each level looks like these are drawn from a normal distribution – and it should. In the plot of the residuals, the values of both groups are shifted so that the mean of each group is at zero. The consequence of the shift is that the combined set of residuals does look like it is drawn from a Normal distribution.

The two plots suggest two different approaches for model checking. First, we could examine the responses within each level of the experimental factor. Or, second, we could examine the residuals of the fit model, ignoring that the residuals come from multiple groups. The first is inefficient because it requires as many checks as there are levels in the factor. The second requires a single check.

Alert Some textbooks that recommend formal hypothesis tests of normality recommend the inefficient, multiple testing on each group separately. This isn't wrong, it's just more work than it needs to be and also suffers from "multiple testing".

11.3.2 A Normal Q-Q plot is used to check for characteristic departures from Normality

A Normal Q-Q plot is a scatterplot of

1. **sample quantiles** on the y axis. The sample quantiles is the vector of N residuals in rank order, from smallest (most negative) to largest (most positive). Sometimes this vector is standardized by dividing the residual by the standard deviation of the residuals (doing this makes no difference to the interpretation of the Q-Q plot).
2. **standard normal quantiles** on the x axis. This is the vector of standard, Normal quantiles given N elements in the vector. "Standard Normal" means a normal distribution with mean zero and standard deviation (σ) one. A Normal quantile is the expected deviation given a probability. For example, if the probability is 0.025, the Normal quantile is -1.959964. Check your understanding: 2.5% of the values in a Normal distribution with mean 0 and standard deviation one are more negative than -1.959964. The Normal quantiles of a Normal Q-Q plot are computed for the set of N values that evenly split the probability span from 0 to 1. For $N = 20$, this would be

```
p <- data.frame(quantile = qnorm(ppoints(1:20)))
row.names(p) <- ppoints(1:20)
```

```
p
```

```
##           quantile
## 0.025 -1.95996398
## 0.075 -1.43953147
## 0.125 -1.15034938
## 0.175 -0.93458929
## 0.225 -0.75541503
## 0.275 -0.59776013
## 0.325 -0.45376219
```

11.3. DIAGNOSTIC PLOTS USE THE RESIDUALS FROM THE MODEL FIT313

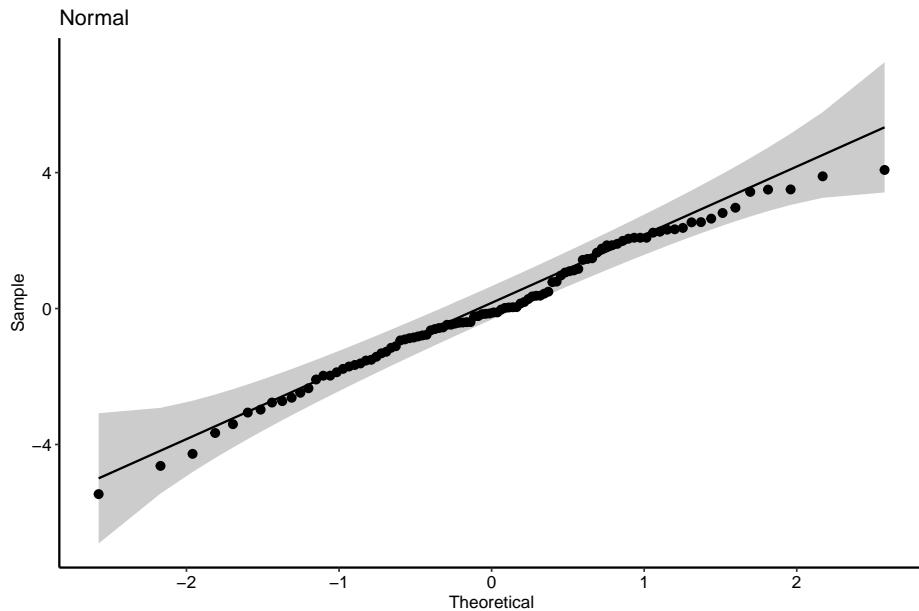
```
## 0.375 -0.31863936
## 0.425 -0.18911843
## 0.475 -0.06270678
## 0.525  0.06270678
## 0.575  0.18911843
## 0.625  0.31863936
## 0.675  0.45376219
## 0.725  0.59776013
## 0.775  0.75541503
## 0.825  0.93458929
## 0.875  1.15034938
## 0.925  1.43953147
## 0.975  1.95996398
```

A Normal Q-Q plot is not a test if the data are Normal. Instead, a Normal Q-Q plot is used to check for characteristic departures from Normality that are signatures of certain well-known distribution families. A researcher can look at a QQ-plot and reason that a departure is small and choose to fit a classic linear model using the Normal distribution. Or, a researcher can look at a QQ-plot and reason that the departure is large enough to fit a generalized linear model with a specific distribution family.

Stats 101 A quantile is the value of a distribution that is greater than p percent of the values in the distribution. The 2.5% quantile of a uniform distribution from 0 to 1 is 0.025. The 2.5% quantile of a standard normal distribution is -1.96 (remember that 95% of the values in a standard normal distribution are between -1.96 and 1.96). The 50% quantile of a uniform distribution is 0.5 and the 50% quantile of a standard normal distribution is 0.0 (this is the median of the distribution – 50% of the values are smaller and 50% of the values are larger).

Stats 201 A Q-Q plot more generally is a scatter plot of two vectors of quantiles either of which can come from a sample or a theoretical distribution. In the GLM chapter, the text will introduce Q-Q plots of residual quantiles transformed to have an expected uniform distribution. These are plotted against theoretical uniform quantiles from 0 to 1.

11.3.2.1 Normal QQ-plot of the fake data generated above



If the sampled distribution approximates a sample from a normal distribution, the scatter should fall along a line from the bottom, left to the top, right of the plot. The interpretation of a normal Q-Q plot is enhanced with a line of “expected values” of the sample quantiles if the sample residuals are drawn from a normal distribution. The closer the sample quantiles are to the line, the more closely the residuals approximate the expectation from a normal distribution. Because of sampling, the sampled values always deviate from the line, especially at the ends. The shaded gray area in the Q-Q plot in Figure ?? are the 95% confidence bands of the quantiles. A pattern of observed quantiles with some individual points outside of these boundaries indicates a sample that would be unusual if sampled from a Normal distribution.

Biological datasets frequently have departures on a Normal Q-Q plot that are characteristic of specific distribution families, including lognormal, binomial, poisson, negative binomial, gamma, and beta. It is useful to learn how to read a Normal Q-Q plot to help guide how to model your data.

What about the interpretation of the Q-Q plot in Figure ??? At the small end of the distribution (bottom-left), the sample values are a bit more negative than expected, which means the left tail is a bit extended. At the large end (upper-right), the sample values are, a bit less positive than expected, which means the right tail is a bit shortened. This is a departure in the direction of a left skewed distribution. Should we fit a different model given these deviations? To guide us, we compare the quantiles to the 95% confidence band of the quantiles. Clearly the observed quantiles are within the range of quantiles that we'd expect

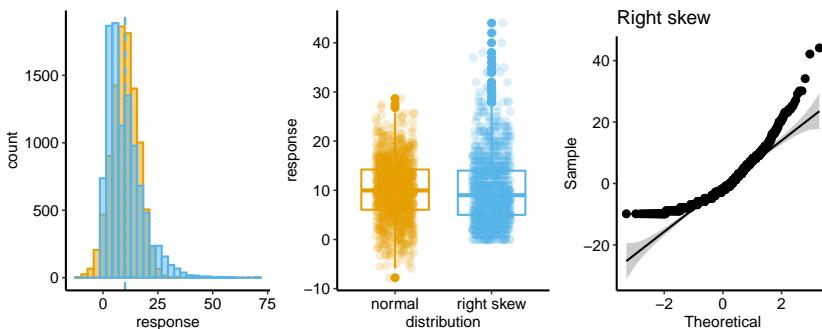
if sampling from a Normal distribution.

11.3.3 Mapping QQ-plot departures from Normality

Let's look at simulated samples drawn from non-normal distributions to identify their characteristic deviations. Each set of plots below shows

1. (left panel) A histogram of 10,000 random draws from the non-Normal distribution (blue). This histogram is superimposed over that of 10,000 random draws from a Normal distribution (orange) with the same mean and variance as that of the non-normal distribution.
2. (middle panel) Box plots and strip chart of a random subset ($N = 1000$) of data in the left panel.
3. (right panel) Normal Q-Q plot of the non_Normal data only.

Skewed-Right Q-Q



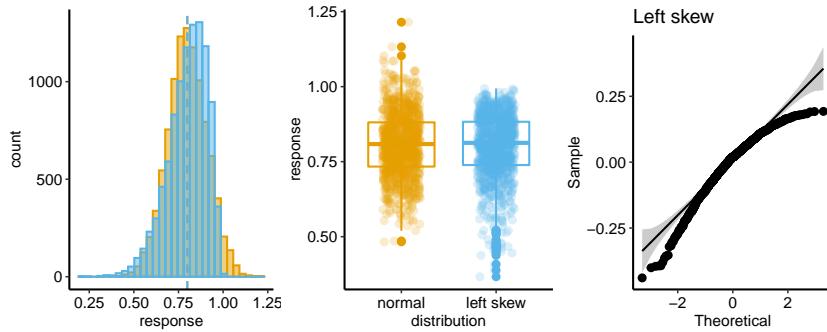
The Normal Q-Q plot of a sample from a right-skewed distribution is characterized by sample quantiles at the high (right) end being more positive than the expected Normal quantiles. Often, the quantiles at the low (left) end are also less negative than the expected normal quantiles. The consequence is a concave up pattern.

The histograms in the left panel explain this pattern. The right tail of the skewed-right distribution extends further than the right tail of the Normal. It is easy to see from this that, if we rank the values of each distribution from small to large (these are the quantiles), the upper quantiles of the skewed-right distribution will be larger than the matching quantile of the Normal distribution. For example, the 99,990th quantile for the skewed-right distribution will be much more positive than the 99,990th quantile for the Normal distribution. The opposite occurs at the left tail, which extends further in the negative direction in the Normal than the skewed-right distribution.

The middle panel compares a boxplot and stripchart of samples from the two distributions to show what researchers should look for in their own publication-ready plots as well as the published plots of colleagues. The skewed-right plot

exhibits several hallmarks of a skewed-right distribution including 1) a median line (the horizontal line within the box) that is closer to the 25th percentile line (the lower end of the box) than to the 75th percentile line (the upper end of the box), 2) a longer upper than lower whisker (the vertical lines extending out of the box), 3) more outliers above the upper whisker than below the lower whisker, and 4) a lengthened, upward smear of the scatter of points at the high end of the values, relative to the more compact smear at the low end of the values.

Skewed-Left Q-Q



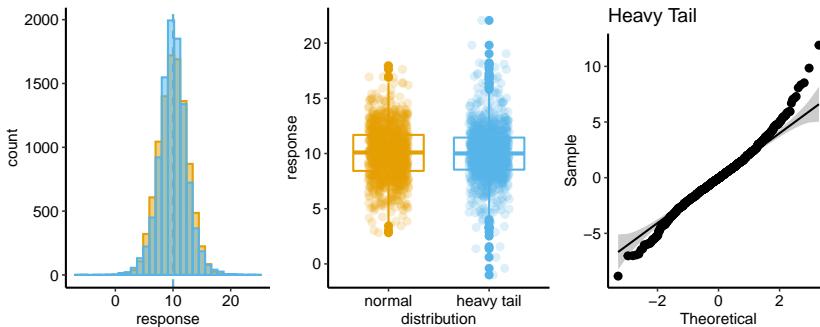
The Normal Q-Q plot of a sample from a left-skewed distribution is characterized by sample quantiles at the low (left) end being more negative than the expected Normal quantiles. Often, the quantiles at the high (right) end are also less positive than the expected normal quantiles. The consequence is a concave down pattern.

The histograms in the left panel explain this pattern. The left tail of the skewed-left distribution extends further than the left tail of the Normal. It is easy to see from this that, if we rank the values of each distribution from small to large (these are the quantiles), the lower quantiles of the skewed-left distribution will be more negative than the matching quantile of the Normal distribution. For example, the 10th quantile for the skewed-left distribution will be much more negative than the 10th quantile for the Normal distribution. The opposite occurs at the right tail, which extends further in the positive direction in the Normal than the skewed-left distribution.

The skewed-left plot in the middle panel highlights several hallmarks of a skewed-left distribution including 1) a median line (the horizontal line within the box) that is closer to the 75th percentile line (the lower end of the box) than to the 25th percentile line (the upper end of the box), 2) a longer lower than upper whisker (the vertical lines extending out of the box), 3) more outliers below the lower whisker than above the upper whisker, and 4) a lengthened, downward smear of the scatter of points at the low end of the values, relative to the more compact smear at the upper end of the values.

Heavy Tail Q-Q

11.3. DIAGNOSTIC PLOTS USE THE RESIDUALS FROM THE MODEL FIT 317



The Normal Q-Q plot of a sample from a heavy-tail distribution is characterized by sample quantiles at the low (left) end being more negative than the expected Normal quantiles and quantiles at the high (right) end that are more positive than the expected normal quantiles.

The histograms in the left panel explain this pattern. At each tail, the heavy-tail distribution has more **density** – there are more values far from the mean – compared to the Normal distribution. This is the origin of “heavy tail”. It is easy to see from this that, if we rank the values of each distribution from small to large (these are the quantiles), the lower quantiles of the heavy tail distribution will be more negative than the matching quantile of the Normal distribution. For example, the 10th quantile for the heavy-tail distribution will be much more negative than the 10th quantile for the Normal distribution. Likewise, the upper quantiles of the heavy tail distribution will be more positive than the matching quantile of the Normal distribution. For example, the 99,990th quantile for the heavy-tail distribution will be much more positive than the 99,990th quantile for the Normal distribution.

The heavy-tail plot in the middle panel shows more boxplot outliers than in the Normal plot. This would be hard to recognize in a plot of real data.

11.3.3.1 Mapping characteristic departures on a Q-Q plot to specific distributions

1. Continuous response variables of length, area, weight, or duration will often look like samples from a continuous probability distribution that is right-skewed, such as the **lognormal** or **gamma** distributions.
2. Count response variables will frequently look like samples from a discrete probability distribution that is right-skewed, such as the **poisson**, **quasi-poisson**, or **negative binomial** distributions.
3. Proportion (fraction of a whole) response variables will frequently look like samples from a continuous probability distribution bounded by 0 and 1, such as the **beta** distribution. Samples from a beta distribution can be left skewed, if the mean is near 1, right-skewed, if the mean is near zero, or symmetrical, if the mean is near 0.5.

11.3.3.2 Pump your intuition – confidence bands of a Q-Q plot

In introducing the confidence bands of the Q-Q plot above, I stated “A pattern of observed quantiles with some individual points outside of these boundaries indicates a sample that would be unusual if sampled from a Normal distribution.” Let’s use a parametric bootstrap to explore this.

1. Sample n values from a Normal distribution
2. Compute the sample quantiles by re-ordering the residuals of the sampled values from the sampled mean, from most negative to most positive.
3. Plot the quantiles against Normal quantiles for n points.
4. Repeat steps 1-3 n_iter times, superimposing the new sample quantiles over all previous sample quantiles. This creates a band of all sample quantiles over n_iter iterations of sampling n values from a Normal distribution.
5. At each value of the Normal quantile, compute the 95 percentile range of the sampled quantiles. Draw a ribbon inside these boundaries.

```

n_iter <- 1000
n <- 20
normal_qq <- ppoints(n) %>%
  qnorm()
sample_qq <- numeric(n_iter*n)
inc <- 1:n
for(iter in 1:n_iter){
  y <- rnorm(n)
  y_res <- y - mean(y)
  sample_qq[inc] <- y_res[order(y_res)]
  inc <- inc + n
}

qq_data <- data.table(normal_qq = normal_qq,
                      sample_qq = sample_qq)

qq_ci <- qq_data[, .(median = median(sample_qq),
                     lower = quantile(sample_qq, 0.025),
                     upper = quantile(sample_qq, 0.975)),
                  by = normal_qq]

ggplot(data = qq_data,
       aes(x = normal_qq,
           y = sample_qq)) +
  geom_point(alpha = 0.2) +
  geom_ribbon(data = qq_ci,
              aes(ymin = lower,

```

```

      ymax = upper,
      y = median,
      fill = "band"),
      fill = pal_okabe_ito[1],
      alpha = 0.3) +
xlab("Normal Quantile") +
ylab("Sample Quantile") +
theme_grid() +
NULL

```

11.3.4 Model checking homoskedasticity

11.4 Using R

Source: Wellenstein, M.D., Coffelt, S.B., Duits, D.E., van Miltenburg, M.H., Slagter, M., de Rink, I., Henneman, L., Kas, S.M., Prekovic, S., Hau, C.S. and Vrijland, K., 2019. Loss of p53 triggers WNT-dependent systemic inflammation to drive breast cancer metastasis. *Nature*, 572(7770), pp.538-542.

Public source

Data source

```

data_from <- "Loss of p53 triggers WNT-dependent systemic inflammation to drive breast cancer met
file_name <- "41586_2019_1450_MOESM3_ESM.xlsx"
file_path <- here(data_folder, data_from, file_name)

treatment_levels <- c("Trp53+/", "Trp53/-")
fig1f <- read_excel(file_path,
                     sheet = "Fig. 1f",
                     range = "A2:B23") %>%
  data.table() %>%
  melt(measure.vars = treatment_levels,
       variable.name = "treatment",
       value.name = "il1beta")
fig1f[, treatment := factor(treatment, treatment_levels)]

# be careful of the missing data. This can create mismatch between id and residual unless specific
# head(fig1f)

```

Fit a linear model

```
m1 <- lm(il1beta ~ treatment,
           data = fig1f)
```

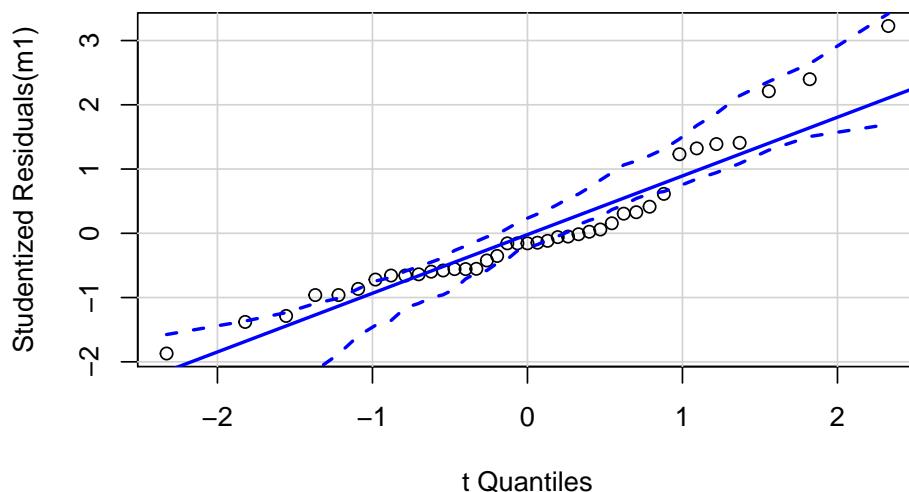
11.4.1 Normal Q-Q plots

`car::qqPlot` has several important arguments to control the type of Q-Q plot. The function uses base graphics instead of `ggplot2`. Typically, these plots would not be published other than possibly a supplement. Q-Q Plots and Worm Plots from `Scratch` is a good source of some of the arguments in `qqPlot`. Three important arguments are:

1. `simulate`. If passing a `lm` object, then the default confidence band is generated by a parametric bootstrap (`simulate = TRUE`). This band will differ somewhat each time you replot unless you set the seed with `set.seed`. Setting the argument `simulate = FALSE` returns the parametric band.
2. `line`. If passing a `lm` object, then the default line is a fit from a robust regression (`line = "robust"`). Setting the argument `line = "quartiles"` fits a line through the 25th and 75th percentile (or “quartiles”) quantiles.
3. `id`. The default identifies the index of the two points with the most extreme quartiles. Set to `FALSE` to hide.

The robust line is more sensitive to departures from Normality than the quartiles line.

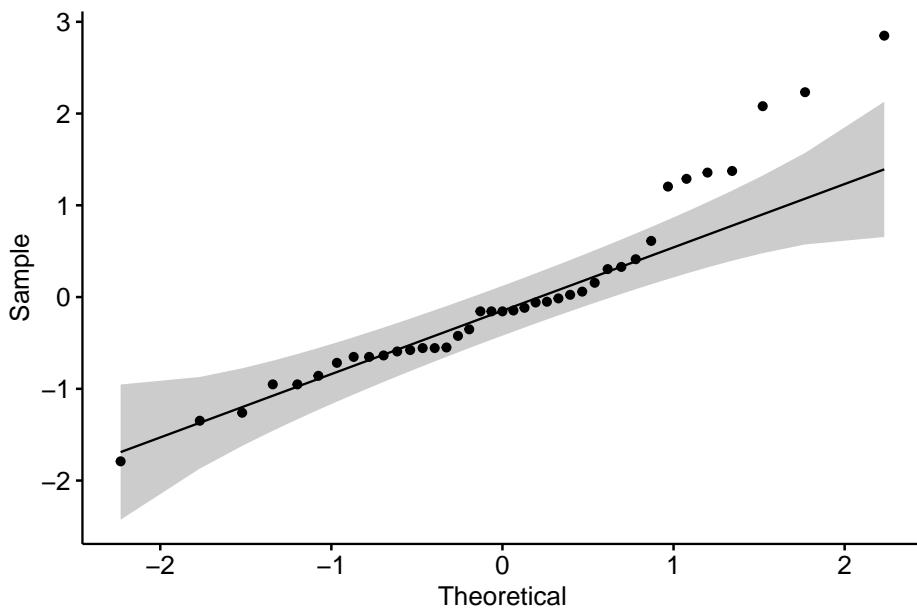
```
# defaults: robust line with bootstrap CI
set.seed(1)
qqPlot(m1, id = FALSE)
```



```
# classic: standard line with parametric CI
qqPlot(m1,
       line = "quartiles",
       simulate = FALSE,
       id = FALSE)
```

`ggpubr::ggqqplot` generates a pretty, `ggplot2` based Normal Q-Q plot, using the standard method for computing the line and confidence band.

```
m1_residuals <- data.table(m1_residuals = residuals(m1))
m1_residuals[, studentized := m1_residuals/sd(m1_residuals)]
ggqqplot(data = m1_residuals,
          x = "studentized")
```



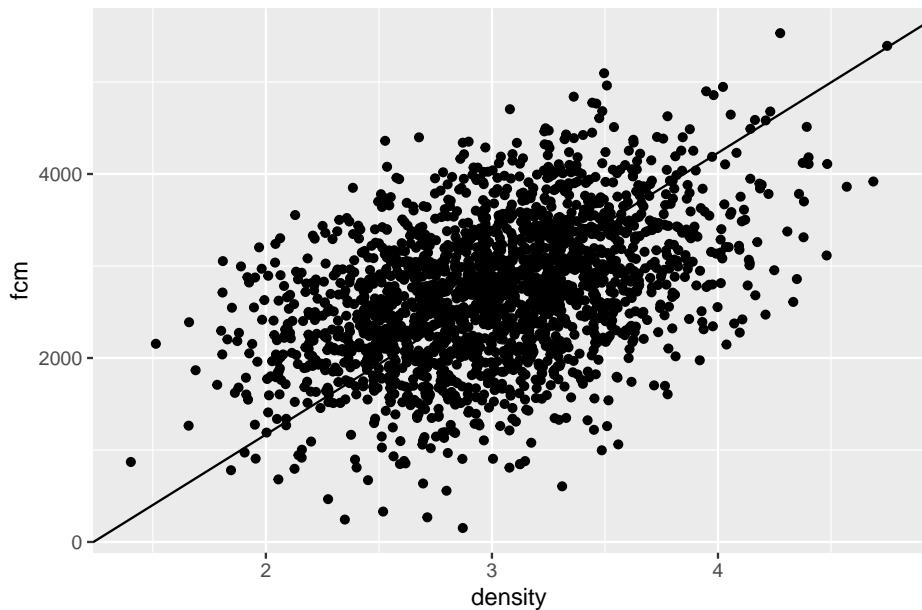
Chapter 12

Model Fitting and Model Fit (OLS)

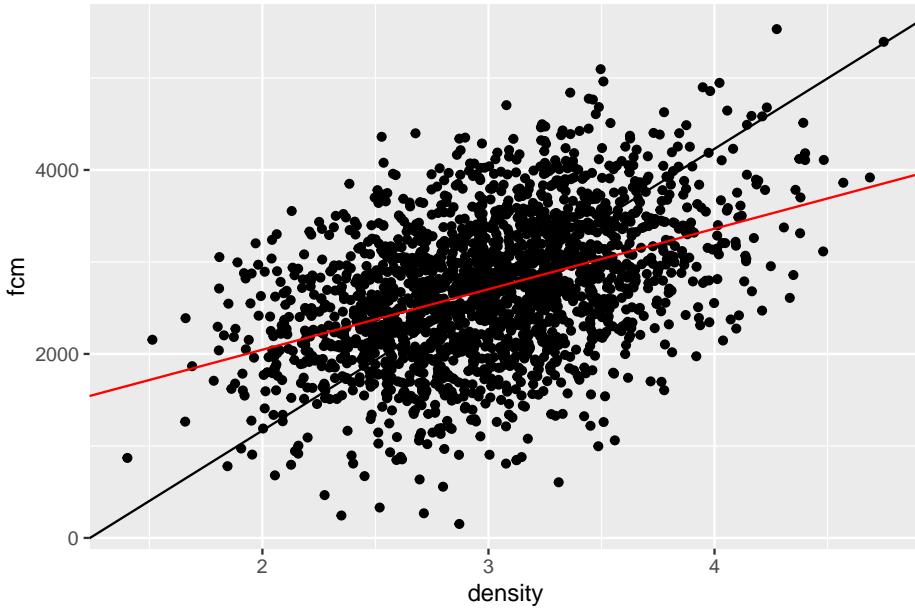
12.1 Least Squares Estimation and the Decomposition of Variance

The linear models in the last chapter and for much of this book are fit to data using a method called “ordinary least squares” (OLS). This chapter explores the meaning of OLS and related statistics, including R^2 , as well as some alternative methods for bivariate regression.

12.2 OLS regression



The fake data illustrated in the scatterplot above (Figure ??) were modeled to look something like the squirrel fecal cortisol metabolite data in the previous chapter. If a typical student is asked to draw a regression line through the scatter, they typically draw a line similar to that in Figure ?? . This line is not the OLS regression line but the major axis of an ellipse that encloses the scatter of points—that students invariably draw this line suggests that the brain interprets the major axis of an elliptical scatter of points as a trend (This major axis line is an alternative method for estimating a slope and is known as standard major-axis regression. More about this at the end of this chapter.)



The OLS regression line is the red line in Figure ?? – the standard major axis line is left for comparison). The OLS regression line

1. passes through the bivariate mean (\bar{x}, \bar{y}) of the scatter, and
2. minimizes the sum of the squared deviations from each point to its modeled value $\sum (y_i - \hat{y}_i)^2$

There are an infinite number of lines that pass through the bivariate mean (think of anchoring a line at the bivariate mean and spinning it). The OLS line is the line that minimizes the squared (vertical) deviations (“least squares”).

For a bivariate regression, the slope (coefficient b_1 of X) of the OLS model fit is computed by

$$b_1 = \frac{\text{COV}(X, Y)}{\text{VAR}(X)} \quad (12.1)$$

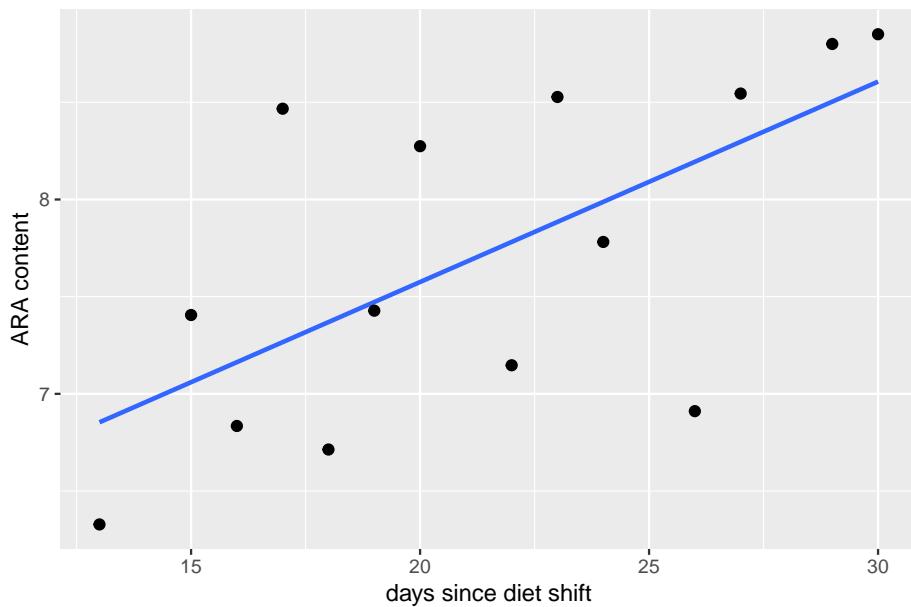
This equation is worth memorizing. We will generalize this into a more flexible equation in a few chapters.

12.3 How well does the model fit the data? R^2 and “variance explained”

Let’s switch to real data.

1. Source: Dryad Digital Repository. <https://doi.org/10.5061/dryad.056r5>
2. File: “Diet-shift data.xls”

Fish require arachidonic acid (ARA) and other highly unsaturated fatty acids in their diet and embryo and yolk-stage larvae obtain these from yolk. Fuiman and Faulk (xxx) designed an experiment to investigate if red drum (*Sciaenops ocellatus*) mothers provision the yolk with ARA from recent dietary intake or from stored sources in somatic tissues. The data below are from experiment 8. The x -axis is the days since a diet shift to more and less ARA (*days*) and the y -axis is the ARA content of the eggs (*ARA*).



The statistic R^2 is a measure of the fit of a model to data. The R^2 for the fit of the egg data is 0.42. R^2 is the fraction of two variances $\frac{\text{VAR}(\text{Model})}{\text{VAR}(Y)}$, or, the fraction of the variance of Y “explained by the model.” The value of R^2 ranges from zero (the fit cannot be any worse) to one (the fit is “perfect”).

To understand R^2 , and its computation, a bit more, let’s look at three kinds of deviations.

Figure 12.1A shows the deviations from the measured values to the mean value (dashed line). These are the deviations in the numerator of the equation to compute the variance of ARA_{EGG_M} . Figure 12.1B shows the deviations of the measured values from the modeled values. The sum of these deviations squared is what is minimized by the OLS fit. The bigger these deviations are, the worse the model fit. Figure 12.1C shows the deviations of the modeled values to the mean value. The bigger these deviations are, the better the model fit.

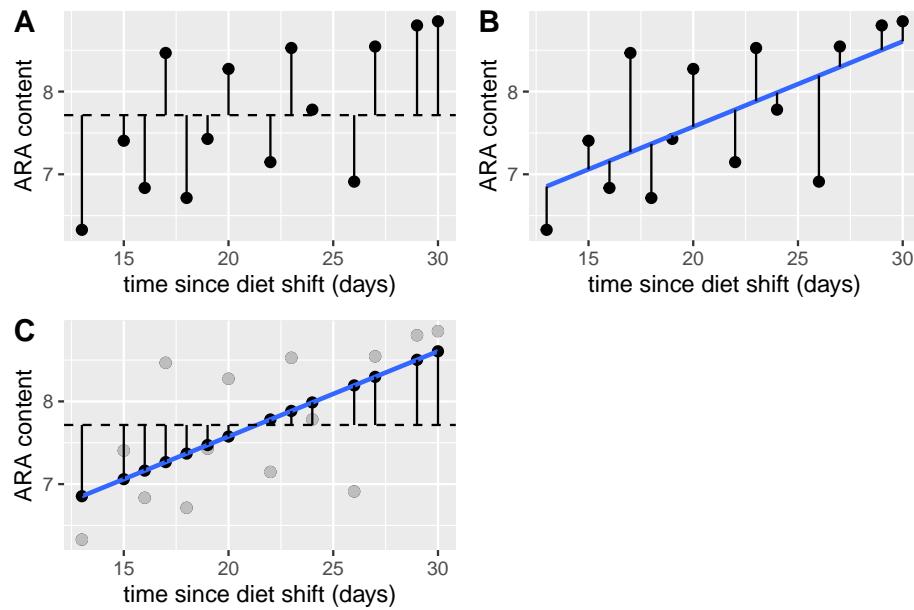


Figure 12.1: Three kinds of deviations from a fit model. A. Deviations of the measured values from the mean. These are in the numerator of the equation of the sample variance. The dashed line is the mean ARA content. B. Deviations of the measured values from the modeled values. The sum of these deviations squared is what is minimized in an OLS fit. C. Deviations of the modeled values from the mean ARA content. The measured values are in gray, the modeled values in black

The sums of the squares of these deviations (or “sums of squares”) have names:

$$\text{SS(total)} = \sum (y_i - \bar{y})^2 \quad (12.2)$$

$$\text{SS(error)} = \sum (y_i - \hat{y}_i)^2 \quad (12.3)$$

$$\text{SS(model)} = \sum (\hat{y}_i - \bar{y})^2 \quad (12.4)$$

Again, SS(total) is the numerator of the equation for the sample variance. It is called “s-s-total” because $\text{SS(total)} = \text{SS(model)} + \text{SS(error)}$. That is, the total sums of squares can be **decomposed** into two **components**: the modeled sums of squares and the error sums of squares. Given these components, it’s easy to understand R^2

$$R^2 = \frac{\text{SS(model)}}{\text{SS(total)}} \quad (12.5)$$

R^2 is the fraction of the total sums of squares that is due to (or “explained by”) the model sums of squares. Above I said that R^2 is the fraction of *variance* explained by the model. Equation xxx is a ratio of variance, but the $(n - 1)^{-1}$ in both the numerator and the denominator cancel out. Finally, many sources give the equation for R^2 as

$$R^2 = 1 - \frac{\text{SS(error)}}{\text{SS(total)}} \quad (12.6)$$

which is an obvious alternative given the decomposition. I prefer the former equation because it emphasizes the model fit instead of model ill-fit.

Chapter 13

Best practices – issues in inference

13.1 Multiple testing

Multiple testing is the practice of adjusting p -values (and less commonly confidence intervals) to account for the expected increase in the frequency of Type I error when there are multiple tests (typically Null Hypothesis Significance Tests). Multiple testing tends to arise in two types of situations:

1. Multiple pairwise contrasts among treatment levels (or combinations of levels) are estimated.
2. The effects of a treatment on multiple responses are estimated. This can arise if
 - a. there are multiple ways of measuring the consequences of something – for example, an injurious treatment on plant health might effect root biomass, shoot biomass, leaf number, leaf area, etc.
 - b. one is exploring the consequences of an effect on many, many outcomes – for example, the expression levels of 10,000 genes between normal and obese mice.

Despite the ubiquitous presence of multiple testing in elementary biostatistics textbooks, in the applied biology literature, and in journal guidelines, the practice of adjusting p -values for multiple tests is highly controversial among statisticians. My thoughts:

1. In situations like (1) above, I advocate that researchers **do not adjust p -values for multiple tests**. In general, its a best practice to only estimate

contrasts for which you care about because of some *a priori* model of how the system works. If you compare all pairwise contrasts of an experiment with many treatment levels and/or combinations, expect to find some false discoveries.

2. In situations like (2a) above, I advocate that researchers **do not adjust p-values for multiple tests**.
3. In situations like (2b) above, adjusting for the **False Discovery Rate** is an interesting approach. But, recognize that tests with small *p*-values are *highly provisional* discoveries of a patterns only and not a discovery of the causal sequelae of the treatment. For that, one needs to do the hard work of designing experiments that rigorously probe a working, mechanistic model of the system.

Finally, recognize that anytime there are multiple tests, Type M errors will arise due to the vagaries of sampling. This means that in a rank-ordered list of the effects, those at the top have measured effects that are probably bigger than the true effect. An alternative to adjusted *p*-values is a **penalized regression** model that shrinks effects toward the mean effect.

13.1.1 Some background

13.1.1.1 Family-wise error rate

The logic of multiple testing goes something like this: the more tests that a researcher does, the higher the probability that a false positive (Type I error) will occur, therefore a researcher should adjust *p*-values so that the Type I error over the set (or “family”) of tests is 5%. This adjusted Type I error rate is the “family-wise error rate”.

If a researcher carries out multiple tests *of data in which the null hypothesis is true*, what is the probability of finding at least one Type I error? This is easy to compute. If the frequency of Type I error for a single test is α , then the probability of no Type I error is $1 - \alpha$. For two tests, the probability of no Type I error in either test is the product of the probability for each test, or $(1 - \alpha)^2$. By the same logic, for m tests, the probability of no type I error in any of the tests is $(1 - \alpha)^m$. The probability of at least one type one error, across the m tests, then, is $1 - (1 - \alpha)^m$. A table of these probabilities for different m is given below. If the null is true in all tests, then at least one Type I error is more likely than not if there are 14 tests, and close to certain if there more than 50 tests. Don’t skip over this paragraph – the logic is important even if I don’t advocate adjusting for multiple tests.

Probability of at least one type I error within the set of multiple tests, for data in which the null hypothesis is true. The Type I error rate for a single test is 0.05. The number of tests is m . The probability is p .

```
m
p
1
0.05
3
0.14
6
0.26
10
0.40
50
0.92
100
0.99
```

13.1.1.2 False discovery rate

If a researcher carries out thousands of tests to “discover” new facts, and uses $p < 0.05$ as evidence of discovery, then what is the frequency of **false discoveries**?

13.1.1.3 p-value filter I – Inflated effects

If a researcher carries out many tests, and ranks the effects by magnitude or p -value, then the effect sizes of the largest effects will be inflated. Before explaining why, let’s simulate this using an experiment of allelopathic effects of the invasive garlic mustard (*Alliaria petiolata*) on gene expression in the native American ginseng (*Panax quinquefolius*). In the treated group, we have ten pots, each with an American ginseng plant grown in a container with a mustard plant. In the control group, we have ten pots, each with an American ginseng plant grown in a container with another American ginseng. I’ve simulated the response of 10,000 genes. The treatment has a true effect in 10% of the 10,000 genes but most effects are very small.

```
set.seed(4)
p <- 10^4 # number of genes
pt <- 0.1*p # number of genes with true response to treatment
n <- 10
```

```

# sample the gene effects from an exponential distribution
theta <- .3
beta <- c(rexp(pt, rate=1/theta),
          rep(0, (p-pt))) # the set of 10,000 effects

# sample the variance of the expression level with a gamma, and set a minimum
sigma <- rgamma(p, shape=2, scale=1/4) + 0.58
# quantile(sigma, c(0.001, 0.1, 0.5, 0.9, 0.999))

Y1 <- matrix(rnorm(n*p, mean=0, sd=rep(sigma, each=n)), nrow=n)
Y2 <- matrix(rnorm(n*p, mean=rep(beta, each=n), sd=rep(sigma, each=n)), nrow=n) # check
# use n <- 10^4 to check
# apply(y2, 2, mean)[1:5]
# b[1:5]
x <- rep(c("cn", "tr"), each=n)
bhat <- numeric(p)
p.value <- numeric(p)
sigma_hat <- numeric(p)
for(j in 1:p){
  fit <- lm(c(Y1[,j], Y2[, j]) ~ x)
  bhat[j] <- coef(summary(fit))["xtr", "Estimate"]
  p.value[j] <- coef(summary(fit))["xtr", "Pr(>|t|)"]
  sigma_hat[j] <- sqrt(sum(fit$residuals^2)/fit$df.residual)
}

```

The top 10 genes ranked by p-value. Rank is the rank of the true effect, from large to small.

effect
estimate
sigma
sd
p.value
relative true effect
rank
2.23
2.67
0.81
0.55
0.0000000

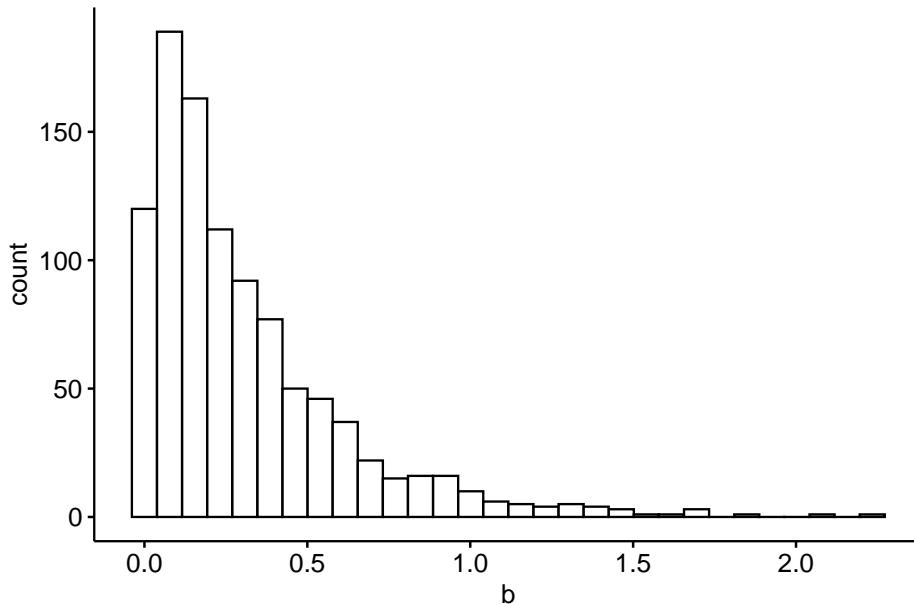


Figure 13.1: A histogram of the distribution of the 10,000 effects

1.00
1
1.59
1.92
0.62
0.57
0.0000005
0.71
7
1.46
1.86
0.84
0.71
0.0000159
0.65
10

1.47
1.78
0.83
0.74
0.0000409
0.66
9
0.00
1.95
1.10
0.85
0.0000717
0.00
0.48
1.26
0.67
0.56
0.0000816
0.21
212
0.97
1.32
0.69
0.60
0.0001004
0.43
45
0.54
1.68
1.06
0.78

0.0001321

0.24

173

0.00

2.05

1.39

0.96

0.0001488

0.00

0.43

-1.78

1.33

0.84

0.0001733

0.19

244

The table above lists the top 10 genes ranked by p -value, using the logic that the genes with the smallest p values are the genes that we should pursue with further experiments to understand the system. Some points

1. Six of the top ten genes with biggest true effects are *not* on this list. And, in the list are three genes with true effects that have relatively low ranks based on true effect size (column "rank") *and* two genes that have no true effect at all. Also in this list is one gene with an estimated effect (-1.78) that is *opposite* in sign of the true effect (but look at the p -value!).
2. The estimate of the effect size for all top-ten genes are inflated. The average estimate for these 10 genes is 1.47 while the average true effect for these 10 genes is 0.92 (the estimate).
3. The sample standard deviation (sd) for all top-ten genes is less than the true standard deviation (σ), in some cases substantially.

The consequence of an inflated estimate of the effect and a deflated estimate of the variance is a large t (not shown) and small p . What is going on is an individual gene's estimated effect and standard deviation are functions of 1) the true value and 2) a random sampling component. The random component will be symmetric, some effects will be overestimated and some underestimated. When we rank the genes by the estimate of the effect or t or p , some of the

genes that have “risen to the top” will be there because of a large, positive, sampling (random) component of the effect and/or a large, negative, sampling component of the variance. Thus some genes’ high rank is artificial in the sense that it is high because of a random fluke. If the experiment were re-done, these genes at the top because of a large, random component would (probably) fall back to a position closer to their expected rank (regression to the mean again).

In the example here, all genes at the top have inflated estimates of the effect because of the positive, random component. This inflation effect is a function of the signal to noise ratio, which is controlled by theta and sigma in the simulation. If theta is increased (try `theta=1`), or if sigma is decreased, the signal to noise ratio increases (try it and look at the histogram of the new distribution of effects) and both the 1) inflation and the 2) rise to the top phenomenon decrease.

13.1.1.4 p-hacking

13.1.2 Multiple testing – working in R

13.1.2.1 Tukey HSD adjustment of all pairwise comparisons

The `adjust` argument in `emmeans::contrast()` controls the method for *p*-value adjustment. The default is “tukey”.

1. “none” – no adjustment, in general my preference.
2. “tukey” – Tukey’s HSD, the default
3. “bonferroni” – the standard bonferroni, which is conservative
4. “fdr” – the false discovery rate
5. “mvt” – based on the multivariate *t* distribution and using covariance structure of the variables

The data are those from Fig. 2D of “Data from The enteric nervous system promotes intestinal health by constraining microbiota composition”. There is a single factor with four treatment levels. The response is neutrophil count.

No adjustment:

```
m1 <- lm(count ~ donor, data=exp2d)
m1.emm <- emmeans(m1, specs="donor")
m1.pairs.none <- contrast(m1.emm, method="revpairwise", adjust="none")
summary(m1.pairs.none, infer=c(TRUE, TRUE))

##   contrast      estimate    SE df lower.CL upper.CL t.ratio p.value
##   gf - wt       -1.502 1.48 58     -4.47     1.47 -1.013  0.3153
##   sox10 - wt      4.679 1.23 58      2.23     7.13  3.817  0.0003
##   sox10 - gf       6.182 1.45 58      3.29     9.08  4.276  0.0001
```

```

##   iap_mo - wt      -0.384 1.53 58     -3.45      2.68 -0.251  0.8025
##   iap_mo - gf      1.118 1.71 58     -2.31      4.54  0.654  0.5159
##   iap_mo - sox10   -5.064 1.49 58     -8.05     -2.07 -3.391  0.0013
##
## Confidence level used: 0.95

```

Tukey HSD:

```

m1.pairs.tukey <- contrast(m1.emm, method="revpairwise", adjust="tukey")
summary(m1.pairs.tukey, infer=c(TRUE, TRUE))

##   contrast      estimate    SE df lower.CL upper.CL t.ratio p.value
##   gf - wt       -1.502 1.48 58     -5.43      2.42 -1.013  0.7426
##   sox10 - wt    4.679 1.23 58      1.44      7.92  3.817  0.0018
##   sox10 - gf    6.182 1.45 58      2.36     10.01  4.276  0.0004
##   iap_mo - wt   -0.384 1.53 58     -4.43      3.66 -0.251  0.9944
##   iap_mo - gf   1.118 1.71 58     -3.41      5.64  0.654  0.9138
##   iap_mo - sox10 -5.064 1.49 58     -9.01     -1.11 -3.391  0.0067
##
## Confidence level used: 0.95
## Conf-level adjustment: tukey method for comparing a family of 4 estimates
## P value adjustment: tukey method for comparing a family of 4 estimates

```

13.1.3 False Discovery Rate

13.2 difference in p is not different

13.3 Inference when data are not Normal

No real data are normal, although many are pretty good approximations of a normal distribution.

I'll come back to this point, but first, let's back up. Inference in statistical models (standard errors, confidence intervals, *p*-values) are a function of the modeled distributions of the parameters (for linear models, this parameter is the conditional (or error) variance σ^2); if the data do not approximate the modeled distribution, then inferential statistics might be to liberal (standard errors are too small, confidence intervals are too narrow, Type I error is more than nominal) or to conservative (standard errors are too large, confidence intervals are too wide, Type I error is less than nominal).

Linear models assume that “the data” (specifically, the conditional response, or, equivalently, the residuals from the model) approximate a Normal distribution.

Chapter xxx showed how to qualitatively assess how well residuals approximate a Normal distribution using a Q-Q plot. If the researcher concludes that the data poorly approximate a normal distribution because of outliers, the researcher can use robust methods to estimate the parameters. If the approximation is poor because the residuals suggest a skewed distribution or one with heavy or light tails, the researcher can choose among several strategies

1. continue to use the linear model; inference can be fairly robust to non-normal data, especially when the sample size is not small.
2. use a generalized linear model (GLM), which is appropriate if the conditional response approximates any of the distributions that can be modeled using GLM (Chapter xxx)
3. use bootstrap for confidence intervals and permutation test for *p*-values
4. transform the data in a way that makes the conditional response more closely approximate a normal distribution.
5. use a classic non-parametric test, which are methods that do not assume a particular distribution

This list is roughly in the order of how I would advise researchers, although the order of 1-3 is pretty arbitrary. I would rarely advise a researcher to use (4) and never advise (5). Probably the most common strategies in the biology literature are (4) and (5). The first is also common but probably more from lack of recognition of issues or because a “test of normality” failed to reject that the data are “not normal”.

On this last point, do not use the *p*-value from a “test for normality” (such as a Shapiro-Wilk test) to decide between using the linear model (or t-test or ANOVA) and an alternative such as a generalized linear model (or transformation or non-parametric test). No real data is normal. Tests of normality will tend to “not reject” normality ($p > 0.05$) when the sample size is small and “reject” normality ($p < 0.05$) when the sample size is very large. But again, a “not rejected” hypothesis test does not mean the null (in this case, the data are normal) is true. More importantly, where the test for normality tends to fail to reject (encouraging a researcher to use parametric statistics) is where parametric inference performs the worst (because of small *n*) and where the test for normality tends to reject (encouraging a researcher to use non-parametric statistics) is where the parametric inference performs the best (because of large sample size) (Lumley xxx).

13.3.1 Working in R

The data for demonstrating different strategies are from Fig. 4A of “Data from The enteric nervous system promotes intestinal health by constraining microbiota composition”. There is a single factor with two treatment levels. The response is neutrophil count.

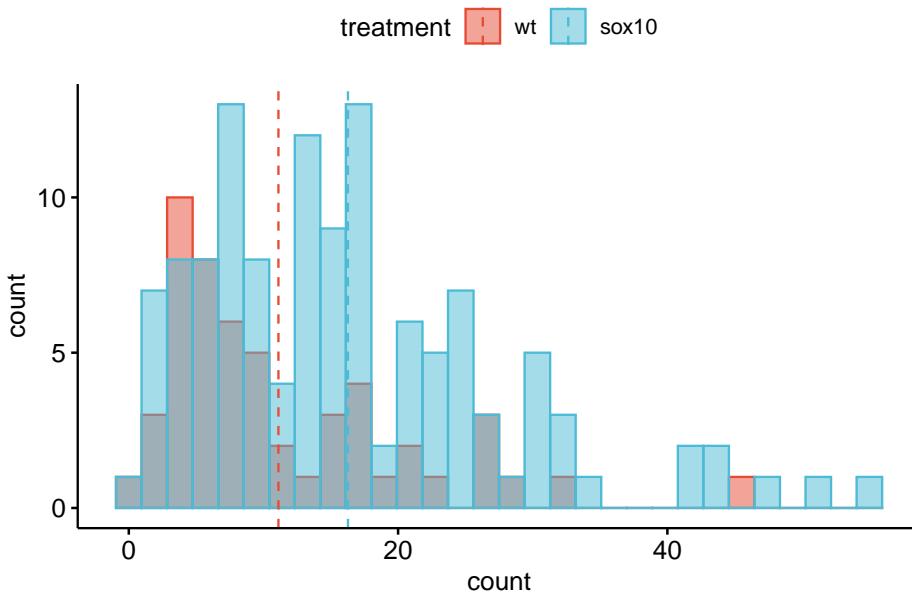


Figure 13.2: Distribution of the counts in the wildtype (WT) and sox10 knockout (sox10-) groups. Both groups show a strong right skew, which is common with count data.

A linear model to estimate the treatment effect and 95% confidence interval.

```
m1 <- lm(count ~ treatment, data=fig4a)
m1_emm <- emmeans(m1, specs="treatment")
summary(contrast(m1_emm, method="revpairwise"),
        infer=c(TRUE, TRUE))

## contrast   estimate    SE df lower.CL upper.CL t.ratio p.value
## sox10 - wt      5.16 1.75 174       1.7      8.62 2.947   0.0037
##
## Confidence level used: 0.95
```

13.3.2 Bootstrap Confidence Intervals

A bootstrap confidence interval is computed from the distribution of a statistic from many sets of re-sampled data. The basic algorithm is

1. compute the statistic for the observed data, assign this to θ_1
2. resample n rows of the data, with replacement. “with replacement” means to sample from the entire set of data and not the set that has yet to

be sampled. n is the original sample size; by resampling n rows with replacement, some rows will be sampled more than once, and some rows will not be sampled at all.

3. compute the statistic for the resampled data, assign these to $\theta_{2..m}$
4. repeat 2 and 3 $m - 1$ times
5. Given the distribution of m estimates, compute the lower interval as the $\frac{\alpha}{2}$ th percentile and the upper interval as the $1 - \frac{\alpha}{2}$ th percentile. For 95% confidence intervals, these are the 2.5th and 97.5th percentiles.

Let's apply this algorithm to the data from fig4A neutrophil count data in the coefficient table above. The focal statistic in these data is the difference in the mean count for the sox10 and wild type groups (the parameter for *treatment* in the linear model). The script below, which computes the 95% confidence intervals of this difference, resamples within **strata**, that is, within each group; it does this to preserve the original sample size within each group.

```

n_iter <- 5000
b1 <- numeric(5000)
inc <- 1:nrow(fig4a) # the rows for the first iteration are all rows, so this is the only one
for(i in 1:n_iter){
  # inc creates the index of rows to resample preserving the sample size specific to each treatment
  b1[i] <- coef(lm(count ~ treatment, data=fig4a[inc, ]))["treatmentsox10"]
  inc <- c(sample(which(fig4a[, treatment] == "wt"), replace=TRUE),
           sample(which(fig4a[, treatment] == "sox10"), replace=TRUE))
}
ci <- quantile(b1, c(0.025, 0.975))
c(contrast = b1[1], ci[1], ci[2])

## contrast      2.5%     97.5%
## 5.163215 2.077892 8.264316

```

The intervals calculated in step 5 are **percentile intervals**. A histogram of the the re-sampled differences helps to visualize the bootstrap (this is a pedagogical tool, not something you would want to publish).

13.3.2.1 Some R packages for bootstrap confidence intervals

Percentile intervals are known to be biased, meaning the intervals are shifted. The **boot** package computes a bias-corrected interval in addition to a percentile interval. **boot** is a very powerful bootstrap package but requires the researcher to write functions to compute the parameter of interest. **simpleboot** provides functions for common analysis that does this for you (in R speak, we say that **simpleboot** is a “wrapper” to **boot**). The function **simpleboot::two.boot** computes a **boot**-like object that returns, among other values, the distribution

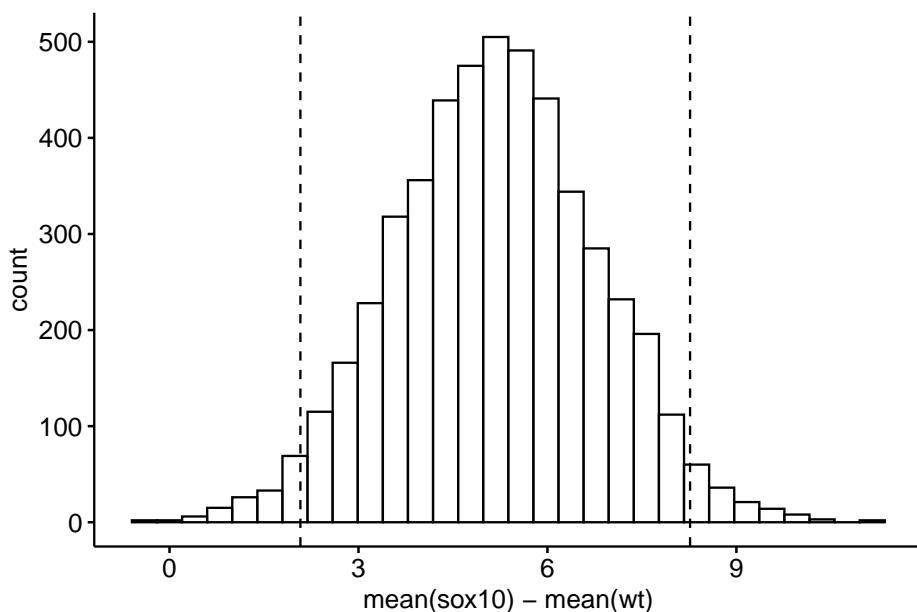


Figure 13.3: Distribution of the 5000 resampled estimates of the difference in means between the sox10 and wt treatment levels. The dashed lines are located at the 2.5th and 97.5th percentiles of the distribution.

of m statistics. The `simpleboot` object is then fed to `boot::boot.ci` to get bias-corrected intervals.

```
bs_diff <- two.boot(fig4a[treatment=="sox10", count],
                     fig4a[treatment=="wt", count],
                     mean,
                     R=5000)
boot.ci(bs_diff, type="bca")

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 5000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = bs_diff, type = "bca")
##
## Intervals :
## Level      BCa
## 95%   ( 2.087,  8.410 )
## Calculations and Intervals on Original Scale
```

13.3.3 Permutation test

A permutation test effectively computes the probability that a random assignment of a response to a particular value of X generates a test statistic as large or larger than the observed statistic. If this probability is small, then this “random assignment” is unlikely. From this we infer that the actual assignment matters, which implies a treatment effect.

The basic algorithm is

1. compute the test statistic for the observed data, assign this to θ_1
2. permute the response
3. compute the test statistic for the permuted data, assign these to $\theta_{2..m}$
4. repeat 2 and 3 $m - 1$ times
5. compute p as

$$p_{perm} = \frac{N_{\theta_i \geq \theta_1}}{m} \quad (13.1)$$

This is easily done with a `for` loop in which the observed statistic is the first value in the vector of statistics. If this is done, the minimum value in the numerator for the computation of p_{perm} is 1, which insures that p_{perm} is not zero.

The test statistic depends on the analysis. For the simple comparison of means, a simple test statistic is the difference in means. This is the numerator of the test statistic in a t -test. The test has more power if the test-statistic is scaled (Manley xxx), so a better test statistic would be t , which scales the difference by its standard error.

Here, I implement this algorithm. The test is two-tailed, so the absolute difference is recorded. The first value computed is the observed absolute difference.

```
set.seed(1)
n_permutations <- 5000
d <- numeric(n_permutations)

# create a new column which will contain the permuted response
# for the first iteration, this will be the observed order
fig4a[, count_perm := count]

for(i in 1:n_permutations){
  d[i] <- abs(t.test(count_perm ~ treatment, data = fig4a)$statistic)

  # permute the count_perm column for the next iteration
  fig4a[, count_perm := sample(count)]
}
p <- sum(d >= d[1])/n_permutations
p

## [1] 0.002
```

13.3.3.1 Some R packages with permutation tests.

`lmPerm::lmp` generates permutation p-values for parameters of any kind of linear model. The test statistic is the sum of squares of the term scaled by the residual sum of squares of the model.

```
set.seed(2)
coef(summary(lmp(count ~ treatment, perm="Prob", Ca=0.01,
                  data=fig4a)))

## [1] "Settings: unique SS"

##           Estimate Iter Pr(Prob)
## (Intercept) 13.694815 5000  0.0042
## treatment1 -2.581608 5000  0.0042
```

13.3.4 Non-parametric tests

1. In general, the role of a non-parametric test is a better-behaved p -value, that is, one whose Type I error is well controlled. As such, non-parametric tests are more about Null-Hypothesis Statistical Testing and less (or not at all) about Estimation.
2. In general, classic non-parametric tests are only available for fairly simple experimental designs. Classic non-parametric tests include
 - Independent sample (Student's) t test: Mann-Whitney-Wilcoxon
 - Paired t test: Wilcoxon signed-rank test

One rarely sees non-parametric tests for more complex designs that include covariates, or multiple factors, but for these, one could 1) convert the response to ranks and fit the usual linear model, or 2) implement a permutation test that properly preserves **exchangeability**.

Permutation tests control Type I error and are powerful. That said, I would recommend a permutation test as a supplement to, and not replacement of, inference from a generalized linear model.

A non-parametric (Mann-Whitney-Wilcoxon) test of the fake data generated above

```
wilcox.test(count ~ treatment, data=fig4a)

##
##  Wilcoxon rank sum test with continuity correction
##
## data: count by treatment
## W = 2275, p-value = 0.001495
## alternative hypothesis: true location shift is not equal to 0
```

13.3.5 Log transformations

Many response variables within biology, including count data, and almost anything that grows, are right skewed and have variances that increase with the mean. A log transform of a response variable with this kind of distribution will tend to make the residuals more approximately normal and the variance less dependent of the mean. At least two issues arise

1. if the response is count data, and the data include counts of zero, then a fudge factor has to be added to the response since $\log(0)$ doesn't exist. The typical fudge factor is to add 1 to *all* values, but this is arbitrary and results do depend on the magnitude of this fudge factor.

2. the estimates are on a log scale and do not have the units of the response. The estimates can be back-transformed by taking the exponent of a coefficient or contrast but this itself produces problems. For example, the backtransformed mean of the log-transformed response is not the mean on the original scale (the arithmetic mean) but the **geometric mean**. Geometric means are smaller than arithmetic means, appreciably so if the data are heavily skewed. Do we want our understanding of a system to be based on geometric means?

13.3.5.1 Working in R – log transformations

If we fit a linear model to a log-transformed response then the resulting coefficients and predictions are on the **log scale**. To make interpretation of the analyses easier, we probably want to **back-transform** the coefficients or the predictions to the original scale of the response, which is called the **response scale**.

```
m2 <- lm(log(count + 1) ~ treatment, data=fig4a)
(m2_emm <- emmeans(m2,
                     specs="treatment",
                     type = "response"))

##  treatment response    SE df lower.CL upper.CL
##  wt          8.22 0.965 174     6.5    10.3
##  sox10      12.59 0.934 174    10.9    14.6
##
## Confidence level used: 0.95
## Intervals are back-transformed from the log(mu + 1) scale
```

The emmeans package is amazing. Using the argument `type = "response"` not only backtransforms the means to the response scale but also subtracts the 1 that was added to all values in the model.

What about the effect of treatment on count?

```
summary(contrast(m2_emm,
                  method="revpairwise",
                  type = "response"),
                  infer=c(TRUE, TRUE))

##  contrast   ratio    SE df lower.CL upper.CL t.ratio p.value
##  sox10 / wt  1.47 0.185 174     1.15    1.89 3.100  0.0023
##
## Confidence level used: 0.95
## Intervals are back-transformed from the log scale
## Tests are performed on the log scale
```

It isn't necessary to backtransform the estimated marginal means prior to computing the contrasts as this can be done in the contrast function itself. Here, the `type = "response"` argument in the contrast function is redundant since this was done in the computation of the means. But it is transparent so I want it there.

Don't skip this paragraph Look at the value in the “contrast” column – it is “`sox10 / wt`” and not “`sox10 - wt`”. The backtransformed effect is a ratio instead of a difference. **A difference on the log scale is a ratio on the response scale** because of this equality

$$\exp(\mu_2 - \mu_1) = \frac{\exp(\mu_2)}{\exp(\mu_1)} \quad (13.2)$$

The interpretation is: If b^* is the backtransformed effect, then, given a one unit increase in X , the expected value of the response increases $b^* \times$. For a categorical X , this means the backtransformed effect is the ratio of backtransformed means – its what you have to multiply the mean of the reference by to get the mean of the treated group. And, because it is the response that is log-transformed, these means are not arithmetic means but geometric means. Here, this is complicated by the model – the response is not a simple log transformation but $\log(response + 1)$. It is easy enough to get the geometric mean of the treated group – multiply the backtransformed intercept by the backtransformed coefficient and then subtract 1 – but because of this subtraction of 1, the interpretation of the backtransformed effect is awkward at best (recall that I told you that a linear model of a log transformed response, and especially the log of the response plus one, leads to difficulty in interpreting the effects).

```
# backtransformed control mean -- a geometric mean
mu_1 <- exp(coef(m2)[1])

# backtransformed effect
b1_star <- exp(coef(m2)[2])

# product minus 1
mu_1*b1_star -1

## (Intercept)
##      12.59357

# geometric mean of treatment group
n <- length(fig4a[treatment=="sox10", count])
exp(mean(log(fig4a[treatment=="sox10", count+1])))-1

## [1] 12.59357
```

Back-transformed effect

```
m1 <- lm(count ~ treatment, data=fig4a)

exp(coef(m2))

##      (Intercept) treatmentsox10
##         9.219770      1.474394
```

13.3.6 Performance of parametric tests and alternatives

13.3.6.1 Type I error

If we are going to compute a p -value, we want it to be uniformly distributed “under the null”. A simple way to check this is to compute Type I error. If we set $\alpha = 0.05$, then we’d expect 5% of tests of an experiment with no effect to have $p < 0.05$.

```
# first create a matrix with a bunch of data sets, each in its own column
n <- 10
n_sets <- 4000
fake_matrix <- rbind(matrix(rnegbin(n*n_sets, mu=10, theta=1), nrow=n),
                      matrix(rnegbin(n*n_sets, mu=10, theta=1), nrow=n))
treatment <- rep(c("cn", "tr"), each=n)

tests <- c("lm", "log_lm", "mww", "perm")
res_matrix <- matrix(NA, nrow=n_sets, ncol=length(tests))
colnames(res_matrix) <- tests
for(j in 1:n_sets){
  res_matrix[j, "lm"] <- coef(summary(lm(fake_matrix[,j] ~ treatment
                                         )))[2, "Pr(>|t|)"]
  res_matrix[j, "log_lm"] <- coef(summary(lm(log(fake_matrix[,j] + 1) ~ treatment
                                             )))[2, "Pr(>|t|)"]
  res_matrix[j, "mww"] <- wilcox.test(fake_matrix[,j] ~ treatment,
                                         exact=FALSE)$p.value
  res_matrix[j, "perm"] <- coef(summary(lmp(fake_matrix[,j] ~ treatment,
                                             perm="Prob", Ca=0.01)))[2, "Pr(Prob)"]
}

apply(res_matrix, 2, function(x) sum(x < 0.05)/n_sets)

##      lm log_lm      mww      perm
## 0.04150 0.05250 0.04350 0.04675
```

Type I error is computed for the linear model, the linear model with a log transformed response, Mann-Whitney-Wilcoxon, and permutation tests. All four tests are slightly conservative for data that look like that modeled. The computed Type I error of the permutation test is closest to the nominal value of 0.05.

13.3.6.2 Power

Power is the probability of a test to reject the null hypothesis if the null hypothesis is false (that is, if an effect exists)

$$\text{Power} = \text{Prob}(p < \alpha | \text{mathrm}{effect} \neq 0) \quad (13.3)$$

If all we care about is a *p-value* then we want to use a test that is most powerful. But, while power is defined using α , we *can* care about power even if we don't consider α to be a very useful concept because increased power also increases the precision of an estimate (that is, narrows confidence intervals).

```
# first create a matrix with a bunch of data sets, each in its own column
n <- 5
n_sets <- 4000
fake_matrix <- rbind(matrix(rnegbin(n*n_sets, mu=10, theta=1), nrow=n),
                      matrix(rnegbin(n*n_sets, mu=20, theta=1), nrow=n))
treatment <- rep(c("cn", "tr"), each=n)

tests <- c("lm", "log_lm", "mww", "perm")
res_matrix <- matrix(NA, nrow=n_sets, ncol=length(tests))
colnames(res_matrix) <- tests
for(j in 1:n_sets){
  res_matrix[j, "lm"] <- coef(summary(lm(fake_matrix[,j] ~ treatment
                                         )))[2, "Pr(>|t|)"]
  res_matrix[j, "log_lm"] <- coef(summary(lm(log(fake_matrix[,j] + 1) ~ treatment
                                         )))[2, "Pr(>|t|)"]
  res_matrix[j, "mww"] <- wilcox.test(fake_matrix[,j] ~ treatment,
                                         exact=FALSE)$p.value
  res_matrix[j, "perm"] <- coef(summary(lmp(fake_matrix[,j] ~ treatment,
                                         perm="Prob", Ca=0.01)))[2, "Pr(Prob)"]
}

apply(res_matrix, 2, function(x) sum(x < 0.05)/n_sets)

##      lm  log_lm      mww      perm
## 0.09200 0.12525 0.08375 0.10600
```

As above, Power is computed for the linear model, linear model with a log-transformed response, Mann-Whitney-Wilcoxon, and permutation, by simulating a “low power” experiment. The effect is huge (twice as many cells) but the power is low because the sample size is small ($n = 5$). At this sample size, and for this model of fake data, all tests have low power. The power of the log-transformed response is the largest. A problem is, this is not a test of the means but of the log transformed mean plus 1. The power of the permutation test is about 25% larger than that of the linear model and Mann-Whitney-Wilcoxon test. An advantage of this test is that it is a p-value of the mean. A good complement to this p-value would be bootstrapped confidence intervals. Repeat this simulation using $n = 40$ do see how the relative power among the three change in a simulation of an experiment with more power.

Part VI: More than one X

– Multivariable Models

Chapter 14

Adding covariates to a linear model

In its most general sense, **Covariates** are simply the X variables in a statistical model. With data from experiments, “covariates” more typically refers to X variables that are added to a model to increase precision of the treatment effects. In observational designs, covariates might be added to a model to 1) increase predictive ability, 2) because the researcher is interested in specific conditional effects, or 3) to eliminate confounding. These are discussed in later chapters.

14.1 Adding covariates can increases the precision of the effect of interest

I use fake data to introduce the concept of **statistical elimination** of a **covariate** in a statistical model. Here I am modeling the effect of a new drug on blood LDL-C levels. LDL is a kind of lipoprotein, which are particles in the blood that transport fats and cholesterol to and from different tissues. LDL-C is cholesterol associated with LDL particles. LDL-C is considered “bad cholesterol” because LDL is believed to transport cholesterol and other lipids to arterial walls, which is the basis for atherosclerosis.

Twenty applied biostats students are recruited and are randomly assigned to either the “placebo” treatment level or “drug” treatment level. The response is blood LDL-C concentration. The drug manufacturer wants a measure of the effect of the new drug on ldlc.

The plot below shows the LDL-C response in the placebo and drug groups, including the group means and 95% confidence intervals.

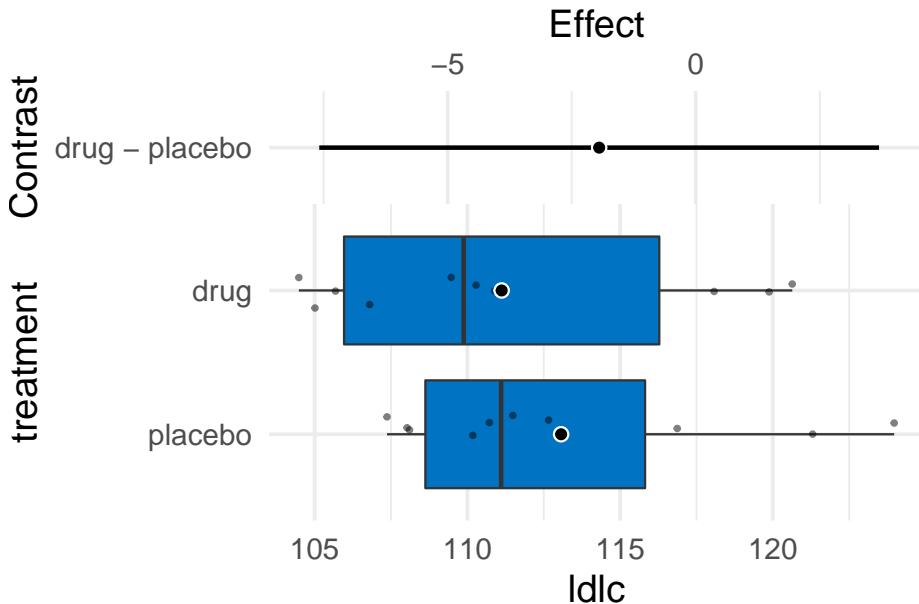


Figure 14.1: The fake LDL-C experiment.

$$ldlc = \beta_0 + \beta_1 treatment + \varepsilon \quad (14.1)$$

where *treatment* is the dummy variable with *placebo* = 0 and *drug* = 1.

The coefficient table is

```
##           Estimate Std. Error t value Pr(>|t|)
## (Intercept) 113.069   1.899  59.548  0.000
## treatmentdrug -1.947    2.685  -0.725  0.478
```

The plot shows large overlap in LDL-C. There “is no effect of the drug ($p = .478$)” is an incorrect interpretation of the hypothesis test of the estimate of β_1 . A correct interpretation is, the estimated effect is -1.9 but everything from large, negative effects to moderate positive effects are consistent with the data.

LDL-C is strongly correlated with age and there is a large range in age among the Applied Bistats students. Consequently, age will contribute to a large fraction of the variance in LDL-C. If so, this age-related variance *might* be masking the effect of the drug. Here is a plot of LDL-C vs. age, with treatment assignment color coded. Remember, these are the exact same values of LDL-C as in figure 14.1 above.

The line is the bivariate regression fit to the data ignoring treatment level.

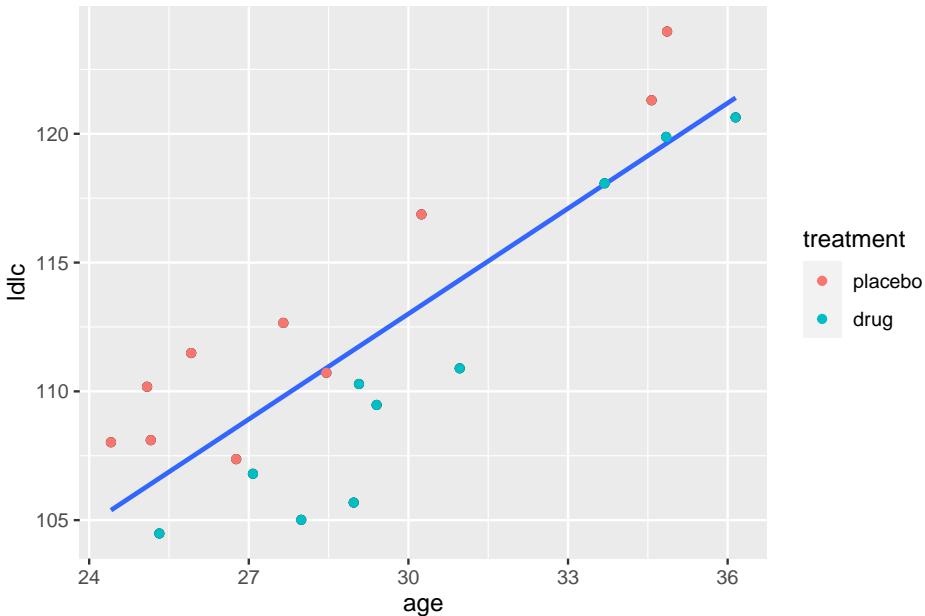


Figure 14.2: Linear regression of $ldlc$ on dietary fat fit to the fake LDL-C data. The points are color coded by treatment.

$$ldlc = \beta_0 + \beta_1 age + \varepsilon \quad (14.2)$$

While the points are color-coded by treatment level, $treatment$ is not in model (14.2). The color-coding makes it clear that most of the “placebo” data points are above the line, or have positive residuals from the model, while the “drug” data points are below the line, or have negative residuals from the model. A better way to think about this pattern is that **at any specific level of age, the LDL-C for drug is lower than the LDL-C for placebo**.

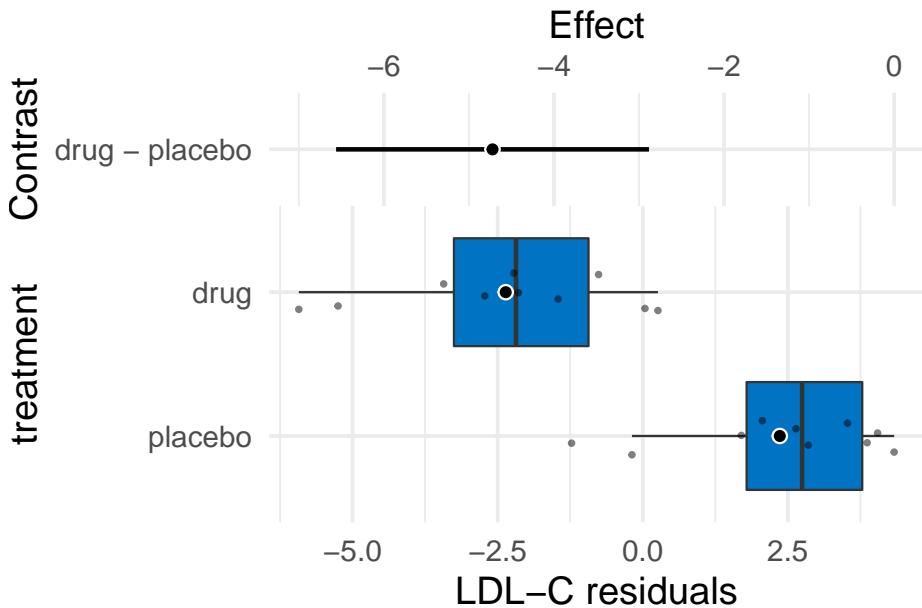
What is happening? Age is contributing to the variance of LDL-C, and the noise in ε in model (14.1), and this added noise makes it harder to measure the effect of the new drug relative to placebo. Age is masking the effect. If we could somehow measure the effect of the drug at a specific age, then we could get a more precise estimate of the effect. But how to do this? Here are three possible methods. The third is *the only* one you should use but the second is useful for understanding the third.

1. We could just analyze a subset of the data, that is, only the cases in which the value of age is nearly equal. This throws away perfectly good data and, consequently, greatly reduces the sample size and thus precision to estimate the effect.

2. We could use the residuals of the fitted model (??) to estimate the effect of drug treatment (this is what we did by eye in figure 14.2). Here is the new model

$$ldlc.r = \beta_0 + \beta_1 treatment + \varepsilon \quad (14.3)$$

where $ldlc.r$ is the set of residuals.

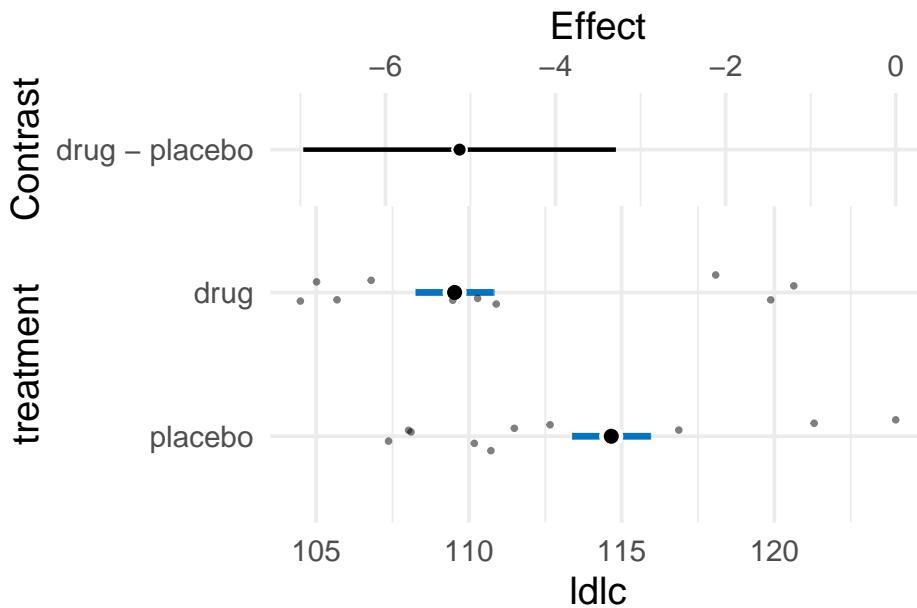


Now the estimate of the effect is -4.7 mg/dL blood and the SE is only 0.88. In this two-stage analysis (stage 1: fit $ldlc \sim age$ to get residuals, stage 2: fit residuals $\sim treatment$), we have *eliminated the effect of age* on the variance of the response and, as a consequence, the estimate of the effect of the drug is much more precise – the effect of *treatment* has a smaller standard error.

3. A better method for this two-stage procedure that increases the precision of the estimate of the treatment effect by eliminating variance of a covariate (*age*) is to simply add the covariate to the original linear model.

$$ldlc = \beta_0 + \beta_1 age + \beta_2 treatment + \varepsilon \quad (14.4)$$

which results in the Harrell Plot



and the coefficient table

Coefficients of the model that includes the covariate age.

Estimate

Std. Error

t value

Pr(>|t|)

(Intercept)

68.8

3.46

19.9

0.0e+00

age

1.6

0.12

13.0

0.0e+00

treatmentdrug

-5.1

0.87

-5.9

1.8e-05

In the linear model that includes the covariate *age* (model (14.4)), the SE of the treatment effect is 0.87. Compare this to SE of the treatment effect in the model without the covariate (model (14.1)), which is 3.1X larger.

14.2 Adding covariates can decrease prediction error in predictive models

14.3 Adding covariates can reduce bias due to confounding in explanatory models

14.4 Best practices 1: A pre-treatment measure of the response should be a covariate and not subtracted from the post-treatment measure (regression to the mean)

It is common to measure the outcome variable (Y) both before and after the experimental treatments are applied and then compare the pre-post *change* in Y in response to the treatment using a *t*-test or ANOVA using this linear model

$$Y_{post} - Y_{pre} = \beta_0 + \beta_1 Treatment + \varepsilon \quad (14.5)$$

Don't do this. Instead, add the pre-treatment measure into the model as a covariate.

$$Y_{post} = \beta_0 + \beta_1 Y_{pre} + \beta_2 Treatment + \varepsilon \quad (14.6)$$

where *Treatment* is a dummy variable for a two-level factor. A pre-treatment measure (Y_{pre}) is often called the *baseline* measure. The change in Y ($\Delta Y = Y_{post} - Y_{pre}$) is sometimes called a change score or gain score. If you really want to estimate the treatment effect on the change from pre-treatment value to post-treatment value, then use model (14.6) with ΔY as the response – the *p*-value will be precisely the same (the estimate and SE will differ of course because the response variable is different).

The reason why a researcher should not model a change score (ΔY) as a function of *Treatment* without Y_{pre} as a covariate is a phenomenon called **regression**

14.4. BEST PRACTICES 1: A PRE-TREATMENT MEASURE OF THE RESPONSE SHOULD BE A COVARIATE

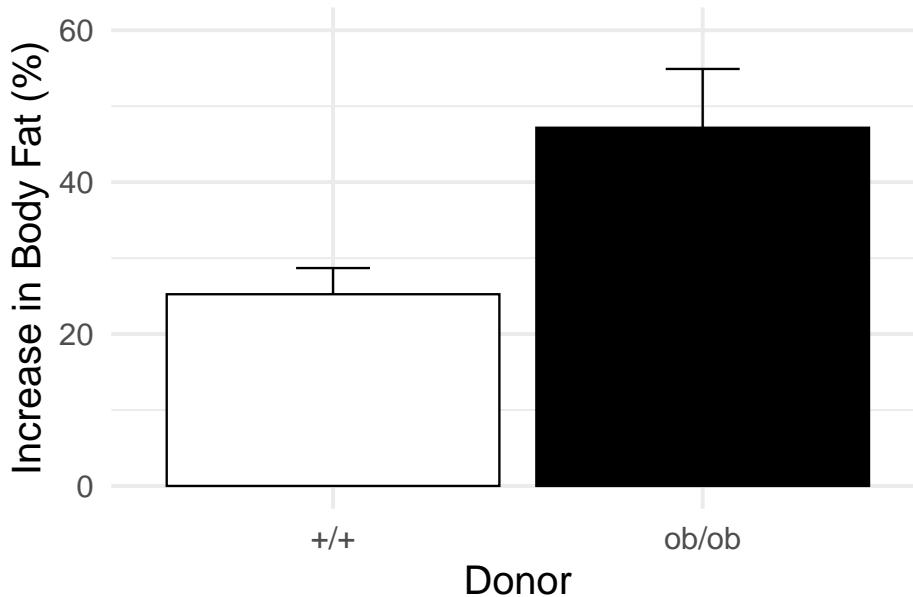


Figure 14.3: Figure 3c of Turnbaugh *et al* 2006. This figure was generated with simulated data matching the summary statistics given in Turnbaugh *et al* 2006

to the mean. To explain regression to the mean, I use fake data simulated to model the results from an important study on gut microbiomes. In this study, the authors (Turnbaugh et al. xxx) showed that mice with feces from obese (genotype *ob/ob*) donors had higher weight gain than mice with feces from lean (genotype *+/+*) donors, presumably because of the differences in microbial communities between the donor types (shown elsewhere in their paper). To support the inference of a large difference in weight change, they illustrated the percent change in each treatment level in their Fig 3C, which is replicated here using simulated data generated to match the original summary statistics (Figure 14.3).

That looks like a big difference, with the mice from the obese-donor treatment level gaining much more fat than the mice from the lean-donor treatment level. Turnbaugh et al. used a simple t-test of this percent change to test the effect of the *ob/ob* treatment. The linear model underneath this *t*-test is

$$\text{percent_change_fat} = \beta_0 + \beta_1 \text{obese} + \varepsilon \quad (14.7)$$

where *percent_change_fat* is the percent change in fat from baseline and *obese* is a dummy variable with *ob/ob* = 1. The percent change in fat is $\frac{\text{fat}_{\text{post}} - \text{fat}_{\text{pre}}}{\text{fat}_{\text{pre}}} \times 100$, so is a function of the change score $\Delta_{\text{fat}} = \text{fat}_{\text{post}} - \text{fat}_{\text{pre}}$.

The model coefficients are

```

##           Estimate Std. Error t value Pr(>|t|)
## (Intercept) 25.24015  5.627515 4.485134 0.0003259533
## treatmentob/ob 21.92156  8.176589 2.681016 0.0157879742

##            2.5 %   97.5 %
## (Intercept) 13.367137 37.11317
## treatmentob/ob 4.670468 39.17266

```

Or, the increase in fat in the obese-treated mice was 21.9% (95%CI: 4.7, 39.2%, $p = 0.016$) greater than the increase in lean-treated mice. This result, if generally verified with replication and rigorous probing, would have spectacular implications for human health.

14.4.1 Regression to the mean in words

Regression to the mean is the phenomenon that if an extreme value is sampled, the next sample will likely be less extreme. This makes sense, if you randomly sample a single human male and that individual is 6'10" (about 4 standard deviations above the mean), the next human you randomly sample will almost certainly be closer to the mean human male. Or, if you randomly sample five human males and the mean height in the group is 5'1" (about 3 standard deviations below the mean), the next sample of five human males that you measure will almost certainly be closer to the mean human male.

How does regression to the mean apply to the analysis of change scores in a pre-post experiment, like the mouse fecal transplant study? In a pre-post experiment, subjects are randomized to treatment group. The response is measured at baseline and again at the conclusion of the experiment. Despite random treatment assignment, the mean fat weight of the *ob/ob* group at baseline was 1.2 standard deviations smaller than that of the *+/+* group. If there is no treatment effect, what is the expected difference at the end?

To answer this, we need to know how an individual's fat weight at the end is related to its fat weight at baseline. An individual's final fat is dependent on its initial fat if factors that contribute to the measurement of fat are the same at baseline and the end. For example, if an individual has relatively high metabolism both at baseline and at the end, then that individual might have relatively low fat at baseline and at the end. This dependence of final value on baseline value is quantified by the correlation between the two measures. This correlation is ρ (the greek letter rho). Factors that change over the duration of the experiment, including random measurement error, cause the correlation to be less than one. The two extremes of this correlation, and the expected difference in fat weight at the end are:

1. $\rho = 0$ – if an individual's final fat is independent of its initial fat then we expect the difference at end to be zero.

14.4. BEST PRACTICES 1: A PRE-TREATMENT MEASURE OF THE RESPONSE SHOULD BE A COVARIATE

2. $\rho = 1$ – if an individual's final fat is entirely dependent on its initial fat, then we'd expect the mean fat weight of the *ob/ob* group to be 1.2 standard deviations smaller than that of the *+/+* group, exactly as it was at baseline.

Regression to the mean happens when $\rho < 1$ and its consequences increase as ρ goes to zero. What is meant by “consequences”?

The fat weight of the *ob/ob* group at baseline is 1.2 standard deviations smaller than that of the *+/+* group. If $\rho = 0$, then we'd expect the difference between mean fat weight at the end of the experiment to be zero. *Given the starting differences in mean weight*, to get to zero difference at the end, the *ob/ob* mice would have to gain more fat weight than the *+/+* mice. Since the expectation of the mean difference at the end is zero the expectation of the change score *must be bigger for the ob/ob mice than for the +/+ mice*. That is the expectation of the *difference* in change score is conditional on (or “a function of”) the difference in fat weight at baseline.

14.4.2 Regression to the mean in pictures

Let's simulate this to pump our intuition about regression to the mean and its consequences on pre-post experiments.

1. randomly sample a normal distribution as the “initial weight” and randomly assign to treatment class
2. let the final weight have some correlation (ρ) with the initial weight. Some correlation should make sense – we expect a mouse that has more fat than average at the start of the experiment to also have more fat than average at the end of the experiment. Run the experiment at different values of this correlation to see how it effects regression to the mean.
3. Do not add a treatment effect. We want to explore the behavior of the null hypothesis.

What's happening in Figure 14.4? Each point is a result for a single, simulated experiment. In total, there are 1000 simulated experiments for each of four values of ρ . The *x*-axis is the difference between the means of the two treatment levels at baseline (*Initial difference*). The *y*-axis is the difference in mean change score between the two treatment levels – that is the difference in the means of ΔY from equation (??). This difference in ΔY is the effect of the treatment the researchers are interested in. The *unconditional* expectation of this difference is zero

$$E(\Delta Y_{ob/ob} - \Delta Y_{+/+}) = 0 \quad (14.8)$$

but the change conditional on baseline is not zero

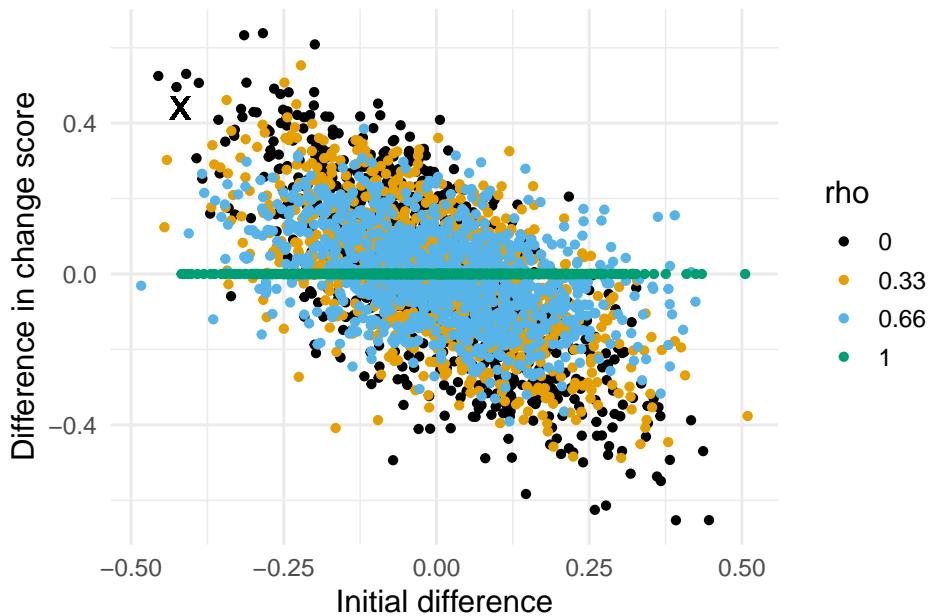


Figure 14.4: Effect of initial difference in weight on the difference in change score. Increased initial difference in weight results in an increased differences in change score between treatment and control. Four different values of ρ (the correlation between initial and final weights) were simulated. Only when $\rho=1$ is there no influence of initial difference, because whatever differences occur at baseline will be perfectly preserved in the final measure. The X gives the values in the original Turnbaugh data

14.4. BEST PRACTICES 1: A PRE-TREATMENT MEASURE OF THE RESPONSE SHOULD BE A COVARIATE

$$E(\Delta Y_{ob/ob} - \Delta Y_{+/+}) \neq 0 \quad (14.9)$$

Instead, the conditional expectation is a function of the difference at baseline. If the initial difference in weight happens to be unusually large and negative, the expected difference in change score is unusually positive. This non-zero expectation means that the estimate of the treatment effect is **conditionally biased** for any model that does not include the baseline fat weight as a covariate. And, from a frequentist perspective, the Type I error for a test of a difference in ΔY is strongly dependent on the initial difference in weight.

The big X in the plot indicates the difference at baseline and difference in ΔY for the original fecal transplant study. The difference in ΔY is unusually positive (about .6% of the $|\delta Y|$ are larger) but very close to the expected value given the unusually large, negative difference at baseline. In other words, the probability of the data, or more extreme than the data, is not 0.006 but something larger and perhaps, much larger (the computed value depends on the observed ρ . From, the plot, the X is very unusual if $\rho = 1$, pretty unusual if $\rho = 0.66$, but pretty common if $\rho = 0.33$ or if $\rho = 0$).

14.4.3 Do not use percent change, believing that percents account for effects of initial weights

Some researchers mistakenly believe that a t -test of percent change automatically adjusts for effects in initial weight, since this initial weight is in the denominator of the percent. This is wrong. The dependency of the difference in change between treatments on the initial difference between treatments is more severe if change is measured as a percent, because the numerator (the change score) is expected to be larger if the denominator is smaller (initial measure). Using the simulated data from above, here is this dependency.

14.4.4 Do not “test for balance” of baseline measures

A test of the null hypothesis of no difference in mean at baseline is a “test for balance.” Researchers frequently test for balance at baseline and use the p -value of the test to decide the next step: 1) if $p > 0.05$, conclude that the pre-treatment means “do not differ” and use something like a simple t test of the post-treatment means, 2) if $p < 0.05$, then use the change score, or the percent change, as the response in a simple t -test, or 3) if $p < 0.05$, then use a linear model with the pre-treatment value as a covariate. Here, and in general, hypothesis tests used to decide which of several ways to proceed do not make sense. First, a null-hypothesis significance test cannot tell you that there is “no difference” – this is not what null-hypothesis tests do. Second, any p -value after the initial test isn’t strictly valid as it does not take into account this decision

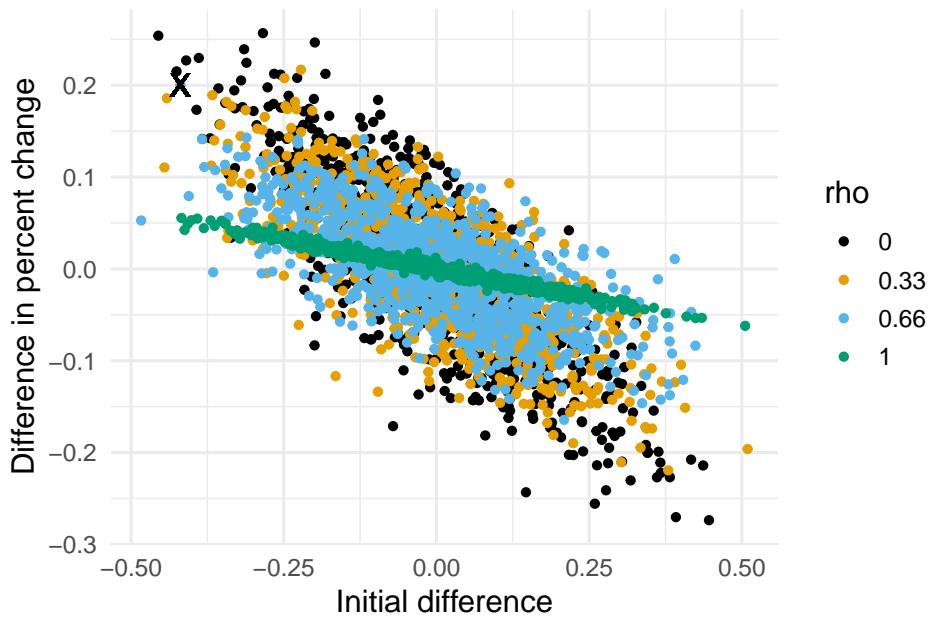


Figure 14.5: Effect of initial difference in weight on the difference in percent change. Increased initial difference in weight results in an increased differences in Percent change between treatment and control. Four different values of ρ (the correlation between initial and final weights) were simulated. Note there is no value of ρ where the difference in percent change is independent of the initial difference. The X gives the values in the original Turnbaugh data.

step, but this is minor. Third, **it doesn't matter**; there will always be some difference in the actual means of the initial measures and, consequently, the conditional expectation of the final measures, or change in measures, or percent change will be dependent on this initial difference. So, if one has initial measures, one should use an linear model that adjusts for baseline measures to estimate the treatment effect in pre-post designs. And, if one isn't planning on taking an initial measure, then maybe you should, because the initial measure used in a linear model allows a better estimate of the treatment effect, as discussed above in Adding covariates can increases the precision of the effect of interest.

14.5 Best practices 2: Use a covariate instead of normalizing a response

Chapter 15

Two (or more) Categorical X – Factorial designs

** “ASK1 inhibits browning of white adipose tissue in obesity” ** assess interaction by comparing simple effects and concluded incorrectly.

15.1 Factorial experiments

A factorial experiment is one in which there are two or more categorical X that are **crossed**, resulting in a group for all combinations of the levels of each factor. Factorial experiments are used to estimate the **interaction** between factors, which occurs when the effect of the level of one factor depends on the levels of the other factors. For example, a researcher wants to estimate the effect of an environmental toxin on basal metabolic rate (BMR) in a fish and designs an experiment with two factors: *Treatment* with levels “control” and “toxin” and *Sex*, with levels “male” and “female”. If the magnitude (and possibly sign) of the effect of the toxin on BMR differs between males and females, there is an interaction between *Treatment* and *Sex*. Interactions are usually denoted with a \times symbol: *Treatment* \times *Sex*. Interactions are ubiquitous, although sometimes they are small enough to ignore with little to no loss of understanding.

This chapter uses data from an experiment measuring the effect of *Temp* and *CO₂* on larval sea urchin metabolic rate (*Resp*) (there are other outcome measures in the study too). The units of metabolic rate are pmol O₂/hr/larva. There are two *Temp* levels (13C and 18C) and two *CO₂* levels (400 μ Atm and 1100 μ Atm) and the factors are fully crossed, which makes this a 2×2 (crossed or factorial) design. There are $n = 6$ replicates for each combination of the levels. A good way to visualize the treatment combinations in a crossed design

is with a $m \times p$ table showing all combinations of the m levels of factor 1 ($Temp$) against the p levels of factor 2 (CO_2)

		CO ₂	
		400 μatm	1100μATM
T		13C	n=6
		18C	n=6

The upper left cell represents the combination of 13 C and 400 μ Atm level within the CO₂ factor. The replicates in this cell were grown with no added treatments, so this cell is the “control” for Temp and the control for CO₂, which we will use as the “reference” group for the linear model. The replicates in the lower left cell were grown with an added temperature treatment (in this case, a 5 C higher temperature). The replicates in the upper right cell were grown with an added CO₂ treatment (700 μ ATM higher CO₂). And finally, the replicates in the bottom right cell were grown with both the added temperature (+5 C) and added CO₂ (+700 μ ATM). Here, I use a “+” or “-” to designate the addition (or not) of the treatment, so our 2×2 treatment levels are Temp-/CO₂-, Temp+/CO₂-, Temp-/CO₂+ and Temp+/CO₂+

15.1.1 Model coefficients: an interaction effect is what is leftover after adding the treatment effects to the control

A factorial design allows a researcher to estimate the interaction between two factors. To clarify this, let's fit the factorial model and look at the coefficient table. The systematic component of the factorial model is

$$Resp = \beta_0 + \beta_1 Temp^+ + \beta_2 CO_2^+ + \beta_3 Temp^+ CO_2^+ \quad (15.1)$$

Again, $Temp^+$ and CO_2^+ are dummy variables. The model also includes $Temp^+ CO_2^+$, which is a dummy variable for the interaction between Temp and CO₂. The value of this interaction dummy variable is literally the product of the two main factor dummy variables ($Temp^+$ and CO_2^+), which can be verified with the model matrix (which here, is computed from the subset of the data that includeds only the first two rows of each treatment combination)

(Intercept)

Temp+

CO2+

Temp+:CO2+

1

0

0

0

1

0

0

0

1

1

0

0

1

1

0

0

1

0

1

0

1

0

1

0

1

1

1

1

1

370CHAPTER 15. TWO (OR MORE) CATEGORICAL X – FACTORIAL DESIGNS

1	
1	
1	
	The coefficient table is
	Coefficient table of the factorial model
	Estimate
	Std. Error
	t value
	Pr($> t $)
	(Intercept)
	8.23
	0.73
	11.3
	0.000
	Temp+
	4.51
	1.03
	4.4
	0.000
	CO2+
	-0.32
	1.03
	-0.3
	0.761
	Temp+:CO2+
	-2.68
	1.45
	-1.9
	0.079

1. The Intercept (b_0) is the mean (8.23) of the reference (Temp-/CO2-) group, and so the mean of the upper left cell in Table 1).

2. The Temp+ coefficient (b_1) is the estimate of the added temperature effect relative to the reference, and so is the mean of the lower left cell minus the mean of the upper left cell ($b_1 = \bar{Y}_{Temp^+} - \bar{Y}_{Temp^-/CO2^-}$). Another way of stating this is, it is the effect of Temp when CO2 is at its reference level.
3. The CO2+ coefficient (b_2) is the estimate of the added CO2 effect relative to the reference, and so is the mean of the upper right cell minus the mean of the upper left cell ($b_2 = \bar{Y}_{CO2^+} - \bar{Y}_{Temp^-/CO2^-}$). Another way of stating this is, it is the effect of CO2 when Temp is at its reference level.
4. The Temp+:CO2+ coefficient (b_3) is the estimate of the **interaction effect**, which is the effect in addition to the Temp+ and CO2+ effects. If you added b_1 and b_2 to b_0 , you would get the mean of the Temp+/CO2+ group *if the effects were purely additive*. So the interaction effect is the difference between the mean of the bottom right cell and the sum of the coefficients of the other three cells ($b_3 = \bar{Y}_{Temp^+CO2^+} - (b_0 + b_1 + b_2)$). An interaction is a **non-additive effect**. Think about this. Adding 5 C increases respiration by 4.51 units. Adding 700 μ ATM CO2 decreases respiration by .32 units. If these effects were purely additive, then adding both 5 C and 700 μ ATM should result in a mean of $8.23 + 4.51 - .32 = 12.42$ units for the Temp+/CO2+ group. What is the mean of this group?

9.74! So the difference between the “additive expectation” and the actual mean is $9.74 - 12.42 = -2.68$, which is the interaction effect (coefficient). A graphical interpretation of these coefficients are in the figure of treatment means below (figure ??)

15.1.2 What is the biological meaning of an interaction effect?

I can dead lift 150 pounds and my friend Jake can deadlift 175 pounds. Working together, we should be able to lift 325 pounds. What if together, we could actually lift 400 pounds? If this were the case, this would be an interaction with an effect equal to 75 pounds. Is this biologically plausible? If so, what is the mechanism? Here is a possible mechanism (although I am highly skeptical of it having a magnitude of 75 pounds): when lifting an object as part of a group, the central nervous system allows increased motor unit recruitment, and so each person can lift more weight than they could if lifting alone. A positive interaction like this is called *synergistic*. Always think about the biological meaning of an interaction effect.

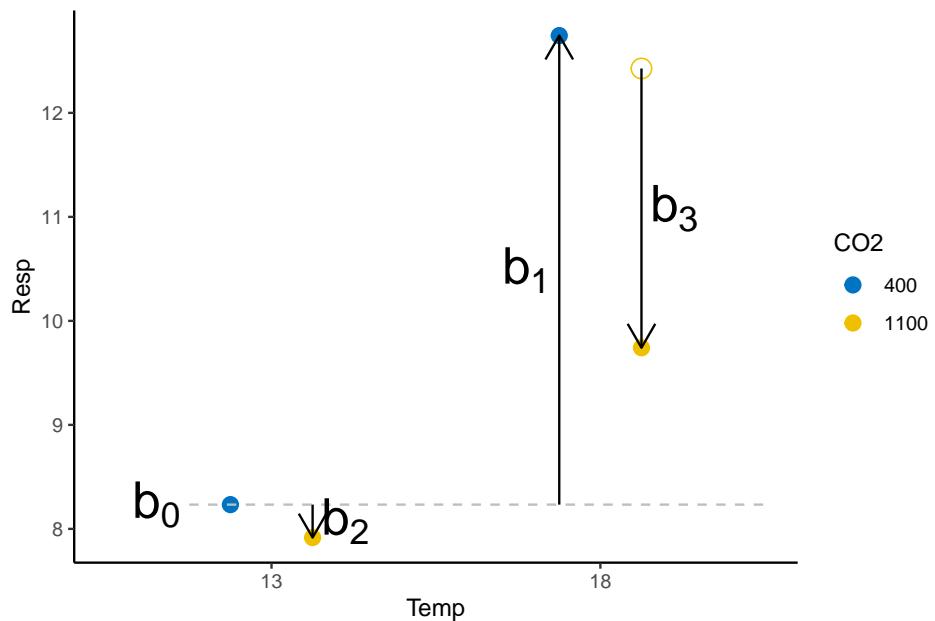


Figure 15.1: Meaning of coefficients in factorial model. b_0 (dashed line) is the mean of the reference. b_1 (length of vector b_1) is the mean of the Temp treatment minus the mean of the reference. b_2 (length of vector b_2) is the mean of the CO₂ treatment minus the mean of the reference. b_3 (length of vector b_3) is the mean of the Temp + CO₂ treatment minus what this value would be if there were no interaction (indicated by the open gold circle)

15.1.3 The interpretation of the coefficients in a factorial model is entirely dependent on the reference...

at least using dummy coding of the factor variables, which is the default in R. To see this, here is the coefficient table of the model but assigning Temp+/CO2+ as the reference (by re-ordering levels in both factors)

Estimate

Std. Error

t value

$\text{Pr}(>|t|)$

(Intercept)

9.74

0.73

13.4

0.000

Temp-

-1.82

1.03

-1.8

0.091

CO2-

3.00

1.03

2.9

0.008

Temp-:CO2-

-2.68

1.45

-1.9

0.079

This dependence of the coefficients on the reference is a feature not a bug. It is what we mean when we pose the questions “Compared to larvae raised at today’s temperature, what is the effect of adding 5° Temp on larval respiration?”,

“Compared to larvae raised at today’s CO₂, what is the effect of adding 700 ppm CO₂ on larval respiration?”, and “Compared to larvae raised at today’s temperature and CO₂, what is the effect of adding 5° Temp and 700 µAtm CO₂ on larval respiration?” If we change the reference, we are asking different questions.

15.1.4 Estimated marginal means

The modeled means (or predicted values) of the factorial model (Model (15.1)) fit to the urchin data are shown in the table below. The values in the last column and row are the **marginal means**, which are the means of the associated row or column. More generally, *marginal* refers to a statistic averaged across multiple levels of another variable

Marginal means from the full factorial model

Temp	
400 µAtm	
1100 µAtm	
mean	
13 C	
8.2333	
7.9167	
8.0750	
18 C	
12.7433	
9.7417	
11.2425	
mean	
10.4883	
8.8292	

The marginal means with their CIs are

Temp	
emmmean	
SE	
df	

lower.CL

upper.CL

13

8.0750

0.5130468

20

7.004803

9.145197

18

11.2425

0.5130468

20

10.172303

12.312697

CO2

emmmean

SE

df

lower.CL

upper.CL

400

10.488333

0.5130468

20

9.418136

11.558530

1100

8.829167

0.5130468

20

7.758970

9.899363

15.1.5 In a factorial model, there are multiple effects of each factor (simple effects)

With a single factor, there was a single effect for each non-reference level of the factor. For example, if the levels are “control”, “knockout”, and “rescue”, the knockout effect is the contrast between knockout and control and the rescue effect is the contrast between rescue and control. In a factorial experiment with crossed A and B factors, there are multiple effects of a non-reference level of factor A – one for each level of factor B. For the urchin experiment, there is an effect of the 18 C level of Temp when CO₂ is 400 µAtm and an effect when CO₂ is 1100 µAtm. Similarly, there is an effect of the 1100 level of CO₂ when Temp is 13 C and when Temp is 18 C. These effects, or **contrasts** (differences in modeled means), are sometimes called the **simple effects**. Another name could be the “conditional” effects, since the value of the effect is conditional on the level factor B.

One way to visualize the simple effects is by using the 2×2 table of treatment combinations. The contrasts in the right-side column are the simple effects of CO₂ at each level of Temp. The contrasts in the bottom row are the simple effects of Temp at each level of CO₂. Note that the first simple effect for each factor has a corresponding row in the table of coefficients of the fit model above.

Conditional (simple) effects of full factorial model fit to urchin data

Temp

400 µAtm

1100 µAtm

simple

13 C

8.2333

7.9167

-0.3167

18 C

12.7433

9.7417

-3.0017

simple

4.5100

1.8250

The 95% confidence intervals and *p*-values of the simple effects of the factorial model (Model (15.1)) are given in the table below.

CO2

Temp

Contrast

Estimate

Lower CI

Upper CI

t

p

400

.

18 - 13

4.5100

2.3696

6.6504

4.3953

0.0003

1100

.

18 - 13

1.8250

-0.3154

3.9654

1.7786

0.0905

.

13

1100 - 400

-0.3167

-2.4571

378 CHAPTER 15. TWO (OR MORE) CATEGORICAL X – FACTORIAL DESIGNS

1.8237
-0.3086
0.7608
. .
18
1100 - 400
-3.0017
-5.1421
-0.8613
-2.9253
0.0084

The first line is the effect of the 18 C level of Temp when CO₂ is 400 μ Atm.
The 3rd line is the effect of the 1100 μ Atm level of CO₂ when Temp is 13 C.

15.1.6 Marginal effects

The average of the simple effects for a factor are the **marginal effects**, or the **main effects** in ANOVA terminology.

Temp
400 μ Atm
1100 μ Atm
simple
marginal
13 C
8.2333
7.9167
-0.3167
18 C
12.7433
9.7417
-3.0017
simple

4.5100

1.8250

3.1675

marginal

-1.6592

The 95% confidence interval and *p*-value of these marginal effects are

Contrast

Estimate

Lower CI

Upper CI

t

p

18 - 13

3.1675

1.6540

4.6810

4.3656

0.0003

1100 - 400

-1.6592

-3.1727

-0.1457

-2.2867

0.0332

Marginal effects can be useful for summarizing a general trend, but, like any average, might not be especially meaningful if there is large heterogeneity of the simple effects, which occurs when the interaction effect is large. The urchin example is a good example of marginal effects that would be highly misleading to present without further comment.

15.1.7 The additive model

If an interaction effect is small, then it can be useful to estimate the effects of the two factors as if the interaction were equal to zero.

$$Resp = \beta_0 + \beta_1 Temp^+ + \beta_2 CO2^+ \quad (15.2)$$

This is a **reduced model** because one of the terms has been removed from the model. This particular reduced model is often referred to as the **additive model**, since it excludes the interaction term, which is a *product* of other terms. The model coefficients of the additive model are given in the table below.

Estimate

Std. Error

t value

$Pr(>|t|)$

(Intercept)

8.90

0.66

13.4

0.000

Temp+

3.17

0.77

4.1

0.000

CO2+

-1.66

0.77

-2.2

0.042

The conditional effects of the reduced model are

CO2

Temp

Contrast

Estimate

Lower CI

Upper CI

t

p

400

.

18 - 13

3.1675

1.5739

4.7611

4.1336

0.0005

1100

.

18 - 13

3.1675

1.5739

4.7611

4.1336

0.0005

.

13

1100 - 400

-1.6592

-3.2527

-0.0656

-2.1652

0.0420

.

18

382CHAPTER 15. TWO (OR MORE) CATEGORICAL X – FACTORIAL DESIGNS

1100 - 400

-1.6592

-3.2527

-0.0656

-2.1652

0.0420

The table shows that all conditional effects within a factor are the same. This makes sense – if the model fit is additive, the interaction effect is set to zero and, consequently there cannot be differences in conditional effects. Probably a better way of thinking about this is, it doesn't make sense to compute or discuss conditional effects in an additive model. Instead, an additive model automatically computes marginal effects.

Contrast

Estimate

Lower CI

Upper CI

t

p

18 - 13

3.1675

1.5739

4.7611

4.1336

0.0005

1100 - 400

-1.6592

-3.2527

-0.0656

-2.1652

0.0420

Compare the table of marginal effects of the additive model to the table of marginal effects of the full model. The estimates are the same but the t -values and p -values differ because of different degrees of freedom (the full model estimates one more parameter, the interaction effect). The estimate is the same

only if the design is balanced, which means that each combination of treatment levels has the same sample size n .

15.1.8 Reduce models for the right reason

Unless one factor truly has no effect, there will always be an interaction. As stated above, interactions are ubiquitous. If an interaction is small, it can make sense to drop the interaction term and re-fit an additive model to estimate marginal effects in order to present a simplified picture of what is going on, with the recognition that these estimates are smoothing over the heterogeneity in conditional (simple) effects that truly exist.

Aided and abetted by statistics textbooks for biologists, there is a long history of researchers dropping an interaction effect because the interaction $p > 0.05$. Don't do this. It doesn't make any sense.

1. The p -value is an arbitrary dichotomization of a continuous variable. Would it make sense to behave differently if the interaction were $p = 0.051$ vs. $p = 0.049$, given that these two p -values are effectively identical?
2. A p -value is not evidence that an effect is zero, or “doesn't exist”, or even that an effect is “trivially small”. This is because p -values are a function of measurement error, sampling error, and sample size, in addition to effect size.

15.1.9 What about models with more than two factors?

A factorial model can have more than two factors, for example, a model with three factors (A, B, and C), each with two levels (which I'll designate with a “+”), is

$$Y = \beta_0 + \beta_1 A^+ + \beta_2 B^+ + \beta_3 C^+ + \beta_4 A^+ B^+ + \beta_5 A^+ C^+ + \beta_6 B^+ C^+ + \beta_7 A^+ B^+ C^+ + \varepsilon \quad (15.3)$$

It is easy enough to get an ANOVA table with p -values for this model but I don't recommend it because

1. If space and/or time and/or materials are limited then it typically makes more sense to prioritize the power to estimate standard errors by choosing one of the two-factor models and increasing sample size
2. Interaction effects in 2-factor models are hard enough to interpret. A 3-way interaction is very, very tough to interpret. If all we did was table up F -ratios and p -values, this wouldn't matter. But it does matter.

15.2 Reporting results

15.2.1 Text results

The effect of the increased temperature at the control CO₂ level was 4.5 pmol O₂/hr/larva (95% CI: 2.4, 6.7; $p < 0.001$). The effect of increased CO₂ at the control temperature was -0.3 pmol O₂/hr/larva (95% CI: -2.4, 1.8; $p = .76$). The interaction effect was -2.7 pmol O₂/hr/larva (95% CI: -5.7, 0.3; $p = 0.079$). Because of the relatively large interaction, the effect of temperature at the high level of CO₂ was less than half the effect at the low level of CO₂ (estimate: 1.82; 95% CI: -0.3, 4.0; $p = 0.091$) and the effect of CO₂ at the high level of Temp was 10 times greater than that at the low level of Temp (estimate: -3.0; 95% CI: -5.1, -9; $p = 0.0084$).

The CI on the interaction includes both large negative values and trivially small values, including zero, and, consequently, our data is compatible with both scientific models (that is, we can neither support nor reject the predictions of the scientific model using these results).

15.3 Working in R

15.3.1 Model formula

A full-factorial model with two factors is specified in the model formula as $y \sim A*B$ where A is the first factor, and B is the second factor. The * indicates to cross A and B. R expands this formula to $y \sim 1 + A + B + A:B$ where the colon indicates an interaction (multiplicative) effect.

```
m1 <- lm(Resp ~ Temp*CO2, data=urchin) # use urchin1 data with relabeled levels
coef(summary(m1))
```

	Estimate	Std. Error	t value	Pr(> t)
## (Intercept)	8.2333333	0.7255577	11.3475922	3.626935e-10
## Temp18	4.5100000	1.0260936	4.3953106	2.792573e-04
## CO21100	-0.3166667	1.0260936	-0.3086138	7.608069e-01
## Temp18:CO21100	-2.6850000	1.4511155	-1.8503007	7.910035e-02

The additive model is specified by the formula $y \sim A + B$

```
m2 <- lm(Resp ~ Temp + CO2, data=urchin) # use urchin1 data with relabeled levels
coef(summary(m2))
```

```
##              Estimate Std. Error   t value   Pr(>|t|) 
## (Intercept) 8.904583  0.6636207 13.418183 9.038657e-12
## Temp18      3.167500  0.7662831  4.133590 4.721000e-04
## CO21100    -1.659167  0.7662831 -2.165214 4.203445e-02
```

15.3.2 Modeled means

Modeled means are estimated using `emmeans::emmeans`. The means for all combinations of Temp and CO2 are obtained with the `specs` argument.

```
m1.emm <- emmeans(m1, specs=c("Temp", "CO2"))
m1.emm

##  Temp CO2  emmean     SE df lower.CL upper.CL
##  13   400    8.23 0.726 20     6.72    9.75
##  18   400   12.74 0.726 20    11.23   14.26
##  13  1100    7.92 0.726 20     6.40    9.43
##  18  1100   9.74 0.726 20     8.23   11.26
##
##  Confidence level used: 0.95
```

15.3.3 Marginal means

The marginal means are

```
m1.emm.temp <- emmeans(m1, specs=c("Temp"))
m1.emm.co2 <- emmeans(m1, specs=c("CO2"))
m1.emm.temp

##  Temp emmean     SE df lower.CL upper.CL
##  13    8.07 0.513 20     7.0    9.15
##  18   11.24 0.513 20    10.2   12.31
##
##  Results are averaged over the levels of: CO2
##  Confidence level used: 0.95

m1.emm.co2

##  CO2  emmean     SE df lower.CL upper.CL
##  400   10.49 0.513 20     9.42   11.6
##  1100   8.83 0.513 20     7.76    9.9
##
##  Results are averaged over the levels of: Temp
##  Confidence level used: 0.95
```

15.3.4 Contrasts

All six pairwise contrasts are computed using `emmeans::contrast`. The `adjust` argument specifies the adjustment for multiple testing. The `method` argument specifies the type of contrast (pairwise and `revpairwise` give all pairwise contrasts. `revpairwise` simply gives the reverse of pairwise)

```
m1.contrast <- contrast(m1.emm, adjust="none", method="revpairwise")
# add CIs
m1.contrast.ci <- summary(m1.contrast, infer=c(TRUE, TRUE))
m1.contrast.ci
```

```

## contrast estimate SE df lower.CL upper.CL t.ratio p.value
## 18,400 - 13,400 4.510 1.03 20 2.370 6.650 4.395 0.0003
## 13,1100 - 13,400 -0.317 1.03 20 -2.457 1.824 -0.309 0.7608
## 13,1100 - 18,400 -4.827 1.03 20 -6.967 -2.686 -4.704 0.0001
## 18,1100 - 13,400 1.508 1.03 20 -0.632 3.649 1.470 0.1571
## 18,1100 - 18,400 -3.002 1.03 20 -5.142 -0.861 -2.925 0.0084
## 18,1100 - 13,1100 1.825 1.03 20 -0.315 3.965 1.779 0.0905
##
## Confidence level used: 0.95

```

15.3.5 Simple effects

The four conditional (simple) effects are a subset of the contrasts above and are computed using the arguments `simple="each"` and `combine=TRUE`.

```
m1.effects <- summary(contrast(m1.emm,
                                method="revpairwise",
                                adjust="none",
                                simple = "each",
                                combine=TRUE),
                                infer=c(TRUE, TRUE))
```

```

##  CO2   Temp contrast      estimate      SE df lower.CL upper.CL t.ratio p.value
## 400   .    18 - 13       4.510 1.03 20    2.370   6.650  4.395  0.0003
## 1100  .    18 - 13      1.825 1.03 20   -0.315   3.965  1.779  0.0905
##  .    13  1100 - 400     -0.317 1.03 20   -2.457   1.824 -0.309  0.7608
##  .    18  1100 - 400     -3.002 1.03 20   -5.142  -0.861 -2.925  0.0084
##
## Confidence level used: 0.95

```

15.3.6 Marginal effects

The marginal effects of the factorial model are

```
m1.emm.1 <- emmeans(m1, specs=c("Temp"))
m1.effects.1 <- summary(contrast(m1.emm.1,
    method="revpairwise",
    adjust="none"),
    infer=c(TRUE,TRUE))
m1.effects.1

##  contrast estimate     SE df lower.CL upper.CL t.ratio p.value
##  18 - 13      3.17 0.726 20     1.65     4.68 4.366   0.0003
##
## Results are averaged over the levels of: CO2
## Confidence level used: 0.95

m1.emm.2 <- emmeans(m1, specs=c("CO2"))
m1.effects.2 <- summary(contrast(m1.emm.2,
    method="revpairwise",
    adjust="none"),
    infer=c(TRUE,TRUE))
m1.effects.2

##  contrast estimate     SE df lower.CL upper.CL t.ratio p.value
##  1100 - 400     -1.66 0.726 20     -3.17    -0.146 -2.287   0.0332
##
## Results are averaged over the levels of: Temp
## Confidence level used: 0.95
```

These can be combined into a single table using `rbind`

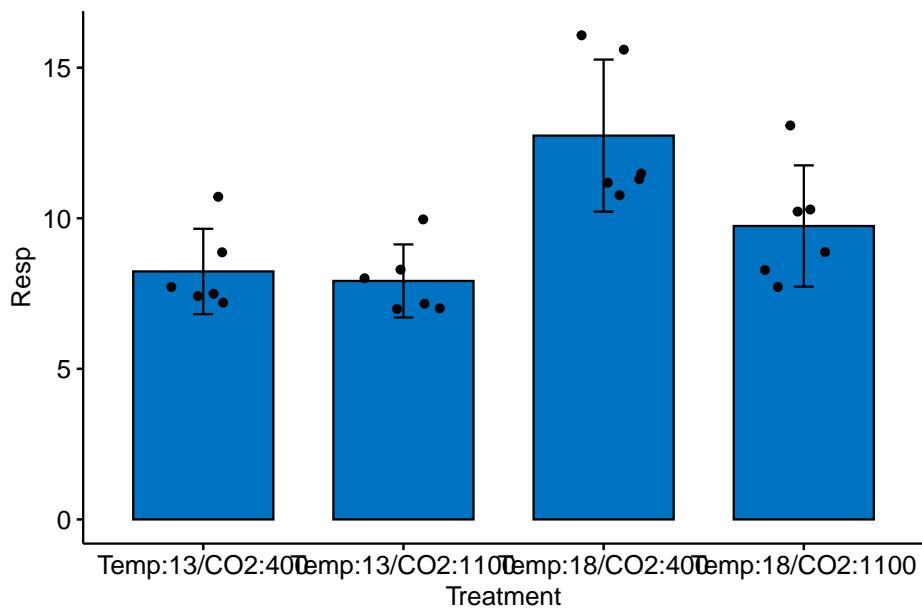
```
m1.effects.marginal <- rbind(data.table(m1.effects.1), data.table(m1.effects.2))
m1.effects.marginal

##      contrast estimate     SE df lower.CL upper.CL t.ratio
## 1: 18 - 13  3.167500 0.7255577 20  1.654013  4.6809869 4.365607
## 2: 1100 - 400 -1.659167 0.7255577 20 -3.172654 -0.1456798 -2.286747
##          p.value
## 1: 0.0002993051
## 2: 0.0332473272
```

15.3.7 Plotting results

15.3.7.1 Bar plot with uniform coloring poorly communicate the factorial design

```
# bar plot with uniform color
urchin[, xlabel := paste0("Temp:",Temp,"/", "CO2:",CO2)]
ggbarplot(x=xlabel,
           y="Resp",
           data=urchin[!is.na(Resp),],
           add=c("mean_ci", "jitter"),
           fill=(pal_jco("default")(4))[1]) +
xlab("Treatment") +
NULL
```



15.3.7.2 Plots that communicate the factorial design

```
# bar-plot with 2nd factor different color
pd <- position_dodge(0.7)
gg1 <- ggbarplot(x="Temp",
                  y="Resp",
                  fill="CO2",
                  data=urchin[!is.na(Resp),],
```

```

      add=c("mean_ci"),
      position=pd) +
geom_point(aes(fill=C02),
            color="black",
            position=position_jitterdodge(jitter.width=0.2),
            show.legend=FALSE,
            alpha=0.5) +
scale_fill_jco() +
NULL

# "interaction" plot
m1.emm.dt <- data.table(summary(m1.emm))
pd = position_dodge(0.7)
gg2 <- ggplot(data=m1.emm.dt,
               aes(x=Temp,
                    y=emmmean,
                    shape=C02,
                    color=C02,
                    group=C02)) +
geom_point(position=pd, size=3) +
geom_errorbar(aes(x=Temp,
                  ymin=lower.CL,
                  ymax=upper.CL,
                  group=C02),
              , position=pd, width=0.1) +
geom_line(position=pd) +
ylab("Resp") +
scale_color_jco() +
theme_pubr() +
#theme(legend.position="bottom") +
NULL

# interaction "jitter" plot
gg3 <- gg2 +
  geom_point(data=urchin[!is.na(Resp),], aes(x=Temp, y=Resp, fill=C02),
             position=position_jitterdodge(jitter.width=0.2)) +
#           position=position_jitter(width=0.2)) +
  theme(legend.position="bottom") +
NULL
gg_response <- gg3 # used below

# box "interaction" plot
m1.emm.dt <- data.table(summary(m1.emm))
pd <- position_dodge(0.8)
gg4 <- ggboxplot(x="Temp",

```

```

y="Resp",
  data=urchin[!is.na(Resp),],
  fill="CO2") +
scale_fill_jco() +
geom_point(data=m1.emm.dt,
            aes(x=Temp, y=emmmean, group=CO2),
            color="red",
            position=pd) +
geom_line(data=m1.emm.dt,
           aes(x=Temp, y=emmmean, group=CO2),
           position=pd) +
theme(legend.position="bottom") +
NULL

plot_grid(gg1, gg2, gg3, gg4, nrow=2, labels="AUTO")

```

A common way to plot the results of factorial models is with an **interaction plot** (Figure 15.2). In the interaction plot of the urchin data, the *X*-axis contains the two *Temp* treatment levels and the *Y*-axis is the outcome (*Resp*). The plot shows the four cell means indicated by the circles (low CO₂ levels) or triangles (high CO₂ levels). The solid lines connect the cell means *across Temp levels within CO₂ levels*.

1. The slope of a line is the effect of *Temp* on *Resp*
2. The relative *elevation* of the two lines is the effect of *CO₂* on *Resp*
3. The difference in slope *or* the relative elevation at each level of *Temp* is the interaction effect

Let's deconstruct this. The top (CO₂-) line is the effect of *Temp* at the control (400 µATM) value of *CO₂*. The slope of the bottom (CO₂+) line is the effect of *Temp* at the high (1100 µATM) value of *CO₂*. *These lines have different slopes*, or the slope *is conditional on* the level of CO₂. This means that the effect of *Temp* on respiration is *conditional on the value of CO₂*. Think about this. This is what an interaction implies-conditional effects.

At the reference temperature (13 °C), the CO₂+ line is barely below the CO₂- line. But at the high temperature (18 °C), the CO₂+ line is far below the CO₂- line. That is, the relative elevation (the *CO₂* effect) is conditional on the level of *Temp*. It will always be the case that if the effect of Factor A is conditional on the levels of Factor B, then the effect of Factor B will be conditional on the levels of Factor A.

An interaction plot is an okay plot. It doesn't show the data, only a minimal, descriptive summary (means and standard errors). If we are interested in the interaction effect, it doesn't give us a very good sense of the error in this effect.

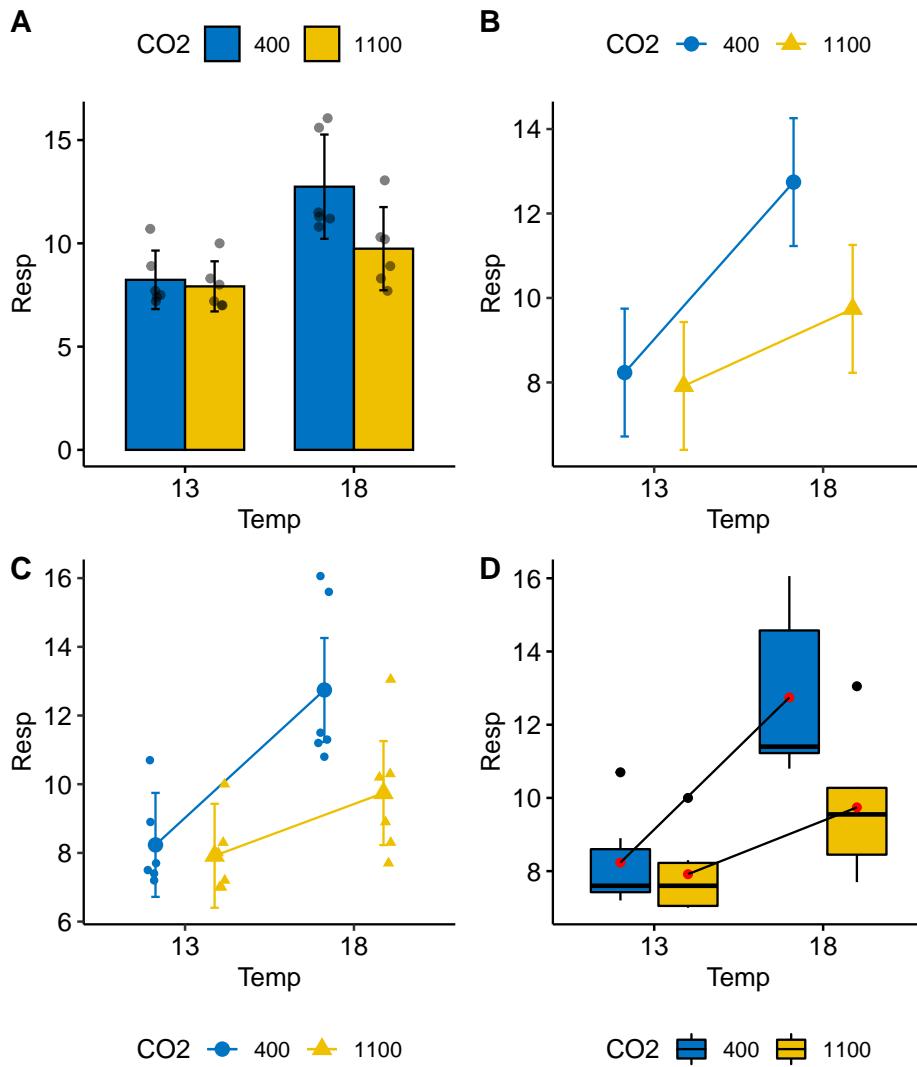


Figure 15.2: Interaction plots. (B) is the classic interaction plot, which is characterized by lines connecting the groups that share the same Factor B level. This line allows one to visual the effect of Factor A (the slope) at each level of Factor B.

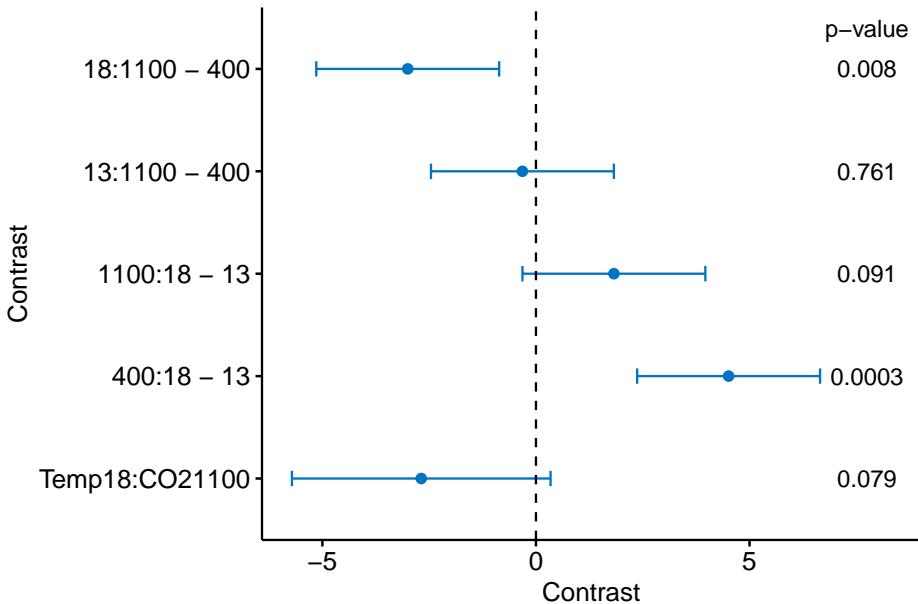
And *that* is a problem because with real data, two lines are never precisely parallel. Our interpretation of the similarity of the slopes would probably mostly reflect our pre-conceived scientific model.

15.3.7.3 Effects plots

```
# need m1.emm and m1.effects from above
# convert to data.table
m1.coefs <- coef(summary(m1))
m1.ci <- confint(m1)
m1.coefs.dt <- data.table(Term=row.names(m1.coefs), m1.coefs, m1.ci)
# convert labels to match those of m1.effects
setnames(m1.coefs.dt,
          old=c("Estimate", "Std. Error", "Pr(>|t|)", "2.5 %", "97.5 %"),
          new=c("estimate", "SE", "p.value", "lower.CL", "upper.CL"))
m1.contrasts.dt <- data.table(m1.effects)
# create a label for each contrast
m1.contrasts.dt[, Term:=ifelse(CO2!=".", ,
                                paste0(CO2, ":", contrast),
                                paste0(Temp, ":", contrast))]
m1.effects.dt <- rbind(m1.coefs.dt[4,], m1.contrasts.dt, fill=TRUE)

# effects plot
# get p-values
pval <- as.character(round(m1.effects.dt$p.value, 3))
pval[2] <- "0.0003"
gg_effects <- ggdotplot(x="Term",
                          y="estimate",
                          data=m1.effects.dt,
                          color = (pal_jco("default")(4))[1],
                          fill = (pal_jco("default")(4))[1],
                          size=0.5) +
  geom_errorbar(aes(x=Term, ymin=lower.CL, ymax=upper.CL),
                width=0.15, color=(pal_jco("default")(4))[1]) +
  ylab("Contrast") +
  geom_hline(yintercept=0, linetype = 2) +
  annotate("text", x = 1:5, y = rep(7.75, 5), label = pval) +
  annotate("text", x=5.4, y=7.75, label="p-value") +
  expand_limits(y = 8.3) +
  xlab("Contrast") +
  coord_flip() +
  NULL

gg_effects
```

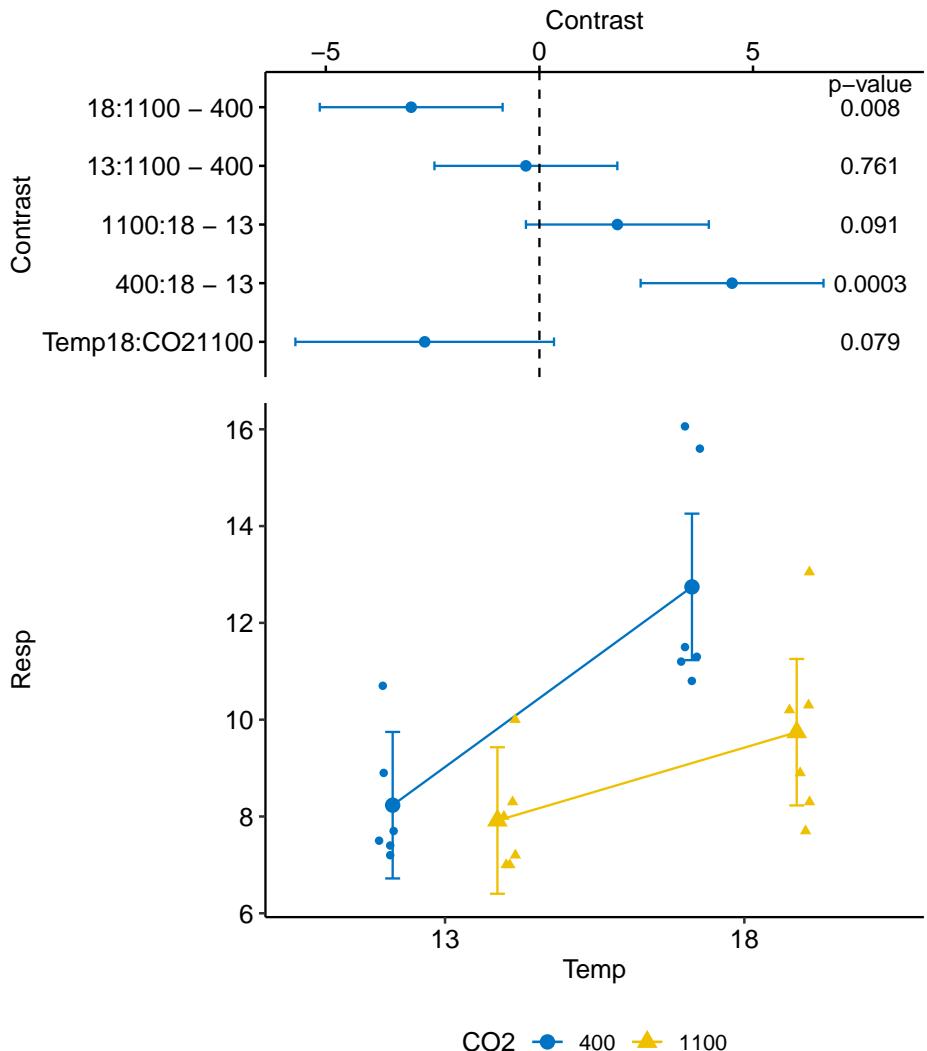


This effects plot shows the four simple effects, the single interaction (Temp18:CO21100), and their 95% confidence intervals. In the original paper, the researchers were testing a scientific (not statistical!) model that predicted no interaction between CO₂ and Temp, and the researchers argued that these data supported this model because of the “not statistically significant” *p*-value for the interaction effect. The data are consistent with this model (one end of the 95% CI for the interaction includes zero) but also support a model of a large, negative interaction (the other end of the 95% CI includes large negative values). The data are too coarse (or the signal:noise ratio is too small) to have much confidence in the size of the interaction effect.

15.3.7.4 Harrell plots

```
gg_effects <- gg_effects + scale_y_continuous(position="right")

plot_grid(gg_effects, gg_response, nrow=2,
          align = "v",
          rel_heights = c(1, 1.75))
```



The effects and interaction plot are combined into a single plot.

15.4 Problems

1. Draw four 2×2 tables and label the row and column headers using the levels of the urchin treatment. In the first table, insert the cell means. In the 2nd table, insert the equation for the coefficient. In the third table, solve the equations. And in the fourth column, insert the estimates from the table above. Are tables 3 and 4 the same? If not, you've goofed somewhere.

2. Frew et al. (2017) showed that increasing atmospheric CO₂ increases grub activity in the soil which in turn increases root damage to sugarcane. They used a 2 x 2 experiment to then show that silicon added to the soil decreased the damage to the roots by the grubs (silicon minerals are very hard and plants uptake silicon from the soil to mineralize tissues to protect against insect damage). There are lots of analyses in the paper – try to reproduce Fig. 4b, but using an interaction plot.

(The treatment assignments are in a different file than the experimental results. Use the `merge` function to glue the two tables together, keying on the common column “plant”)

1. **file name:** “canegrub_feedingtrial.csv”
2. **file name:** “treatments.csv”
3. **source:** <https://datadryad.org/resource/doi:10.5061/dryad.r3s16>

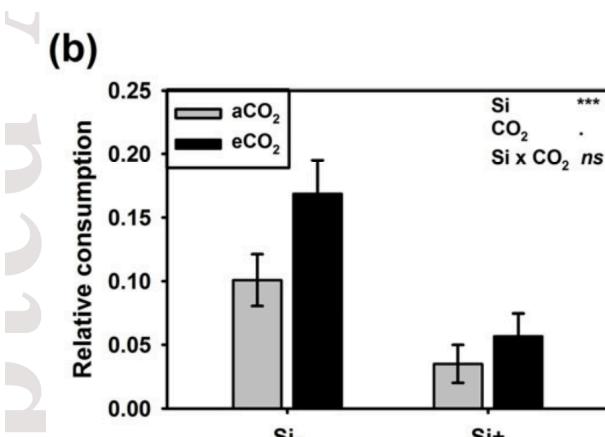


Figure 4. (a) Effects of silicon treatments on the relative growth rate [=mass gained (g)/time(days)] and (b) relative consumption of roots [=food ingested (mg change in dry mass)/ initial body mass (mg fresh mass)] under aCO₂ and eCO₂. Levels of significance are shown for effects of silicon and CO₂ treatments. Values are means ± SE. Degrees of significance are indicated as follows: ns = not significant, . P<0.1, * P<0.05, ** P<0.01, *** P<0.001.

3. Kardol et al investigated the effect of moss growth in response to rainfall and community structure. Analyze the effect of these two factors on biomass gain and generate a Harrell plot alternative to their bar plot in Fig. 3 (see below). What is striking about your plot compared to theirs?

Filename “Data file for Dryad.xlsx” sheet “Data” **Source**: <https://datadryad.org/resource/doi:10.5061/dryad.66d5f>

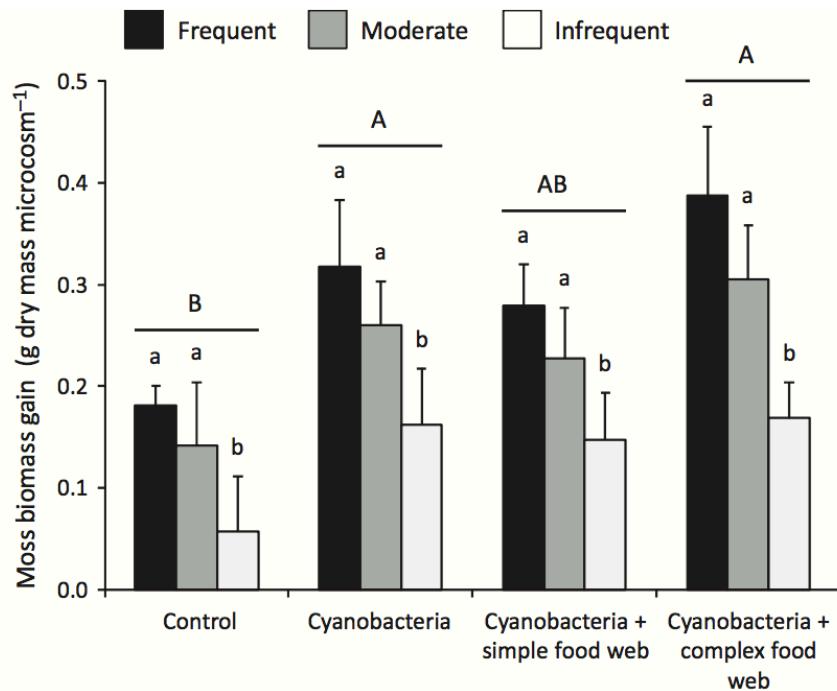


Figure 3 Responses of moss biomass gain to food-web structure (control consisting of sterilised moss with no cyanobacteria added, cyanobacteria, cyanobacteria + simple food web, cyanobacteria + complex food web) and precipitation regime (frequent, moderate, infrequent). Uppercase letters above groups of bars denote significant differences (*post hoc* comparisons based on estimated marginal means, $P < 0.05$) among moss food-web treatments. Lowercase letters above bars indicate significant differences among precipitation regimes *across* food-web treatments. Data are means \pm SE ($N = 16$). ANOVA results are shown in Table 2.

4. (Grad students only) Generate a fake experiment! The experiment should have two factors each with two levels. Experiment with power by varying sample size and effect size.

Chapter 16

ANOVA Tables

Treatment effects are most often analyzed using ANOVA, which is short for “Analysis of Variance”. This is somewhat of an odd name for a method to test for treatments effects - what do differences in means have to do with an analysis of variance? The name makes sense in light of the decomposition of the total variance into a model variance and the residual variance (chapter xxx). If there are differences among the means, then the total variance is increased because of variation among groups.

The engine underneath modern ANOVA is a linear model. If the model has a single categorical factor, the ANOVA is **one-way**. If the model has two categorical factors it is a two-way ANOVA. If the model has a single categorical factor and one continuous factor it is an ANCOVA, short for **analysis of covariance** (next chapter). More complex experimental designs classically analyzed with ANOVA are nested, split-plot, latin-square, and many others.

16.1 Summary of usage

If you choose to report an ANOVA, also report the effects and their uncertainty in some way, either the model coefficients or contrasts.

1. ANOVA generates a table with one row for each term in the linear model. A term is a factor or a covariate or an interaction. For a two-way factorial ANOVA, these terms are the two main effects and the interaction effect.
2. The ANOVA generates an F and p -value for the whole model and for each term in the ANOVA table.
3. The p -value of an interaction term is often used as a decision rule to interpret the main effects. If $p \leq 0.05$ then do not interpret the main effects but instead examine the condition (“simple”) effects. If $p > 0.05$,

then interpret the main effects. Regardless, this sort of decision rule is itself controversial, and for good reason.

4. If the main effects are to be interpreted, some statisticians advocate refitting the model without the interaction effect, others advocate interpreting the main effects with the interaction term in the model. This only matters if the design is unbalanced (see below).
5. Regardless of any decision, always plot the data using a Harrell plot or interaction plot to understand and communicate the magnitude and pattern of interaction.
6. For factors with more than two levels, the p -value is often used as a decision rule to dissect the factor with post-hoc tests, such as Tukey HSD.
7. A design is balanced if all the cells have the same number of replicates. A design is unbalanced if one or more of the cells has a different number of replicates. Unbalanced designs make it necessary to make decisions, none of which are perfect, and all of which are controversial. Some statisticians have even advocated randomly excluding data until the design is back in balance. Don't do this.
8. There are multiple ways to decompose the sum of squares. I highlight the major three: Type I (sequential), Type II (partial sequential), and Type III. Most statistics software and introductory statistics books default to Type III and, consequently, many researchers are unaware that Types I and II exist. R's default is Type I, and this can make a difference if the design is unbalanced. This is *not* a rare error in publications.
9. Because R defaults to Type I sum of squares, the p -value of a factor depends on the order of the factors in the model if the design is unbalanced. This is a feature, not a bug.
10. ANOVA based on type II sum of squares do not depend on factor order if the design is unbalanced, but it does assume that the interaction is zero.
11. ANOVA based on type III sum of squares do not depend on order if the design is unbalanced and does not assume the interaction is zero.
12. If the design is balanced, Type I, II, and III sum of squares generate the same ANOVA table. And the ANOVA table of just the main effects is the same as the ANOVA table that includes the interaction term. None of this is true when the design is unbalanced. However, the decision to use type II or type III is very controversial.

16.2 Example: a one-way ANOVA using the vole data

The vole data has a single factor (“treatment”) with three levels (“control”, “vitamin_E”, “vitamin_C”). In statistics textbooks that emphasize hypothesis testing, the “Which test should I use” flowchart would guide a researcher given this design to a single classification, or one-way ANOVA, since a t-test can only compare two levels but an ANOVA can compare more than two levels. There

are better ways to think about what ANOVA is doing, but okay.

Here is an ANOVA table of the vole data:

Df	
Sum Sq	
Mean Sq	
F value	
Pr(>F)	
treatment	
2	
248446	
124223.0	
2.95	
0.057	
Residuals	
93	
3912751	
42072.6	

I'll explain all the parts of the ANOVA table later, but for now, focus on the p -value, which is that most researchers want out of the table. What null hypothesis does this p -value test? The p -value gives the probability of the observed F or larger F , if the null were true. The null hypothesis models the data as if they were sampled from a single, normal distribution and randomly assigned to different groups. Thus the null hypothesis includes the equality of the means among factor levels. In the vole data, the single treatment factor has three levels and a small p -value could occur because of a difference in means between the vitamin_E treatment and control, or between the vitamin_C treatment and control, or between the two vitamin treatments. The p -value or ANOVA table doesn't indicate what is different, only that the observed F is unexpectedly large if the null were true. As a consequence, researchers typically interpret a low p -value in an ANOVA table as evidence of "an effect" of the term but have to use additional tools to dissect this effect. The typical additional tools are either **planned comparisons**, which are contrasts among a subset of a priori identified treatment levels (or groups of levels) or unplanned comparisons ("post-hoc" tests) among all pairs of levels.

The p -value in the ANOVA table acts as a decision rule: if $p < 0.05$ then it is okay to further dissect the factor with planned comparisons or post-hoc tests because the significant p "protects" the type I error of further comparisons. I'm not fond of using p -values for these sorts of decision rules.

16.3 Example: a two-way ANOVA using the urchin data

Let's use the urchin data from the previous chapter xxx to explore the ANOVA table, which is what is typically reported. The experiment has two factors (*Temp* and *CO2*), each with two levels. Here is the linear model

$$Resp = \beta_0 + \beta_1 Temp + \beta_2 CO2 + \beta_3 TempCO2 + \varepsilon \quad (16.1)$$

In order to understand factorial ANOVA (or any ANOVA with multiple factors), it is useful to know the difference between **conditional means** and **marginal means**

```
##          CO2-  CO2+ Temp-mm
## Temp-    8.233 7.917   8.075
## Temp+   12.743 9.742  11.243
## CO2-mm 10.488 8.829   9.659
```

In the table above, the upper, left 2×2 grid of cells are the conditional means, which are the means of each group, where a group is a specific combination of factor levels. The first two values of the third row are the marginal means for *CO2*. The first (10.488) is the mean of the two means when *CO2*=*CO2-*. This can be written as $E(Resp|CO2-)$. The second (8.829) is the mean of the two means when *CO2*=*CO2+*, or $E(Resp|CO2+)$. The first two elements of the third column are the marginal means for *Temp*. These are $E(Resp|Temp-)$ and $E(Resp|Temp+)$. The bottom right value (9.659) is the grand mean.

A **conditional effect** is a difference between conditional means. For example the conditional effect of *Temp* conditional on *CO2*=*CO2-* is $12.743 - 8.233$. A **marginal effect** is a difference in marginal means within a factor, for example the marginal effect of *Temp* is $11.243 - 8.075$.

Here is the ANOVA table of the urchin data

```
Df
Sum Sq
Mean Sq
F value
Pr(>F)
Temp
1
60.2
```

60.2

19.1

0.0003

CO2

1

16.5

16.5

5.2

0.0332

Temp:CO2

1

10.8

10.8

3.4

0.0791

Residuals

20

63.2

3.2

This ANOVA table uses what are called Type 3 sum of squares, which is *NOT* the default in R but is the default in many other statistics software and is, therefore, the *only* type of ANOVA that many researchers know (and, many researchers are unaware that there are multiple types of ANOVA table). Understanding these differences is important, at least if one is reporting ANOVA tables. I'll return to the importance of this later.

16.3.1 How to read an ANOVA table

An ANOVA table has a row for each term in the underlying linear model – each of these adds a component of variance to the total, and a row for the residual variance (this residual variance row is frequently excluded from the published table). The urchin model has three terms (one level of *Temp*, one level of *CO2*, and one interaction). The statistics for each term are

1. **Degrees of Freedom (df)** – If the term is a factor, the df will equal the number of levels (k) for the factor minus 1. Think of it this way: the contribution of the variance due to a factor is a function of the variability of the k level means around the grand mean. How many degrees of independent variation do these level means have, given that we know the grand mean? The answer is $k - 1$ – once the values of $k - 1$ level means are written down, the k th level mean has no freedom to vary; its value has to be $\bar{Y} - \sum_i^{k-1} Y_i$. For an interaction term, the df is the product of the df of each of the factors in the interaction.
2. **Sum of Squares** – the sum of squared differences between the modeled value and the grand mean. In addition to a sum of squares for each term, a **residual mean square** is computed as the sum of squared differences between the measured and modeled values.
3. **Mean Square** – The sum of squares divided by the df (this is a “mean” with df acting as the number of things that were summed).
4. **F-ratio** – the Mean Square of the term divided by the residual mean square.
5. **p-value** – the p-value for the F-ratio. F is compared to an F-distribution, which is a distribution of F-ratios under the null.

16.3.1.1 Each row in the table tests a null hypothesis

The row for each term in an ANOVA table tests a null hypothesis. In order to understand the null hypotheses, I need to define a few more terms

For the ANOVA table above, which uses Type 3 sum of squares, the probabilities are

1. Temp – $p = \text{prob}(F \geq F_o | CO2, Temp : CO2)$. The null is no difference in means conditional on the level of CO2 and Temp:CO2. This is equivalent to no difference between the grand mean and the marginal mean of Temp+, or

$$b_1 = \overline{\overline{Resp}} - E(Resp | Temp^+) \quad (16.2)$$

2. CO2– $p = \text{prob}(F \geq F_o | Temp, Temp : CO2)$. The null is no difference in means conditional on the level of Temp and Temp:CO2. This is equivalent to no difference between the grand mean and the marginal mean of CO2+, or

$$b_2 = \overline{\overline{\overline{Resp}}} - E(Resp | CO2^+) \quad (16.3)$$

3. Temp:CO2 – $p = \text{prob}(F \geq F_o | \text{Temp}, \text{CO2})$. The null is no difference in means conditional on the level of Temp and CO2. This is equivalent to the difference between the conditional mean of Temp+/CO2+ and the expected conditional mean of Temp+/CO2+ if there were no interaction.

$$b_3 = E(\text{Resp} | \text{Temp}^+, \text{CO2}^+) - (\overline{\text{Resp}} - b_1 - b_2) \quad (16.4)$$

As noted in the equations, these three differences are the coefficients of the linear model behind the ANOVA. Here is the coefficient table

Estimate

Std. Error

t value

$\text{Pr}(>|t|)$

(Intercept)

9.66

0.36

26.6

0.00000

Temp1

-1.58

0.36

-4.4

0.00030

CO21

0.83

0.36

2.3

0.03325

Temp1:CO21

-0.67

0.36

-1.9

0.07910

In ANOVA with type 3 sum of squares, the dummy variables are coded using effect coding, which differs from the dummy coding introduced in chapter xxx. The consequence is that the **grand mean** (the mean of *Resp* across all values) is now the “reference” value. The intercept in this table, then, is the grand mean. The coefficients are *differences from the grand mean*, as described above.

Use the table of conditional and marginal effects above to check that the coefficients equal the differences in the equations above. Also note that the *p*-values for the effects in the coefficient table equals the *p*-values in the ANOVA table.

It is important to note that this table differs from the coefficient table with dummy coding because that reference is the mean of Temp-/CO2- and not the grand mean.

Estimate

Std. Error

t value

$\text{Pr}(>|t|)$

(Intercept)

8.23

0.73

11.3

0.00000

TempTemp+

4.51

1.03

4.4

0.00028

CO2CO2+

-0.32

1.03

-0.3

0.76081

TempTemp+:CO2CO2+

-2.68

1.45

-1.9

0.07910

Importantly, note that p -values for b_1 (the Temp effect) and b_2 differ between the two tables. This is because the t -value tests different hypotheses! In the coefficient table with effect coding (that behind the ANOVA with type 3 sums of squares), the p -value tests marginal effects and so is a function of both marginal means within a factor. By contrast, in the coefficient table with dummy coding, the p -value tests conditional effects, and so is only a function of the conditional means when the other factor is at its reference level (right? The coefficient b_1 in the dummy coded coefficient table is the effect of only increasing $\text{Temp} - \text{CO}_2$ is left at its reference level). For the interaction effect, the coefficient differs between the effects coded model and the dummy coded model (because different reference means) but the p -value ultimately tests the same hypothesis (non-additive effects of the factors) and so the t and p values are the same.

16.3.1.2 What to do after ANOVA?

Researchers frequently report ANOVA statistics (F and p values) for factorial models in a way that suggests that they misunderstand the hypotheses tested. It probably doesn't help that there is a long-standing debate among statisticians about the most sensible strategy for interpreting factorial ANOVA results. And it doesn't help that the default ANOVA table in R can suggest a very different interpretation than the default ANOVA table in some other software packages.

Here are three strategies for interpreting a factorial ANOVA table that uses Type III sum of squares. All strategies use p -values to make a series of decision rules. In the first strategy, which is a type of model simplification or model selection, a researcher starts with the interactions at the bottom of the ANOVA table and works up, eliminating terms with $p > 0.05$ and re-fitting the reduced model before interpreting main effects. In the second strategy, the researcher uses the original ANOVA table that includes all terms to interpret main effects.

Strategy 1

1. is interaction $p < 0.05$?
 - a. if yes, then do NOT test main effects. Show a graph to show pattern of conditional effects. Test conditional effects if this is of interest.
 - b. if no, then refit model without the interaction and test main effects
 - This now is equivalent to ANOVA using Type II sum of squares.
2. is main effect $p < 0.05$?
 - a. if yes, then keep in model
 - b. if no, then refit model without that main effect

Strategy 2

2. is interaction $p < 0.05$?
 - a. if yes, then do NOT test main effects. Show a graph to show pattern of conditional effects. Test conditional effects if this is of interest.
 - b. if no, then use the same table as the test of the main effects. This is interpreting the main effects with the interaction term in the model. This is the logic of ANOVA using type III sum of squares.

Strategy 3

3. is interaction $p < 0.05$?
 - a. if yes, then look at interaction plot to determine if it makes sense test main effects. For example, if CO2+ had obviously lower *Resp* at both levels of *Temp*, even if one was much lower (ie. interactaction), then some people would say that the test of the main effect is meaningful. Test conditional effects if this is of interest.
 - b. if no, then use the same table as the test of the main effects

In general, statisticians advise against strategy 3 (interpreting main effects in the presence of interaction) – its not wrong, its just that a main effect has an awkward interpretation if there is an interaction. Of course this is true if there is *any* interaction term in the model, not just a statistically significant term. The controversy is more, if the interaction p is not significant, then do we implement stategy 1 (refit model excluding interaction to test main effects) or strategy 2 (use full factorial anova table to test main effects).

Df

Sum Sq

Mean Sq

F value

Pr(>F)

Temp

1

45.2

45.2

14.5

0.0011

CO2

1

4.1
 4.1
 1.3
 0.2630
 Temp:CO2
 1
 14.8
 14.8
 4.8
 0.0413

then one shouldn't report the ANOVA results using something like "Temperature had a significant effect on metabolism ($F_{1,20} = 14.5, p = 0.001$). There was no effect of CO₂ on metabolism ($F_{1,20} = 4.1, p = 0.26$)". There was a significant interaction effect between Temperature and CO₂ on metabolism ($F_{1,20} = 14.8, p = 0.04$)". If one accepts that the small interaction *p*-value is evidence of an interaction effect then this interpretation of the main factors makes no sense, as the first two results imply that the interaction effect is zero (or, that there is a constant effect of *Temp* or *CO₂* across both levels of the other factor), which is then contradicted by the third result.

More specifically, if one is using a *p*-value to guide decision making, then a significant interaction *p* indicates that there is no single "main" effect of a factor. Instead, the effect of *Temp* is conditional on the level of *CO₂*, and the effect of *CO₂* is conditional on the level of *Temp*. This is easily seen in the interaction plot, where the effect of *Temp* is large when *CO₂* is high but much smaller when *CO₂* is low. Indeed, the effect of *Temp* at the low *CO₂* is 0.16.

Instead of interpreting the factors as constant effects, A better strategy is to compare the **conditional effects**, that is, the effects of *Temp* within each level of *CO₂* and the effects of *CO₂* within each level of *Temp* (conditional effects are sometimes called the "simple effects").

The controversy arises in what to do after an ANOVA if the interaction effect has a non-significant *p*-value. At this point, I am punting instead of explaining the basis for the controversy, because ultimately I think the major problem with both strategies is the use of null hypothesis significance testing to make analysis decisions.

In fact, the entire reason that I use the urchin data as the example for factorial ANOVA is because it beautifully illustrates the absurdity of the interaction *p*-value decision rule. Why should we interpret the results of the ANOVA when the interaction *p* is 0.079 differently than when the interaction *p* is 0.04? Remember, the *p*-value is a "sample" statistic (in the sense that it is entirely a function of

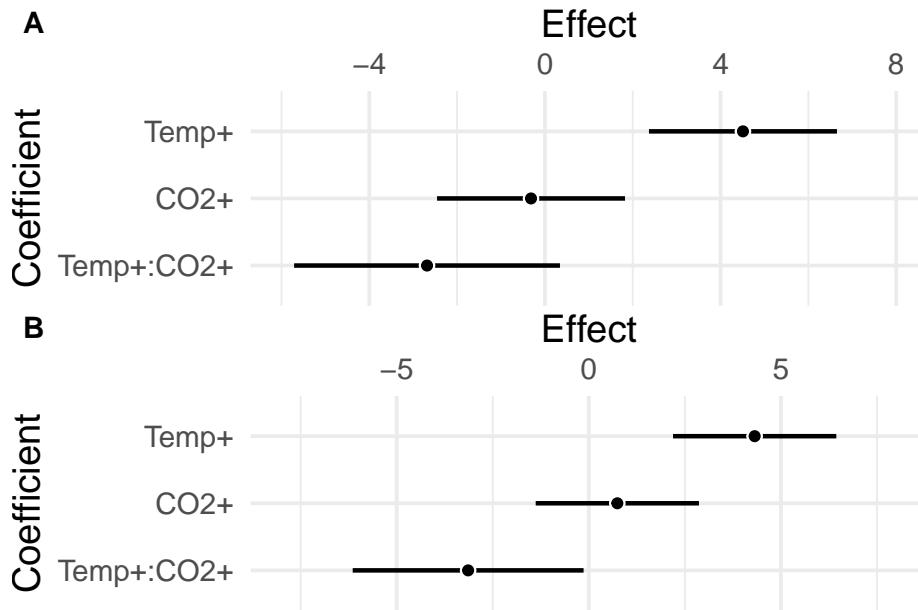


Figure 16.1: Forest plots (the upper part of a Harrell plot) of the actual and fake urchin data. A) Real urchin data. The interaction effect is not significant ($p = 0.079$). B) Fake urchin data. The interaction effect is significant ($p = 0.04$).

the sampled data) and in conditions of low power (which is likely, but not necessarily, true for the urchin data given $n=6$), a p -value is highly variable.

There are several problems with this approach. 1) a p -value is not evidence of “no effect”, 2) the power to test interaction effects is small relative to that for the main effects (this is a general rule, not something specific to these data), 3) the interaction SS accounts for about 7.2% of the total SS, which doesn’t seem inconsequential, and 4) the interaction p -value is small enough to raise a red flag, and, most importantly, 5) the confidence interval of the interaction effect indicates that the large, negative values of the interaction are *as consistent with the data* as trivially small values (or a value of zero). But the CI is not in an ANOVA table and many researchers fail to report it. These five points suggest that this experiment be replicated, with a larger sample size, to get a better estimate of the interaction effect. The problem of course is that experiments are rarely replicated, except in biomedical research.

The absurdity of the p -value decision rule strategy for interpretation of effects after an ANOVA is highlighted by comparing the forest plot of model coefficients of the real and fake urchin data. It would be absurd to use an ANOVA table to interpret these patterns as radically different (one without an interaction and constant main effects, the other with an interaction and conditional effects).

16.3.2 How to read ANOVA results reported in the text

ANOVA results are often reported in the text of a results section, using something like “Temperature had a significant effect on metabolism ($F_{1,20} = 14.5$, $p = 0.001$). There was no effect of CO₂ on metabolism ($F_{1,20} = 4.1$, $p = 0.26$)”. The subscripts of the F statistic are the numerator and denominator degrees of freedom (df) of the F -value (These df are a column in the ANOVA table. The denominator df may not appear in the table if it is the residual df and the row for the residual term was not reported). Sometimes I find the reported df are not consistent with the description of the design and analysis, which means the data were not analyzed as stated.

16.3.3 Better practice – estimates and their uncertainty

As emphasized in the previous chapter, the decision to include or exclude an interaction effect in the model should not be based on a p -value but on the goals of the model.

1. If the goal is the interaction (because a scientific model predicts one, or because this is biology and everything is conditional), then estimate the interaction effect (as a coefficient of the model!) and its uncertainty, including a CI and p -value. There is no controversy on how to estimate this effect and its uncertainty. The coefficient will be different between dummy and effect coded models but this is okay because they have different specific interpretations but the same general interpretation. Use a Harrel plot with the coefficients (including the interaction coefficient) to show this estimate and uncertainty.
2. If the goal is to estimate constant main effects, then exclude the interaction effect from the model and report the main effects (again, as coefficients from the model or contrasts if other pairwise effects are desired) with their uncertainty. Use an interaction plot (or bottom part of the harrell plot) to justify forcing the interaction to zero (for example the interaction effect adds little to the total sum of squares or the interpretation of a single main effect or two (or more) conditional effects would be the same. Use a Harrel plot that excludes the interaction term to show these main effects and uncertainty.
3. And if a researcher is interested in the effects of the factors but there is strong evidence for a non-trivial interaction, then report the conditional effects (as contrasts) with their uncertainty. Use a Harrel plot that includes the interaction term to show these conditional effects and uncertainty. If there is an obvious interaction, it probably doesn't make sense to interpret the main effects, contrary to what some people argue. If there is a positive effect of factor A across all levels of factor B, we don't really need

a p -value to test that the average of these positive effects is significant. This doesn't add value to the plot and any conditional effects that are reported.

Notice that an ANOVA table has no role in this recommendation.

16.4 Unbalanced designs

My recommendation above is to not bother with ANOVA, but to simply compute the contrasts of interest using the linear model. But if you really want to use ANOVA, you should be aware that **if the design is unbalanced, factor order matters in the default R anova function** and that I routinely find published ANOVA tables that report statistics (F and p values) that are not what the authors think they are.

An **unbalanced** design is one in which the number of replicates differs among the cell. The urchin data is balanced because there are six replicates in each cell. If the respirometer broke before taking the respiratory measures of the final tank, the design would be unbalanced, one of the cells would have only five replicates.

Let's look at the effect of row order on the statistics of the urchin data using R's default anova function.

Df
Sum Sq
Mean Sq
F value
Pr(>F)
Temp
1
60.20
60.20
19.06
0.00030
CO2
1
16.52
16.52

5.23
0.03325
Temp:CO2
1
10.81
10.81
3.42
0.07910
Df
Sum Sq
Mean Sq
F value
Pr(>F)
CO2
1
16.52
16.52
5.23
0.03325
Temp
1
60.20
60.20
19.06
0.00030
CO2:Temp
1
10.81
10.81
3.42
0.07910

Now let's unbalance the data, by removing three random replicates (these may be both in one cell or spread across cells. First, here is the number of replicates in each cell:

```
##          CO2- CO2+
##  Temp-      6    4
##  Temp+      6    5
```

And here are the two tables with the order of Temp and CO2 reversed in the model

Df	
Sum Sq	
Mean Sq	
F value	
Pr(>F)	
Temp	
1	
62.25	
62.25	
18.44	
0.00049	
CO2	
1	
21.49	
21.49	
6.36	
0.02190	
Temp:CO2	
1	
6.38	
6.38	
1.89	
0.18720	

Df

Sum Sq

Mean Sq

F value

$\text{Pr}(>F)$

CO2

1

17.59

17.59

5.21

0.03561

Temp

1

66.14

66.14

19.59

0.00037

CO2:Temp

1

6.38

6.38

1.89

0.18720

Several observations are important.

1. the statistics for the last row, which is the interaction, does not change.
2. if these data were analyzed in the software package JMP, or SAS, or SSPS
then **order wouldn't matter**. Here is what the tables would look like

Sum Sq

Df

F value

Pr(>F)

Temp

58.77

1

17.41

0.00064

CO2

19.93

1

5.90

0.02648

Temp:CO2

6.38

1

1.89

0.18720

Sum Sq

Df

F value

Pr(>F)

CO2

19.93

1

5.90

0.02648

Temp

58.77

1

17.41

0.00064

CO2:Temp

6.38
1
1.89
0.18720

3. Order does not change the statistics in the coefficient table, even for unbalanced data:

Estimate

Std. Error

t value

$\Pr(>|t|)$

(Intercept)

9.50

0.407

23.367

0.0000

Temp1

-1.70

0.407

-4.172

0.0006

CO21

0.99

0.407

2.430

0.0265

Temp1:CO21

-0.56

0.407

-1.374

0.1872

Estimate
Std. Error
t value
Pr(> t)
(Intercept)
9.50
0.407
23.367
0.0000
CO21
0.99
0.407
2.430
0.0265
Temp1
-1.70
0.407
-4.172
0.0006
CO21:Temp1
-0.56
0.407
-1.374
0.1872

16.4.1 What is going on in unbalanced ANOVA? – Type I, II, III sum of squares

Type I sum of squares. Here is the (default) ANOVA table using Type I sum of squares for the urchin data with the three missing rows.

Df
Sum Sq

Mean Sq

F value

Pr(>F)

Temp

1

62.248

62.248

18.4

0.0005

CO2

1

21.488

21.488

6.4

0.0219

Temp:CO2

1

6.377

6.377

1.9

0.1872

Residuals

17

57.399

3.376

The default coding of dummy variables in R's `lm` function is dummy coding, which is the coding used for Type I or **Sequential Sum of Squares**. The hypothesis tested by each row in the ANOVA table using Type I sum of squares is the effect of that row's term conditional on all terms before it in the model (or above it in the table) and ignoring all terms after it in the model (or below it in the table).

1. The hypothesis tested by the p -value for $Temp$ is the same as if $Temp$ were the only term in the model (other than the intercept). That is, the means are estimated for each level of $Temp$ ignoring the fact that half the replicates within each level of $Temp$ experienced low CO_2 and half experienced high CO_2
2. The hypothesis tested by the p -value for CO_2 is conditional on $Temp$. That is, the difference in metabolism between CO_2+ and CO_2- when $Temp$ is “held constant” (or for all cases where $Temp$ takes the same value). This is equivalent to the hypothesis that the difference in the marginal means of CO_2 is zero.
3. The hypothesis tested by the p -value for the interaction is conditional on all other terms and nothing is ignored.

Type II sum of squares. Here is the ANOVA table using Type II sum of squares for the urchin data with missing values. The interaction term is excluded from the linear model, because type II sum of squares are used to estimate main effects ignoring the interaction (so this would make sense only if a plot of the effects suggested a small interaction relative to the main effects). The sum of squares for the main effects would be the same if the interaction were included but the residual df, and thus the F and P-values would differ.

Df	
Sum Sq	
Mean Sq	
F value	
Pr(>F)	
Temp	
1	
66.145	
66.145	
18.7	
0.0004	
CO2	
1	
21.488	
21.488	
6.1	

0.0241
Residuals
18
63.776
3.543

The hypothesis tested by each row in the ANOVA table using Type II sum of squares is the effect of that row's term conditional on all terms *at the same level or below* but ignoring all terms at a higher level in the model (or below it in the table). For example, the hypothesis test for a factor is conditioned on other factors but ignores interaction terms among the factors. Consequently, these hypotheses tested are

1. The hypothesis tested by the *p*-value for *Temp* is conditional on *CO2*. This is the same hypothesis that would occur using Type I sum of squares but placing *Temp* second in the model, after *CO2* (and it is in fact how I computed it for the table).
2. The hypothesis tested by the *p*-value for *CO2* is conditional on *Temp*. This is exactly the hypothesis for *CO2* using the Type I sum of squares above.

Type III sum of squares. Here is the ANOVA table using Type III sum of squares for the urchin data for missing data. The interaction term is excluded from the linear model, and advocates of using Type III sum of squares explicitly want this in the model.

Sum Sq
Df
F value
Pr(>F)
Temp
58.770
1
17.406
0.0006
CO2
19.935
1

5.904
0.0265
Temp:CO2
6.377
1
1.889
0.1872
Residuals
57.399
17

The hypothesis tested by each row in the ANOVA table using Type III sum of squares is the effect of that row's term conditional on all terms in the model.

1. The hypothesis tested by the p -value for $Temp$ is conditional on $CO2$ and $Temp : CO2$.
2. The hypothesis tested by the p -value for $CO2$ is conditional on $Temp$ and $Temp : CO2$.
3. The hypothesis tested by the p -value for $Temp : CO2$ is conditional on $Temp$ and $CO2$. This is the same for Type I sum of squares (and Type II, if the interaction term were included)

16.4.2 Back to interpretation of main effects

16.4.3 The anova tables for Type I, II, and III sum of squares are the same if the design is balanced.

16.5 Working in R

16.5.1 Type I sum of squares in R

The base R function `anova()` computes the ANOVA table using Type I sum of squares for any fit model object, such as that returned by `lm`. Here is a script for the urchin data. I first create unbalanced data by deleting the first row that is the control row.

```

cn_rows <- which(urchin[, Temp]=="Temp-" & urchin[, CO2]=="CO2-") # gives the rows of the control
urchin_unbalanced <- urchin[-cn_rows[1],] # deletes the row that is in first element of cn_rows
urchin.t1 <- lm(Response ~ Temp*CO2, data=urchin_unbalanced)
anova(urchin.t1)

## Analysis of Variance Table
##
## Response: Response
##           Df Sum Sq Mean Sq F value    Pr(>F)
## Temp       1 55.696 55.696 16.9244 0.0005907 ***
## CO2        1 18.411 18.411  5.5946 0.0288072 *
## Temp:CO2   1  9.204  9.204  2.7970 0.1108298
## Residuals 19 62.527  3.291
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

16.5.2 Type II and III Sum of Squares

Type II sum of squares can be computed manually simply by fitting the model twice, once with the factors ordered one way and then with the factors ordered the opposite way. The car package has the function `Anova` that specifically outputs Type II and Type III ANOVA tables.

Type II sum of squares can be fit with the interaction in the model, and this generates the Type II sum of squares for the main terms but the residual is wrong for the F -ratio because it is the residual from the full model and Type II assumes the interaction effect is zero. So, if one wants an ANOVA table with a F and p that reflect this, then the interaction should be dropped from the model.

```

urchin.t2 <- lm(Response ~ Temp*CO2, data=urchin_unbalanced)
Anova(urchin.t2, type="2")

```

```

## Anova Table (Type II tests)
##
## Response: Response
##           Sum Sq Df F value    Pr(>F)
## Temp       52.711  1 16.0173 0.0007624 ***
## CO2        18.411  1  5.5946 0.0288072 *
## Temp:CO2   9.204  1  2.7970 0.1108298
## Residuals 62.527 19
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

```
urchin.t2 <- lm(Resp ~ Temp + CO2, data=urchin_unbalanced)
Anova(urchin.t2, type="2")
```

```
## Anova Table (Type II tests)
##
## Response: Resp
##           Sum Sq Df F value    Pr(>F)
## Temp      52.711  1 14.6968 0.001038 **
## CO2       18.411  1  5.1333 0.034725 *
## Residuals 71.731 20
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

To get type III sum of squares, we need to specify effects coding for the model matrix. The safest way to do this is something like this

```
con3 <- list(Temp=contr.sum, CO2=contr.sum) # change the contrasts coding for the mode
urchin.t3 <- lm(Resp ~ Temp*CO2, data=urchin_unbalanced, contrasts=con3)
Anova(urchin.t3, type="3")
```

```
## Anova Table (Type III tests)
##
## Response: Resp
##           Sum Sq Df F value    Pr(>F)
## (Intercept) 2148.60  1 652.8939 3.559e-16 ***
## Temp        54.71   1 16.6241 0.0006422 ***
## CO2         17.15   1  5.2119 0.0341221 *
## Temp:CO2    9.20   1  2.7970 0.1108298
## Residuals   62.53 19
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Chapter 17

Predictive Models

This chapter focusses on modeling **observational data** with multiple X variables, both continuous and categorical. The classical analysis of multiple X variables is **multiple regression**, sometimes called **multivariable regression** and occasionally, but incorrectly, called **multivariate regression** – “multivariate” refers to multiple Y variables.

The models in this chapter have the structure

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \dots \beta_p X_p + \varepsilon \quad (17.1)$$

% where p is the number of X variables or **predictors** in the model. This equation is easily generalized to both generalized linear models, linear mixed models, and generalized linear mixed models.

17.1 Overfitting

When a model is fit to data, the model coefficients are estimates of the parameters that “generated the data”. The value of an estimate is partly a function of the signal (the parameter) and partly a function of the noise, which is unique to the sample. At a low signal to noise ratio a model is mostly fitting the noise. A measure of how well the model “fits” the data is R^2 , which is

$$R^2 < -1 - \frac{SS_{residual}}{SS_{total}} \quad (17.2)$$

As X variables are added to a model, the R^2 necessarily increases. Part of this increase is due to added signal, but part is due to added noise. If the added noise is more than the added signal, then the model fit – that is the parameter

estimates – increasingly reflects the noise unique to the sample rather than the signal common to every sample. This is the basis of **overfitting**.

To demonstrate overfitting, I fit completely random X variables to the lifespans for the control voles.

Think about it this way: if I create fake data in there are ten X variables that are correlated which Y is a simple column of random, normal variables that are not a function of

17.2 Model building vs. Variable selection vs. Model selection

17.2.1 Stepwise regression

17.2.2 Cross-validation

17.2.3 Penalization

17.2.3.1 AIC

17.2.3.2 LASSO

17.3 Shrinkage

Part VII – Expanding the Linear Model

Chapter 18

Linear mixed models

18.1 Random effects

Researchers often collect data in batches, for example

1. An ecologist interested in the effects of insectivorous birds on tree seedling performance in a forest stake out ten 1 m^2 plots and use a wire-mesh cage to cover half of each plot ¹. The cage allows insect herbivores into the seedlings inside but excludes insectivorous birds that eat the insects from the seedlings. In every plot, five seedlings are planted within the exclosure and five outside of the exclosure. At the end of the experiment, the total leaf mass is measured on each seedling. Small, uncontrolled, environmental factors (including soil factors and density of insectivorous birds) will differ between plots but will be common to all seedlings within a plot and we would expect a common response to this uncontrolled variation on top of the differential response to each treatment. As a consequence, the ten measures of leaf mass within a plot are not independent.
2. A nutrition researcher wants to compare the effect of glucose vs. fructose on glucose metabolism in humans. Ten individuals are recruited. Each individual has blood insulin measured 60 minutes after a noon meal over six successive days. The meal alternates between high glucose and high fructose on each day. Each individual has three measures under high glucose treatment and three measures under high fructose treatment. Small, uncontrolled, environmental factors (including metabolic variation, other meals, activity levels) will differ between the individuals but be common within an individual and we would expect a common response to this uncontrolled variation on top of the differential response to each treatment.

¹Giffard, B., Corcket, E., Barbaro, L., & Jactel, H. (2012). Bird predation enhances tree seedling resistance to insect herbivores in contrasting forest habitats. *Oecologia*, 168(2), 415-424

As a consequence, the six measures of insulin within an individual are not independent.

3. An ecologist wants to measure the effect of an invasive plant on the reproduction of a native plant. They stake-out ten, 2 m^2 plots in a forest and divide each plot into four quadrants, with each quadrant assigned a different treatment: control, activated carbon (a procedural control), extract from the invasive plant's leaves, and both activated carbon and extract from the invasive plant's leaves. The response is seedling count. Small, uncontrolled, environmental factors (including soil, drainage, and light) will differ between plots but will be common to all four quadrants within a plot and we would expect a common response to this uncontrolled variation on top of the differential response to each treatment. As a consequence, the four sets of counts within a plot are not independent.
4. A physiologist has skeletal muscle cells growing in 5 control cultures, and 5 treated cultures. The Y variable is cell diameter, which is measured in 10 cells per culture. Small, uncontrolled, environmental factors (including chemical) will differ between cultures but will be common to all cells within a culture and we would expect a common response to this uncontrolled variation on top of the differential response to each treatment. As a consequence, the ten measures of diameter within a culture are not independent.
5. A behavioral biologist wants to measure the effect of a predator fish on the preferred feeding location (open water or vegetation) of a prey fish. Ten tanks are set up with equal amounts of vegetated and unvegetated area. One-third of each tank is screened off to house a predator fish, which are added to five of the tanks. Ten prey fish are added to each tank. The response is minutes spent foraging in the open water as a fraction of total time foraging, which is measured in each fish in each tank. Small, uncontrolled, environmental factors (including temperature, water chemistry, light, and fish behavior) will differ between the tanks but be common within tanks and we would expect a common response to this uncontrolled variation on top of the differential response to each treatment. As a consequence, the ten measures of foraging of each fish within a tank are not independent.
6. A microbiologist wants to measure the effect of the microbiome on autism-spectrum-disorder(ASD)-like behavior in mice². Intestinal microbial communities from five humans with ASD and five humans without ASD are transferred to germ-free mice via fecal transplant. Each human donor is used to colonize three mice. The response is time socializing in a direct social interaction. Uncontrolled features of each microbial community (species composition and proportions) will differ among human donors but will be the same within human donors and we would expect a common

²Sharon, G., Cruz, N.J., Kang, D.W., Gandal, M.J., Wang, B., Kim, Y.M., Zink, E.M., Casey, C.P., Taylor, B.C., Lane, C.J. and Bramer, L.M., 2019. Human Gut Microbiota from Autism Spectrum Disorder Promote Behavioral Symptoms in Mice. *Cell*, 177(6), pp.1600-1618.

response to this uncontrolled variation in addition to any differential response to ASD-associated microbiota. As a consequence, the measures of behavior in the three mice within a donor group are not independent.

The batches – plots in experiment 1, individuals in experiment 2, plots in experiment 3, cultures in experiment 4, tanks in experiment 5, and mice in experiment 6 – are the experimental units, meaning that it is at this level that the experimenter is controlling the conditions. In each of these studies, there is systematic variation at two levels: among treatments due to treatment effects and among batches due to **batch effects**. This among-batch variation is the **random effect**. An assumption of modeling random effects is that the batches (plots/individuals/cultures/tanks/donor) are a random sample of the batches that could have been sampled. This is often not strictly true as batches are often **convenience samples**.

The multiple measures within a batch are **subsamples** but are often called **repeated measures** if the batch is an individual (experiment 2 is an example). If multiple measures within a treatment level within a batch (that is, within a *batch × treatment* combination) are taken over time, the data are **longitudinal**. Not infrequently, subsamples within a treatment within a batch are called “replicates”, but this can be confusing because the treatments are replicated at the level of the batch and not at the level of the subsamples within a treatment by batch combination. The batches are the independent experimental units. The subsamples within a batch are not replicates.

The variation among batches/lack of independence within batches has different consequences on the uncertainty of the estimate of a treatment effect. The batches in experiments 1-3 are similar in that each contains all treatment levels. In these, the researcher is interested in the treatment effect but not the variation due to differences among the batches. The batches are nuisance factors that add additional variance to the response, with the consequence that estimates of treatment effects are less precise, unless the variance due to the batches is explicitly modeled. In experiments like 1-3, the batches are known as **blocks**. Including block structure in the design is known as **blocking**. Adding a blocking factor to a statistical model is used to increase the precision of an estimated treatment effect.

Experiments 1 and 2 are examples of a **randomized complete block with subsampling** design. “Complete” means that each block has all treatment levels or combinations of levels if there is more than one factor. *The subsampling is not replication*. The replicates are the blocks, because *it was at this level that treatment assignment was randomized*. Experiment 3 is an example of a **randomized complete block** design. The blocks are complete but there is only one measure of the response per treatment.

The batches in experiments 4-6 are similar in that treatment is randomized *to* batch, so each batch contains only a single treatment level. In these **segregated** experimental designs, the variation among batches that arises from non-

treatment related differences among batches **confounds** the variation among batches due to a true treatment effect. An extreme example of this would be experiment 4 (muscle cell cultures) with only a single culture with control conditions and a single culture with treatment conditions. Imagine 1) the true effect of the treatment is zero and 2) an unmeasured growth factor that happens to be more concentrated in the treatment culture at the beginning of the experiment. At the end of the experiment the cells in the treatment culture have an average diameter twice that of that in the control culture. The researcher is fooled into thinking that the treatment caused the increased growth. Again, the replicates are at the level of the cultures, because it was at this level that treatment assignment was randomized. This means the researcher has a single replicate (or, $n = 1$) in each treatment level, regardless of the number of cells that are measured within each culture. A statistical analysis that uses the subsampling within a replicate as the sample size is an example of **pseudoreplication** (Hurlbert 1984 xxx).

18.2 Random effects in statistical models

In all of the above examples, the researcher is interested in the treatment effect but not the variation due to differences among the blocks. The blocks are nuisance factors that add additional variance to the response, with the consequence that estimates of treatment effects are less precise, unless the variance due to the blocks is explicitly modeled. Including block structure in the design and in the statistical model is known as **blocking**. A natural way to think about the block factor is as a **random effect**, meaning that plots in experiment 1 or the mice in experiment 3 are simply random samples from a population of plots or mice. Modeling this using the residual-error specification looks like

$$y_{ij} = (\beta_0 + \beta_{0j}) + (\beta_1 + \beta_{1j})x_i + \varepsilon_i \quad (18.1)$$

where i indexes the observation and j indexes the block (culture, plot, mouse, etc). The intercept parameter β_{0j} is a **random intercept** and the slope parameter β_{1j} is a **random slope**. The intercept for observation i (that is, its expected value when $X = 0$) has a **fixed** component (β_0) that is common to all observations and a random component (β_{0j}) that is common within a block but differs among blocks (see table below). In the above equation, I've used parentheses to show how these combine into the random intercept that is unique for each block. Similarly, the random slope (treatment effect) has a fixed part (β_1) that is common to all observations and a random component (β_{1j}) that is common within a block but differs among blocks (see table below). Again, these are collected within a pair of parentheses in the equation above.

The linear mixed model specified above estimates a fixed intercept and fixed slope (treatment effect) that are common to all observations and a random

intercept and random slope for each block, each of which is common among observations within a block but differ among observations in different blocks.

block

b_0

b_{0j}

b_1

b_{1j}

1

b_0

$b_{0,j=1}$

b_1

$b_{1,j=1}$

2

b_0

$b_{0,j=2}$

b_1

$b_{1,j=2}$

3

b_0

$b_{0,j=3}$

b_1

$b_{1,j=3}$

4

b_0

$b_{0,j=4}$

b_1

$b_{1,j=4}$

5

b_0

$b_{0,j=5}$

b_1

$b_{1,j=5}$

6

b_0

$b_{0,j=6}$

b_1

$b_{1,j=6}$

Linear mixed models are called “mixed models” because they are a mix of fixed and random factors. Another useful way to specify this model is to think about it hierarchically, using

$$y_{ij} = \beta_{0j} + \beta_{1j}x_i + \varepsilon_i \quad (18.2)$$

$$\varepsilon_i \sim N(0, \sigma) \quad (18.3)$$

$$\beta_{0j} = \beta_0 + N(0, \sigma_0) \quad (18.4)$$

$$\beta_{1j} = \beta_1 + N(0, \sigma_1) \quad (18.5)$$

The first line states that the response is a function of a block-specific intercept and a block specific slope plus some error that is unique to each observation. The third and fourth lines state that these block-specific effects are themselves a function of a common effect and a random term that is unique to each block. That is, we have a hierarchical or multi-level structure to the model. Line 1 is the top level and the effects that are specified in line 1 are a function of effects at a second, lower level, which are specified in lines 3 and 4. Because of this structure, linear mixed models are sometimes called hierarchical or multi-level models.

Finally, it’s useful to think how to specify a linear mixed model using the random-draw specification, as this leads naturally to generalized linear mixed models, or GLMMs.

$$y_{ij} \sim N(\mu_{ij}, \sigma) \quad (18.6)$$

$$\mu_{ij} = \beta_{0j} + \beta_{1j}x_i \quad (18.7)$$

$$\beta_{0j} \sim N(\beta_0, \sigma_0) \quad (18.8)$$

$$\beta_{1j} \sim N(\beta_1, \sigma_1) \quad (18.9)$$

18.3 Linear mixed models are flexible

The linear mixed model in Equation (18.1) specifies both a random intercept and a random slope but a researcher might limit the random effect to the inter-

cept only, or less commonly, the slope only. Excluding the random slope from Equation (18.1) results in the model

$$y_{ij} = (\beta_0 + \beta_{0j}) + \beta_1 x_i + \varepsilon_i \quad (18.10)$$

We might use a random-intercept-only model if we think that features of the block would effect the mean response among blocks but not effect the difference in treatment level (or treatment effect) among blocks. For example, differences in the immune systems among the individual mice in experiment 3 might effect growth in both the wild-type and engineered strains of staph but won't effect the difference in growth between wild-type and engineered strains from one mouse to another.

Not more than you should know – For more complex mixed models, matrix algebra makes the specification of the model much more manageable than the scalar algebra in ??.

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u} + \boldsymbol{\varepsilon} \quad (18.11)$$

where \mathbf{y} is the vector of the response, $\mathbf{X}\boldsymbol{\beta}$ is the linear predictor of fixed effects and $\mathbf{Z}\mathbf{u}$ is the linear predictor of random effects. \mathbf{X} is the model matrix for the fixed effects and $\boldsymbol{\beta}$ is the vector of fixed-effect terms (the fixed part of the intercept (β_0), including the fixed-effect coefficients for each of the

18.4 Blocking

18.4.1 Visualizing variation due to blocks

To visualize random effects due to block, Let's create fake data that look something like experiment 1, with a single factor with two treatment levels, $k = 10$ blocks, and $n = 3$ measures for each treatment level within each block. This is a randomized complete block design with subsampling and has a total of $N = 2 \times k \times n$ measures of Y (and rows of the data.table).

Figure 18.1A shows the response as a function of treatment. The responses are nicely symmetric around the treatment means (the blue and yellow lines). A linear model (and generalized linear models, more generally) assumes that a response, conditional on the X , are independent. Figure 18.1B shows how this assumption is violated for the simulated data. That pattern of residuals within a block around the treatment means does not look at all random. Instead, there is a distinct pattern within a block for the points to cluster either below the treatment means or above the treatment means. In blocks a, b, g, and i, all or most of the responses are below their treatment mean (for example in block b, all the yellow points are below the yellow line and two of three blue points are

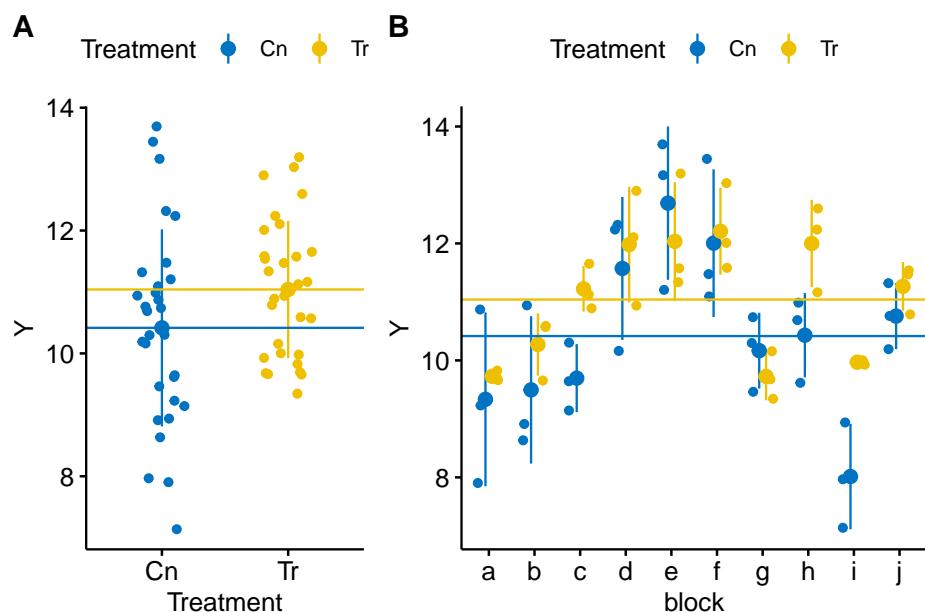


Figure 18.1: Visualizing random effects. A) The response in the two treatment levels. B) The same data but separated by block. The blue line is at the control mean and the yellow line is at the treated mean. The black dots are the mean response within a block.

below the blue line). In blocks d, e, f, and j, all or most of the responses are above their treatment mean (for example, in block e, all three yellow points are above the yellow line and all three blue points are above the blue line). In other words, the responses within a block covary together. For a linear model, this is known as **correlated error**.

18.4.2 Blocking increases precision of point estimates

Block effects are differences in expected mean response among blocks due to unmeasured factors that are shared within blocks but not among blocks. A classical linear model fails to model this component of the total variance in the response, and as a consequence, this block-specific variance is part of the error variance. One way to think about this is by moving the random intercept and random slope components of equation (18.1) to the right and combining it with the observation-specific (or “conditional”) error (ε_i)

$$y_{ij} = \beta_0 + \beta_1 x_i + (\beta_{0j} + \beta_{1j} + \varepsilon_i) \quad (18.12)$$

A linear mixed model estimates the random effects parameters, so the residual from observation i is ε_i . A linear model does not estimate the random effects parameters, so the residual of observation i from a linear model is $\beta_{0j} + \beta_{1j} + \varepsilon_i$. Consequently, the error variance from a linear model is larger than the error variance from a linear mixed model fit to the same data. The consequence of this on inference depends on the variance of the random effects relative to the variance of the observation-specific error and on the subsample size. If the variance due to random effects is relatively big and subsample size is relatively small, then a linear mixed model estimates treatment effects with much more precision (and p -values will be smaller).

This increased precision is seen in the coefficient table of three models fit to the fake data in Figure 18.1.

A linear model fit to all data

term

estimate

std.error

statistic

p.value

conf.low

conf.high

TreatmentTr

0.62441

0.3566

1.75

0.085

-0.089

1.338

A linear model fit to the means of each treatment level with each block term

estimate

std.error

statistic

p.value

conf.low

conf.high

TreatmentTr

0.62441

0.54713

1.14

0.269

-0.525

1.774

A linear mixed model with both a random intercept and random slope fit to all data

term

estimate

std.error

statistic

df

p.value

conf.low

conf.high

TreatmentTr

0.62441

0.26995

2.31

9

0.046

0.095

1.154

Note that the estimates of treatment effect do not differ among models. What does differ is the estimated standard error of the treatment effect, which is 24% smaller in the linear mixed model relative to that in the linear model, and 51% smaller in the linear mixed model relative to that in the linear model fit to the block means. This difference in SE propagates to the confidence intervals and *p*-values.

Let's explore this a wee bit more systematically using a simple, Monte Carlo simulation experiment. 5000 fake data sets were generated. Each data set simulated an experiment with a single treatment factor with two levels ("control" and "treatment"). The design is a randomized complete block with subsampling. There are 8 blocks and 3 subsamples in each treatment level per block. The treatment effect (β_1) is 1. The observation-specific (or "within-block") variance ($\sigma_{varepsilon}^2$) is 1. The ("among-block") variance of the random intercept ($\sigma_{\beta_0j}^2$) is 1 – the same as the the within-block variance. The variance of the random slope ($\sigma_{\beta_1j}^2$), which is due to a variable response of the treatment among blocks, is 0.1².

The following three models were fit to all 5000 data sets

1. a linear model fit to all data, ignoring blocks (violating the independence assumption)
2. a linear model fit to the treatment means of the blocks (valid, but throwing out data)
3. a linear mixed model that models the random intercept³.

From each fit, I saved the 95% confidence interval of the treatment effect and the *p*-value. The median width of the confidence interval and the power, which is the relative frequency of *p*-values less than 0.05. The simulation was re-run using the same parameters except setting the treatment effect (β_1) to 0. In this new simulation, the relative frequency of *p*-values less than 0.05 is the Type I error.

³a random slope is not modeled because many of the models specifying a random slope fail to converge, which is expected given the relatively small variance of the random slope

Model	
CI width	
Power	
Type I	
1. lm	
1.58	
0.75	
0.012	
2. lm of means	
2.39	
0.31	
0.002	
3. lmm b0j	
1.16	
0.92	
0.051	

For data similar to that simulated, the linear mixed model using a blocking factor has much more power than the linear model ignoring block (and ignoring the lack of independence) or the linear model comparing the treatment level means of the blocks. This lost power in the two linear models is due to the conservative Type I error rate (extreme in the case of the linear model of the group means). One take-home lesson here is, don't throw away perfectly good data if the design of the experiment includes a blocking factor!

18.5 Pseudoreplication

18.5.1 Visualizing pseudoreplication

Subsamples from batches are not replicates. Inference from a model fit to subsampled observations without modeling the batches is called pseudoreplication, a term famously coined by Hurlbert (1984). For data from a randomized complete block design, ignoring the batches in the model will typically result in larger standard errors, wider confidence intervals, and too conservative p -values. In a segregated experiment, with only a single treatment level per batch (like that in experiments 4-6 above), ignoring the lack of independence in the model has the opposite effect. Standard errors are too small. Confidence intervals are too narrow. P -values are too liberal. Type I error is inflated. Let's visualize a pseudoreplicated data set like this.

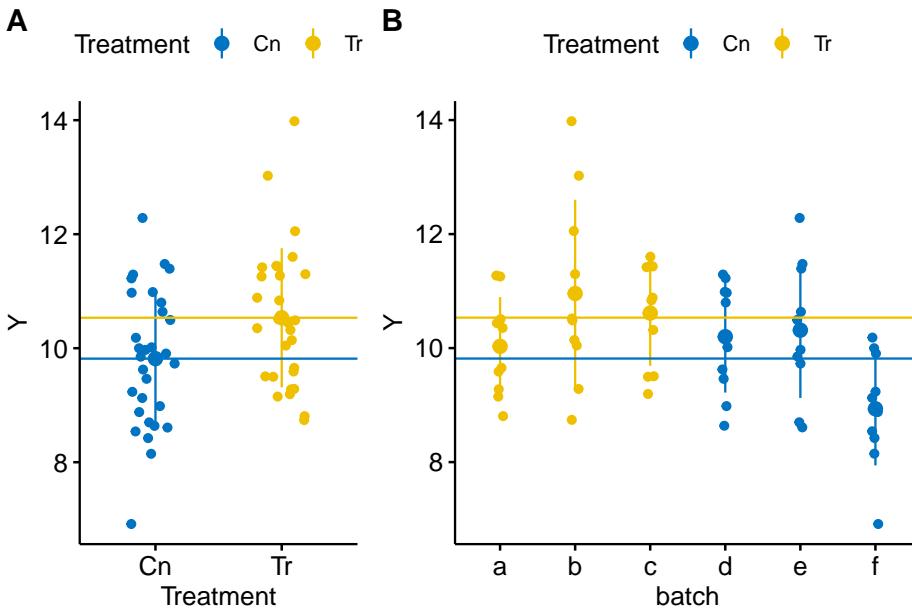


Figure 18.2: A data set in which treatment and batch are confounded because there is only one treatment level per batch.

This consequence of pseudoreplication when batch and treatment are confounded is seen in the coefficient table of four models fit to the fake data in Figure 18.2.

A linear model fit to all data (pseudoreplication).

```
term
estimate
std.error
statistic
p.value
conf.low
conf.high
TreatmentTr
0.7175
0.31278
2.29
0.025
```

0.091

1.344

A linear model fit the batch means.

term

estimate

std.error

statistic

p.value

conf.low

conf.high

TreatmentTr

0.7175

0.51803

1.39

0.238

-0.721

2.156

A linear mixed model with both random intercept and random slope.

term

estimate

std.error

statistic

df

p.value

conf.low

conf.high

TreatmentTr

0.7175

0.51797

1.39

3.32

```
0.252
-0.298
1.733

A linear mixed model with random intercept.
term
estimate
std.error
statistic
df
p.value
conf.low
conf.high
TreatmentTr
0.7175
0.51803
1.39
4
0.238
-0.298
1.733
```

As with the blocked design, all models compute the same estimate of the treatment effect. But in contrast to the blocked design, in this confounded design, the standard error of the treatment effect is smaller in the linear model of all data than that of the linear mixed models. This increased precision is an illusion because the model fails to account for the lack of independence within the batches.

One interesting result to note is the equivalence of the standard error, test statistic, *p*-value, and confidence intervals of the linear model on the batch means and the linear mixed model with a random intercept (but no random slope) only. The two are equivalent. Murtaugh (xxx) has argued that with this kind of design, it is much simpler to the analyst, and to your audience, to simply use the linear model on the batch means. This raises the question, why bother with subsampling within batch? One answer is, subsampling increases the precision of the batch mean, and therefore, the precision of the treatment effect. That said, *the precision of a treatment effect is increased more efficiently by adding more replicates (more batches), not more subsamples.*

Let's explore the consequence of pseudoreplication with a confounded with a Monte Carlo simulation experiment. 5000 fake data sets were generated. Each data set simulated an experiment with a single treatment factor with two levels ("control" and "treatment"). The design is treatment level randomized to batch (only one treatment level per batch). There are 8 batches and 3 subsamples in each batch. The treatment effect (β_1) is zero. The observation-specific (or "within-block") variance ($\sigma_{varepsilon}^2$) is 1. The ("among-block") variance of the random intercept ($\sigma_{\beta_0j}^2$) is 1 – the same as the the within-block variance. The variance variance of the random slope ($\sigma_{\beta_1j}^2$), which is due to a variable response of the treatment among blocks, is 0.1²).

Because the effect is zero, the frequency of p -values less than 0.05 is the type I error. The same three models fit to the simulated blocked data are fit to these data. I don't simulate an effect in order to compute power (at that effect size) because the increased power in the linear model of all data is, again, an illusion. It only comes at the cost of strongly elevated Type I error.

Model

CI width

Type I

- | | |
|----------------|--|
| 1. lm | |
| 2.22 | |
| 0.18 | |
| 2. lm of means | |
| 3.80 | |
| 0.05 | |
| 3. lmm b0j | |
| 3.80 | |
| 0.05 | |

When treatment level is randomized *to* batch, the type I error rate of a linear model fit to all data (and ignoring the lack of independence) is highly inflated. Don't do this.

18.6 Mapping NHST to estimation: A paired t-test is a special case of a linear mixed model

Specifically, a **paired t-test** is equivalent to a linear mixed model with a single factor with two treatment levels, k blocks, and a single measure of each treatment level within each block. A good example is the wild type vs. engineered staph count in mice in experiment 3 above. A linear mixed model is much more

flexible than a paired t -test because it allows a researcher to add treatment levels, additional factors, and covariates to the model. In addition, a linear mixed model can handle missing data.

Here is fake data similar in design to experiment 3 with a single factor with two treatment levels and both levels applied to the same experimental unit.

```
set.seed(2)
n <- 10 # number of mice (blocks)
x <- rep(c("WT", "Tr"), each=n) # treatments
id <- rep(letters[1:n], 2) # block id
y <- c(rnorm(n, mean=10), rnorm(n, mean=11))
fake_data <- data.table(Y=y, X=x, ID=id)
```

The t -test p -value is

```
t.test(Y~X, data=fake_data, paired=TRUE)$p.value
```

```
## [1] 0.05336815
```

and the coefficient table of the fixed effect in the linear mixed model is

```
coef(summary(lme(Y~X, random = ~1|ID, correlation=corCompSymm(form=~1|ID), data=fake_data)))
```

	Value	Std.Error	DF	t-value	p-value
## (Intercept)	11.1797704	0.3438775	9	32.510914	1.212113e-10
## XWT	-0.9686188	0.4358740	9	-2.222245	5.336815e-02

18.7 Advanced topic – Linear mixed models shrink coefficients by partial pooling

In experiment 1 above, there are 10 sites (maybe different woodlots). In each plot, five seedlings are planted inside a cage and five outside the cage. The cage excludes insectivorous birds but not herbivorous insects. The researchers are investigating how birds affect plant growth indirectly – by eating insects that feed on the plants. The response is total leaf area in each seedling.

Let's say we want to know the treatment effect in each of these sites. There are several ways of estimating this.

1. Fit k separate models, one for each site. The intercept (control mean) and slope (treatment effect) parameters for each site are estimated independently from all other sites. Consequently, the model parameters are

computed using **no pooling**. For the estimation of the β terms, this is equivalent to a single, factorial linear model with *Site* modeled as a **fixed effect** (this is not true for the estimate of the standard errors of these terms since these are computed from the residual sum of squares of the model. For balanced data, all of the “intercept” or “slope” terms will have the same SE in the factorial analysis but differ among the k independent analyses).

2. Fit a linear model to all the data combined as if these were from a single site, and assign the intercept and treatment effect parameters to all sites. The model parameters are computed using **complete pooling**.
3. Fit a linear mixed model to all the data, using site as a random factor to estimate both random intercepts and slopes. Similar to the no-pooling analysis, there will be different intercept (control mean) and slope (treatment effect) estimates for each site, but unlike the no-pooling analysis, these estimates are computed by combining information from the other sites. The information used to estimate parameters in a linear mixed model is somewhere in between no pooling and complete pooling and is sometimes called **partial pooling**.

The consequence of partial pooling in a linear mixed model is that site intercepts (control means) are pulled toward the single intercept in the complete-pooling analysis and the site slopes (treatment effects) are pulled toward the single slope in the complete-pooling analysis. This has the consequence that the **differences** in parameter estimates among sites are shrunk toward zero. A consequence of this shrinkage is that the variance of the intercept estimates or of the slope estimates is smaller than that in the no-pooling analysis. Figure 18.3 shows this shrinkage effect using fake data simulating the seedling experiment.

The linear mixed model estimates of the treatment effects for each site are a type of **shrinkage estimate** and a linear mixed model is one kind of **shrinkage estimator**. Shrinkage estimates have fascinating properties:

1. the variance of shrinkage estimates is less than that of ordinary least squares estimates (no-pooling, or using the block as a fixed factor)
2. shrinkage estimates are **biased** but the OLS estimates are not. This means that the expected value of a coefficient from the linear mixed model *does not equal* the true (parameter) value! Or, more formally, $E(b_j) \neq \beta_j$.
3. the **mean square error** of shrinkage estimates will be smaller than that for OLS estimates.

The first property was discussed above and shown in Figure 18.3. The second property raises the question, if we want to estimate the treatment effects within each site, why would we ever want to use *Site* as a random instead of fixed effect? The answer is the third property, which can be summarized as, “if we

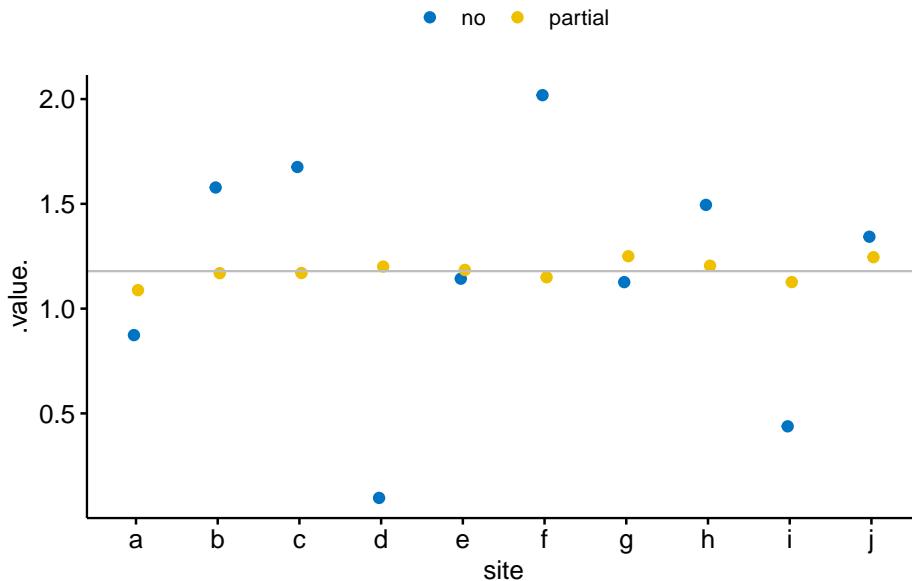


Figure 18.3: Shrinkage estimates of the treatment effect in a linear mixed model. The grey line is the estimate using complete pooling (so there is only one estimate which is assigned to each site). In general, the partial-pooling (linear mixed model) estimates (yellow) are generally closer to the complete pooling estimate than the no-pooling (separate linear models) estimates (blue). More specifically, if the no-pooling estimate is far from the complete pooling estimate, the partial pooling estimate is *much* closer to the complete pooling estimate. The consequence of partial pooling is that the differences among the estimates are shrunk toward zero.

were to replicate the experiment many times, the shrinkage estimates will be, on average, less wrong (or closer to the true value) than the OLS estimates, where "wrong" is the absolute deviation from the true value."

When shrinkage estimators were first discovered, the third property surprised statisticians. The third property has profound consequences. Consider a scenario where researchers want to compare the performance of a new expression vector to that of an existing expression vector on protein production using *E. coli*. The researchers have ten different *E. coli* strains and are interested in strain-specific effects because they will choose the three strains with the largest effects for further testing. The researchers measure the response of each strain five times.

Effect of new expression vector on protein production in ten strains of *E. coli* using a fixed effect factorial model and linear mixed model.

Strain

β_{1j}

fixed b_{1j}

random b_{1j}

a

0.91

1.07

0.98

b

0.87

0.94

0.85

c

0.90

-1.03

0.30

d

0.81

0.64

0.63

e

1.09

1.00

1.07

f

0.62

0.91

1.14

g

1.33

2.26

1.36

h

1.27

1.48

0.96

i

1.61

0.57

1.13

j

0.89

1.50

0.93

The table above shows the true strain-specific effect and both the fixed (OLS) and random (LMM) effect estimates. The largest OLS estimate is 70% larger than the true effect and the strain with the largest true effect is not among the top three biggest OLS estimates (its ranking is 9/10). By contrast, the LMM estimates are closer to the true effects and the top strain is among the three largest LMM estimates.

These results are specific to these fake data but more generally, 1) the largest OLS estimates are inflated (larger error from the true effect), relative to the largest LMM estimates 2) overall, the LMM estimates will be closer than the OLS estimates to the true effects

To understand this, rank order the treatment effects for each strain. An individual strain's position in this rank is the sum of the true effect for that strain and

some random error. Because OLS, relative to shrinkage estimates, have greater variance in the estimate (that is, the random error component is bigger), the biggest effects estimated by OLS are more likely to be big because of the error component, compared to shrinkage estimates.

Not more than you want to know – Shrinkage estimators are not only useful when we are interested in block-specific effects but are also useful for estimating effects when there are **multiple responses**. For example, consider a researcher interested in measuring the effects of some exercise treatment on gene expression in adipose cells. The researcher measures expression levels in 10,000 genes. Given the typical content in undergraduate biostatistics courses, a researcher would probably model these responses using 10,000 *t*-tests, or equivalently, 10,000 separate linear models. If the tests were ranked by *p*-value or absolute effect size, many of the genes with largest absolute effect would be there because of a large error component and many of the largest effects would be massively overinflated. Re-imagining the design as a single, linear mixed model with each gene modeled as a block would lead to a rank order in which the biggest measured effects more closely approximate the true effects.

18.8 Working in R

The major function for working with linear mixed models is `lmer()` from the `lme4` package. An older, and still sometimes used and useful function is `lme()` from the `nlme` package. The authors of the `lme4` package argue that the df in a linear mixed model are too approximate for a useful *p*-value and, consequently, the `lme` function does not return a *p*-value. Many biological researchers want a *p*-value and typically use the `lmerTest` package to get this.

Specifying a linear mixed model using `lme`. The random factor is in the column “block”. To conform to some packages that use `lme4` objects, any variable used to model a random effect should be converted to type `factor`.

1. add a random intercept using `y ~ treatment + (1|block)`. This adds a random intercept for each level of treatment.
2. add a random slope and intercept using `y ~ treatment + (treatment|block)`. This adds a random intercept for each level of treatment and a random slope for each level of treatment.

A message that might appear is “boundary (singular) fit: see `?isSingular`”. This does not mean there is a problem with the fit model.

A warning that “Model failed to converge with 1 negative eigenvalue” does mean there is a problem. A solution is to simplify the model by, for example, removing a random slope.

18.8.1 coral data

Source Zill, J. A., Gil, M. A., & Osenberg, C. W. (2017). When environmental factors become stressors: interactive effects of vermetid gastropods and sedimentation on corals. *Biology letters*, 13(3), 20160957.

Dryad source <https://datadryad.org/resource/doi:10.5061/dryad.p59n8>

file name “VermetidSedimentData_ZillGilOsenberg_DRYAD.xlsx”

```
folder <- "Data from When environmental factors become stressors- interactive effects of vermetids"
fn <- "VermetidSedimentData_ZillGilOsenberg_DRYAD.xlsx"
sheet_i <- "Coral Growth Rate Data"
file_path <- here(data_path, folder, fn)
coral <- read_excel(file_path, sheet=sheet_i) %>%
  clean_names() %>%
  data.table()
coral[, vermetids:=factor(vermetids)]
coral[, sediment:=factor(sediment)]

# recode levels of factors since 0 and 1
coral[, vermetids := fct_recode(vermetids,
                                   absent = "0",
                                   present = "1")]
coral[, sediment := fct_recode(sediment,
                               control = "0",
                               addition = "1")]
```

18.8.1.1 Fitting models

```
# to reproduce the results
# observation 2 should be excluded from the analysis
inc <- c(1, 3:nrow(coral))

# random intercept only
m1 <- lmer(growth_rate ~ vermetids*sediment + (1|block), data=coral[inc])
# random intercept and slope
m2 <- lmer(growth_rate ~ vermetids*sediment + (vermetids|block) + (sediment|block), data=coral[inc])
# to include the interaction as a random effect we'd need subsampling within each factorial treatment
```

The conditional effects of m1 are

```
# results using lmer fit
fit.emm <- emmeans(m1, specs=c("vermetids", "sediment"))
```

```

summary(contrast(fit.emm,
                  method = "revpairwise",
                  simple = "each",
                  combine = TRUE,
                  adjust="none"),
      infer=c(TRUE, TRUE))

##  sediment vermetids contrast      estimate    SE   df lower.CL
##  control .      present - absent  0.00466 0.209 23.0  -0.428
##  addition .      present - absent -0.76889 0.217 23.6  -1.217
##  .      absent     addition - control  0.28520 0.217 23.6  -0.163
##  .      present    addition - control -0.48834 0.209 23.0  -0.921
##  upper.CL t.ratio p.value
##  0.4373  0.022  0.9824
##  -0.3211 -3.547  0.0017
##  0.7330  1.316  0.2009
##  -0.0557 -2.335  0.0286
##
## Degrees-of-freedom method: kenward-roger
## Confidence level used: 0.95

```

There is no “correct” way to compute the degrees of freedom for inferential statistics (SE, CIs, *p*-values). Two common choices are “Kenward-Roger” and “Satterthwaite”. There is little empirical reason to vastly prefer one over the other (they each seem to perform a wee bit better under different conditions).

In `emmeans`, the Kenward-Roger degrees of freedom are the default. For Satterthwaite, use the `lmer.df` argument:

```

fit.emm <- emmeans(m1,
                     specs=c("vermetids", "sediment"),
                     lmer.df = "satterthwaite")
summary(contrast(fit.emm,
                  method = "revpairwise",
                  simple = "each",
                  combine = TRUE,
                  adjust="none"),
      infer=c(TRUE, TRUE))

##  sediment vermetids contrast      estimate    SE   df lower.CL
##  control .      present - absent  0.00466 0.209 22.9  -0.428
##  addition .      present - absent -0.76889 0.216 23.5  -1.215
##  .      absent     addition - control  0.28520 0.216 23.5  -0.161
##  .      present    addition - control -0.48834 0.209 22.9  -0.921
##  upper.CL t.ratio p.value

```

```

##      0.4374  0.022  0.9824
##     -0.3226 -3.559  0.0016
##      0.7315  1.320  0.1994
##     -0.0556 -2.335  0.0287
##
## Degrees-of-freedom method: satterthwaite
## Confidence level used: 0.95

```

If you want to compute the coefficient table or an ANOVA, lmer does not output test statistics. To get test statistics, you have to load the library “lmerTest” (which automatically loads “lme4”). With `lmerTest`, the Satterthwaite degrees of freedom are the default. For Kenward-Roger, use the `ddf` argument in either `summary()` (for the coefficients) or `anova()` (for ANOVA).

```
coef(summary(m1)) # default is Satterthwaite
```

	Estimate	Std. Error	df
## (Intercept)	1.268411111	0.1541680	30.42768
## vermetidspresent	0.004655556	0.2091398	22.94243
## sedimentaddition	0.285202305	0.2160126	23.53130
## vermetidspresent:sedimentaddition	-0.773546750	0.3006674	23.24638
##	t value	Pr(> t)	
## (Intercept)	8.22745788	3.129900e-09	
## vermetidspresent	0.02226049	9.824326e-01	
## sedimentaddition	1.32030428	1.994327e-01	
## vermetidspresent:sedimentaddition	-2.57276556	1.693404e-02	

```
coef(summary(m1, ddf="Kenward-Roger"))
```

	Estimate	Std. Error	df
## (Intercept)	1.268411111	0.1541680	30.43671
## vermetidspresent	0.004655556	0.2091398	23.03604
## sedimentaddition	0.285202305	0.2167687	23.62024
## vermetidspresent:sedimentaddition	-0.773546750	0.3012111	23.33762
##	t value	Pr(> t)	
## (Intercept)	8.22745788	3.122797e-09	
## vermetidspresent	0.02226049	9.824319e-01	
## sedimentaddition	1.31569876	2.009032e-01	
## vermetidspresent:sedimentaddition	-2.56812158	1.708128e-02	

Compare the output from `emmeans` and `coef(summary())` using the different methods for computing the df.

Chapter 19

Generalized linear models I: Count data

Biologists frequently count stuff, and design experiments to estimate the effects of different factors on these counts. For example, the effects of environmental mercury on clutch size in a bird, the effects of warming on parasite load in a fish, or the effect of exercise on RNA expression.

Count data differ from data with normal error in many ways, including 1) counts are discrete, and can be zero or positive integers only, 2) counts tend to bunch up on the small side of the range, creating a distribution with a positive skew, 3) a sample of counts can have an abundance of zeros, and 4) the variance of counts increases with the mean (see Figure 19.1 for some of these properties). Some count data can be approximated by a normal distribution and reasonably modeled with a linear model but more often, count data are modeled with **Poisson distribution** or **negative binomial distribution** using a **generalized linear model** (GLM). Poisson and negative binomial distributions are **discrete probability distributions** with two important properties: 1) the distribution contains only zero and positive integers and 2) the variance is a function of the mean. Back before modern computing and fast processors, count data were often analyzed by either **transforming** the response or by **non-parametric hypothesis tests**. One reason to prefer a statistical modeling approach with a GLM is that we can get interpretable parameter estimates. By contrast, both the analysis of transformed data and non-parametric hypothesis tests are really tools for computing “correct” p -values.

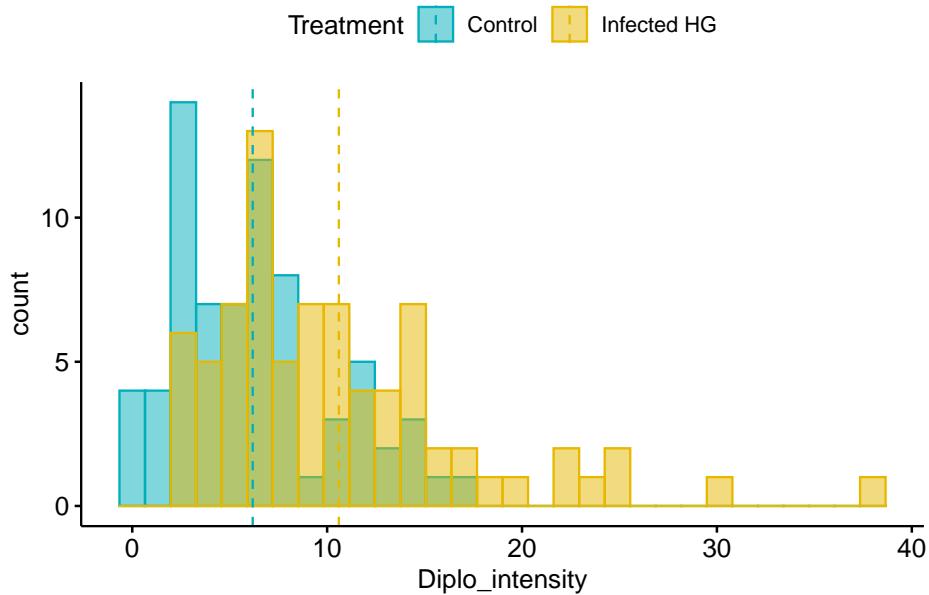


Figure 19.1: Histogram of the count of a trematode parasite larvae in Control vs. Infected fish. Fish in the Infected treatment are infected with a tapeworm.

19.1 The generalized linear model

As outlined in section [Assumptions for inference with statistical models] in Chapter 1, a common way that biological researchers think about a response variable is

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i \quad (19.1)$$

$$\varepsilon \sim N(0, \sigma) \quad (19.2)$$

That is, we can think of a response as the sum of some systematic part and “random error”, which is a random draw from a normal distribution with mean zero and variance σ^2 . This way of thinking about the generation of the response is useful for linear models, and model checking linear models, but is not useful for generalized linear models or model checking generalized liner models. For example, if we want to model the number of parasites that have infected a fish using a Poisson distribution, the following is the **wrong** way to think about the statistical model

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i \quad (19.3)$$

$$\varepsilon \sim Poisson(\lambda) \quad (19.4)$$

That is, we should not think of a count as the sum of a systematic part and a random draw from a Poisson distribution. Why? Because it is the counts (or the counts conditional on X) that are poisson distributed, not the residuals from the fit model.

Thinking about the distribution of count data using model (19.4) leads to absurd consequences. For example, if we set the mean of the Poisson “error” to zero (like with a normal distribution), then the error term for every observation would *have to* be zero (because the only way to get a mean of zero with non-negative integers is if every value is zero). Or, if the study is modeling the effect of a treatment on the counts (that is, the X are dummy variables) then β_0 is the expected mean count of the control (or reference) group. But if we add non-zero Poisson error to this, then the mean of the control group would be larger than β_0 . This doesn’t make sense. And finally, equation (19.4) generates a continuous response, instead of an integer, because β_0 and β_1 are continuous.

A better way to think about the data generation for a linear model, because this naturally leads to the *correct* way to think about data generation for a generalized linear model, is

$$y_i \sim N(\mu_i, \sigma) \quad (19.5)$$

$$\text{E}(Y|X) = \mu \quad (19.6)$$

$$\mu_i = \beta_0 + \beta_1 x_i \quad (19.7)$$

That is, a response is a random draw from a normal distribution with mean μ (not zero!) and variance σ^2 . Line 1 is the stochastic part of this specification. Line 3 is the systematic part.

The specification of a generalized linear model has both stochastic and systematic parts but adds a third part, which is a **link function** connecting the stochastic and systematic parts.

1. **The stochastic part**, which is a probability distribution from the exponential family (this is sometimes called the “random part”)

$$y_i \sim \text{Prob}(\mu_i) \quad (19.8)$$

2. **the systematic part**, which is a linear predictor (I like to think about this as the deterministic part)

$$\eta = \mathbf{X}\boldsymbol{\beta} \quad (19.9)$$

3. a **link function** connecting the two parts

$$\eta_i = g(\mu_i) \quad (19.10)$$

μ (the Greek symbol mu) is the conditional mean (or expectation $E(Y|X)$) of the response on the **response scale** and η (the Greek symbol eta) is the conditional mean of the response on the **link scale**. A GLM models the response with a distribution specified in the stochastic part. The probability distributions introduced in this chapter are the Poisson and Negative Binomial. The natural link function for the Poisson and Negative Binomial is the “log link”, $\eta = \log(\mu)$. More generally, while each distribution has a natural (or, “canonical”) link function, one can use alternatives. Given this definition of a generalized linear model, a linear model is a GLM with a normal distribution and an Identity link ($\eta = \mu$).

When modeling counts using the Poisson or negative binomial distributions with a log link, the link scale is linear, and so the effects are additive on the link scale, while the response scale is nonlinear (it is the exponent of the link scale), and so the effects are multiplicative on the response scale. If this doesn’t make sense now, an example is worked out below. The inverse of the link function backtransforms the parameters from the link scale back to the response scale. So, for example, a prediction on the response scale is $\exp(\hat{\eta})$ and a coefficient on the response scale is $\exp(b_j)$.

19.2 Count data example – number of trematode worm larvae in eyes of threespine stickleback fish

The example is an experiment measuring the effect of the parasitic tapeworm *Schistocephalus solidus* infection on the susceptibility of infection from a second parasite, the trematode *Diplostomum pseudospathaceum*, in the threespine stickleback fish *Gasterosteus aculeatus*¹. The treatment levels are “Control” (unexposed to the tapeworm), “Uninfected” (exposed to the tapeworm but uninfected), “Infected LG” (exposed and infected with the low growth population of the tapeworm), and “Infected HG” (exposed and infected with the high growth population of tapeworm). The response is the number of trematode larvae counted in the eyes (right and left combined) of the fish. A histogram of the counts is shown in Figure 19.1 for the control and Infected HG treatment levels.

19.2.1 Modeling strategy

NHST blues – Students are often encouraged by textbooks, colleagues, or the literature to start the analysis by first “testing” assumptions with hypothesis

¹Benesh, D. P., & Kalbe, M. (2016). Experimental parasite community ecology: intraspecific variation in a large tapeworm affects community assembly. *Journal of Animal Ecology*, 85(4), 1004-1013

19.2. COUNT DATA EXAMPLE – NUMBER OF TREMATODE WORM LARVAE IN EYES OF THREE-SPINE STickleBACKS

tests – for example using a Shapiro-Wilks test of normality as a decision rule to decide if to use a parametric test such as a *t*-test or ANOVA if the null hypothesis of normality is not rejected, or a non-parametric test such as a Mann-Whitney U test if the null hypothesis of normality is rejected. I advise against this, because 1) this pre-test filtering automatically invalidates the *p*-value of the hypothesis test as it does not adjust for the filtering procedure, 2) real data are only approximately normal and as *n* increases, a normality test will reject any real dataset, and 3) hypothesis tests are pretty robust to non-normality anyway.

Instead of testing assumptions of a model using formal hypothesis tests *before* fitting the model, a better strategy is to 1) fit a model, and then do 2) **model checking** using **diagnostic plots**, diagnostic statistics, and simulation.

With these data, a researcher would typically fit a GLM with a Poisson or negative binomial distribution and log link. Here, I start with a linear model to illustrate the interpretation of diagnostic plots with non-normal data. I use the “linear model” specification (equation (19.2)) because the diagnostic plots for model checking a linear model use the residuals of the fit model.

$$Diplo_intensity_i = \beta_0 + \beta_1 Uninfected_i + \beta_2 Infected_LG_i + \beta_3 Infected_HG_i + \varepsilon_i \quad (19.11)$$

$$\varepsilon \sim N(0, \sigma) \quad (19.12)$$

19.2.2 Checking the model I – a Normal Q-Q plot

Figure 19.2A shows a histogram of the residuals from the fit linear model. The plot shows that the residuals are clumped at the negative end of the range, which suggests that a model with a normally distributed conditional outcome (or normal error) is not well approximated.

A better way to investigate this is with the **Normal Q-Q** plot in Figure 19.2B, which plots the sample quantiles for a variable against their theoretical quantiles. If the conditional outcome approximates a normal distribution, the points should roughly follow the line. Instead, for the worm data, the points are above the line at both ends. At the left (negative) end, this means that we aren’t seeing the most negative values that would be expected (the observed values are more positive than the theoretical values). Remembering that this plot is of residuals, if we think about this as counts, this means that our smallest counts are not as small as we would expect given the mean and a normal distribution. This shouldn’t be surprising – the counts range down to zero and counts cannot be below zero. At the positive end, the sample values are again more positive than the theoretical values. Thinking about this as counts, this means that the largest counts are larger than expected given the mean and a normal distribution. This pattern is exactly what we’d expect of count data, or at least count data that borders zero.

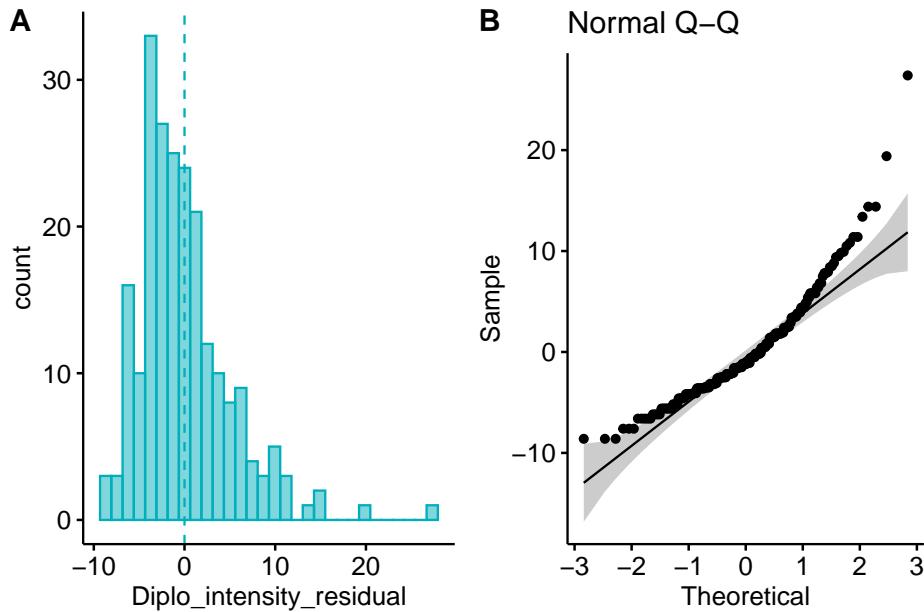


Figure 19.2: Diagnostic plots of stickleback parasite data. A) Distribution of the residuals of the fit linear model. B) Normal Q-Q plot of the residuals of the fit linear model.

Intuition Pump – Let’s construct a Normal Q-Q plot. A **quantile** (or percentile) of a vector of numbers is the value of the point at a specified percentage rank. The median is the 50% quantile. The 95% confidence intervals are at the 2.5% and 97.5% quantiles. In a Normal Q-Q plot, we want to plot the quantiles of the residuals against a set of theoretical quantiles.

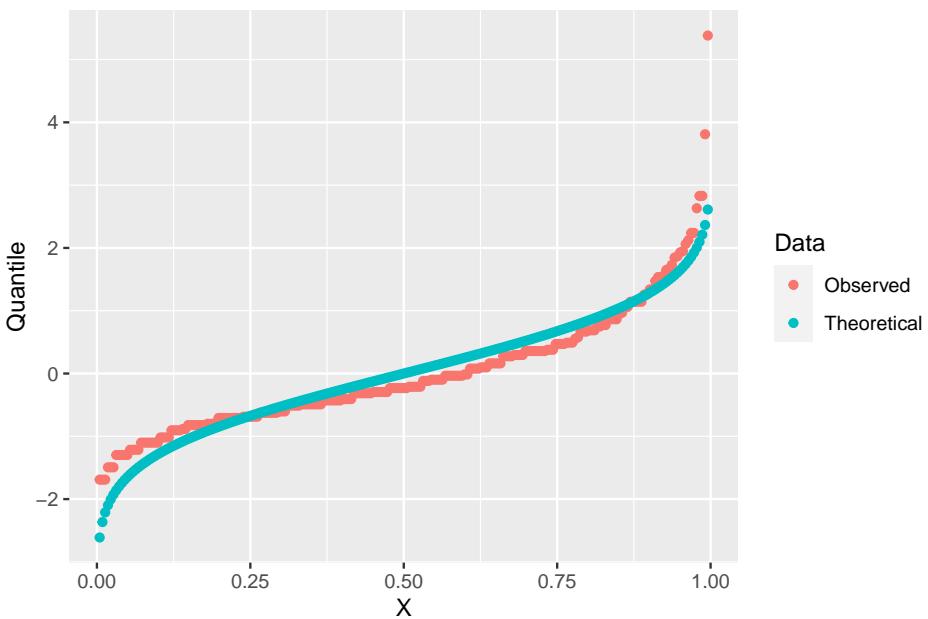
1. To get the observed quantiles, rank the residuals of the fit linear model from most negative to most positive – these are your quantiles! For example, if you have $n = 145$ residuals, then the 73rd point is the 50% quantile.
2. A theoretical quantile from the normal distribution can be constructed using the `qnorm` function which returns the normal quantiles for a specified vector of percents. Alternatively, one could randomly sample n points using `rnorm`. These of course will be sampled quantiles so will only approximate the expected theoretical quantiles, but I add this here because we use this method below.

Now simply plot the observed against theoretical quantiles. Often, the **standardized** quantiles are plotted. A standardized variable has a mean of zero and a standard deviation of one and is computed by 1) centering the vector at zero by subtracting the mean from every value, and 2) dividing each value

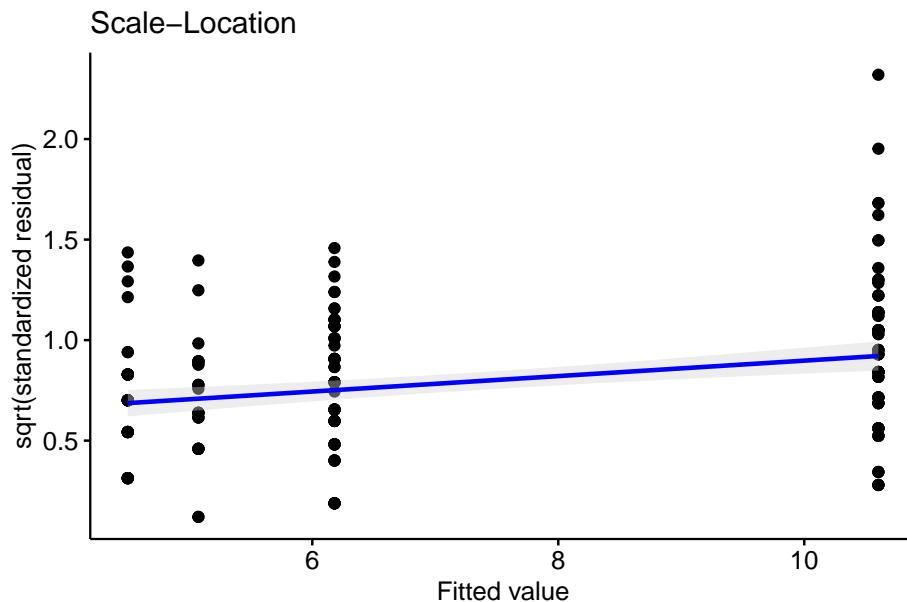
19.2. COUNT DATA EXAMPLE – NUMBER OF TREMATODE WORM LARVAE IN EYES OF THREE-SPINE STickleBACKS

by the standard deviation of the vector. Recognize that because a standard deviation is a function of deviations from the mean, it doesn't matter which of these operations is done first. A standardized theoretical quantile is specified by `qnorm(p, mean = 0, sd = 1)`, which is the default.

Below, I've plotted the standardized observed and theoretical quantiles against the vector of percents (from 0 to 100%). This plot also nicely shows how the residuals of the worm data deviate from that expected if these had a normal distribution. The plot nicely shows that the most negative observed quintiles are not as negative as expected given a normal distribution, which again makes sense because this would imply negative counts since the mean is close to zero. And it nicely shows that the most positive observed quantiles are more positive than expected given a normal distribution, again this makes sense in right skewed count data. Finally, the plot nicely shows that the median is less positive than that expected given a normal distribution, which is at the mean (a right skew tends to pull the mean to the right of the median).



19.2.3 Checking the model II – scale-location plot for checking homoskedasticity



A linear model also assumes the error has constant variance (that is, the error variance is not a function of the value of X), or homoskedasticity. The fit model can be checked for homoskedasticity using a scale-location plot, which is a scatterplot of the positive square-root of the standardized residuals against the fitted values². If the residuals approximate a normal distribution, then a regression line through the scatter should be close to horizontal. The regression line in the scale-location plot of the fit of the linear model to the worm data shows a distinct increase in the “scale” (the square root of the standardized residuals) with increased fitted value, which is expected of data that are lognormally, Poisson, or negative binomially distributed.

19.2.4 Two distributions for count data – Poisson and Negative Binomial

The pattern in the normal Q-Q plot in Figure 19.2B should discourage one from modeling the data with a normal distribution and instead model the data with an alternative distribution using a Generalized Linear Model. There is no unique mapping between how data are generated and a specific distribution, so this decision is not as easy as thinking about the data generation mechanism and then simply choosing the “correct” distribution. Section 4.5 in Bolker (xxx) is an

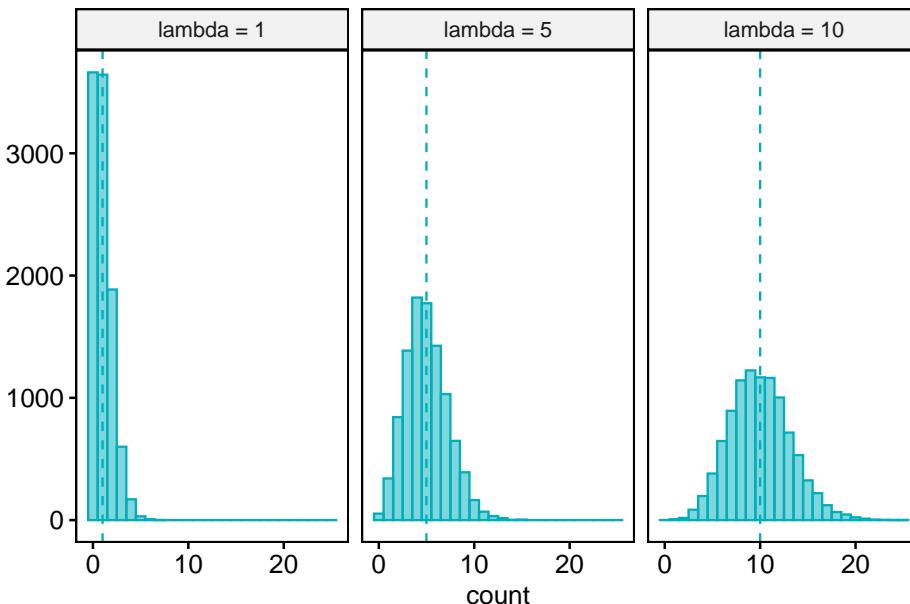
²fitted values are the predicted values, \hat{Y}

19.2. COUNT DATA EXAMPLE – NUMBER OF TREMATODE WORM LARVAE IN EYES OF THREE-SPINE STicklebacks

excellent summary of how to think about the generating processes for different distributions in the context of ecological data. Since the response in the worm data are counts, we need to choose a distribution that generates integer values, such as the Poisson or the negative binomial.

1. Poisson – A Poisson distribution is the probability distribution of the number of occurrences of some thing (an egg, a parasite, or a specific mRNA transcript) generated by a process that generates the thing at a constant rate per unit effort (duration or space). This constant rate is λ , which is the expectation, so $E(Y) = \mu = \lambda$. Because the rate per effort is constant, *the variance of a Poisson variable equals the mean*, $\sigma^2 = \mu = \lambda$. Figure ?? shows three samples from a Poisson distribution with λ set to 1, 5, and 10. The plots show that, as the mean count (λ) moves away from zero, a Poisson distribution 1) becomes less skewed and more closely approximates a normal distribution and 2) has an increasingly low probability of including zero (less than 1% zeros when the mean is 5).

A Poisson distribution, then, is useful for count data in which the conditional variance is close to the conditional mean. Very often, biological count data are not well approximated by a Poisson distribution because the variance is either less than the mean, an example of **underdispersion**³, or greater than the mean, an example of **overdispersion**⁴. A useful distribution for count data with overdispersion is the negative binomial.



³the variance is less than that expected by the probability model

⁴the variance is greater than that expected by the probability model

2. Negative Binomial – The negative binomial distribution is a discrete probability distribution of the number of successes that occur before a specified number of failures k given a probability p of success. This isn't a very useful way of thinking about modeling count data in biology. What is useful is that the Negative Binomial distribution can be used simply as a way of modeling an “overdispersed” Poisson process. The mean of a negative binomial variable is $\mu = k \frac{p}{1-p}$ and the variance is $\sigma^2 = \mu + \mu^2/k$. As a method for modeling an overdispersed Poisson variable, k functions as a **dispersion parameter* controlling the amount of overdispersion and can be any real, positive value (not simply a positive integer), including values less than 1.

19.2.5 Fitting a GLM with a Poisson distribution to the worm data

Let's fit a GLM with a Poisson distribution to the worm data. The model is

$$Diplo_intensity_i \sim Poisson(\mu_i) \quad (19.13)$$

$$E(Diplo_intensity|Treatment) = \mu \quad (19.14)$$

$$\mu_i = \exp(\eta_i) \quad (19.15)$$

$$\eta_i = \beta_0 + \beta_1 Uninfected_i + \beta_2 Infected_LG_i + \beta_3 Infected_HG_i \quad (19.16)$$

1. The first line of the model is the stochastic part stating the response is modeled as a random Poisson variable with mean and variance μ (the rate parameter λ of the Poisson distribution).
2. The second line states the μ is the conditional mean or conditional expectation
3. The third line connects the conditional mean on the link scale (η) with the conditional mean on the response scale (μ)
4. The fourth line is the linear predictor, and includes three dummy variables.

Remember that the conditional mean is the expected/predicted/fitted/modeled value when $X = x_i$.

19.2.6 Model checking fits to count data

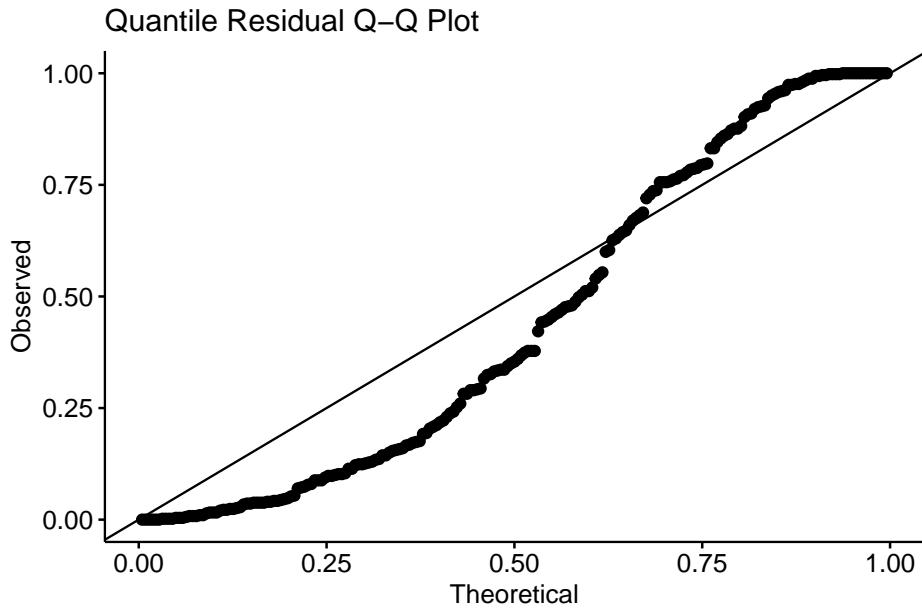
we use the fit model to check 1. the overall similarity of observed and theoretical distributions 2. if the observed distribution is over or under dispersed 3. if there are more zeros than expected by the theoretical distribution. If so, the observed distribution is **zero-inflated**

19.2.6.1 Model checking a GLM I – the quantile residual Q-Q plot

A quantile-quantile (Q-Q) plot is used to check overall similarity of the observed distribution with the distribution that would be expected under the model. An alternative to a Normal Q-Q plot for a GLM fit is a quantile residual Q-Q plot of observed vs. expected **quantile residuals**. The basic algorithm for this is

1. Use the model parameters to simulate p fake values of the response for each row of the data. This will be a $n \times p$ matrix of fake data where each column is a new, random sample of a population with parameters equal to that estimated for the observed data. For the Poisson, the parameter for each observation will be $\hat{\mu}_i$, the modeled value of observation i . For the negative binomial, the parameters will be $\hat{\mu}_i$ and the dispersion parameter k , which is the same for all observations.
2. For each observation (each row of the matrix of fake data), compute the fraction of simulated values smaller than the observed value of the response variable for that row. This fraction is the observed **quantile residual**, which ranges in value from 0 to 1. If the true data are distributed as that specified by the model, then quantile residuals will have a uniform distribution.
3. Sort the observed quantile residuals from smallest to largest and plot against theoretical quantile residuals from a uniform distribution. One could transform the quantile residuals to standard, normal residuals and then plot using a traditional Normal Q-Q plot but this step isn't necessary (if reported, a Normal Q-Q plot of transformed quantile residuals might confuse readers who failed to read the fine print).

Misconceivable – A common misconception is that if the distribution of the response approximates a Poisson distribution, then the residuals of a GLM fit with a Poisson distribution should be normally distributed, which could then be checked with a Normal Q-Q plot, and homoskedastic, which could be checked with a scale-location plot. Neither of these is true because a GLM does not transform the data and, in fact, the model definition does not specify anything about the distribution of an “error” term – there is no ε in the model definition above! This is why thinking about the definition of a linear model by specifying an error term with a normal distribution can be confusing and lead to misconceptions when learning GLMs.



The Q–Q plot using quantile residuals with a Poisson distribution indicates that the counts of *Diplostomum* larvae in the eyes of the threespine stickleback are not well approximated by a Poisson distribution – there are too many observed values near the ends of the expected tails, indicating the expected values are not spread out enough. This pattern emerges because the observed counts are overdispersed compared to a Poisson distribution.

19.2.6.2 Model checking a GLM II – a dispersion plot

If observed counts are Poisson distributed, then the **Pearson residuals** (r_i) and the residual degrees of freedom of the fit model (df) can be used to compute a dispersion statistic

$$\frac{\sum r_i}{df} \tag{19.17}$$

that has an expected value of 1. Instead of a formal hypothesis test of this statistic, I use a simulation approach and ask, “if the observed counts are Poisson distributed, what is the expected frequency distribution of this dispersion statistic?” and then use simulation to generate this expected distribution. The algorithm for this is

1. For each observation i , generate a random Poisson count using $\hat{\mu}$ as the parameter.
2. Fit the model and compute the dispersion statistic.

19.2. COUNT DATA EXAMPLE – NUMBER OF TREMATODE WORM LARVAE IN EYES OF THREE-SPINE STicklebacks

Observed vs. expected dispersion statistic

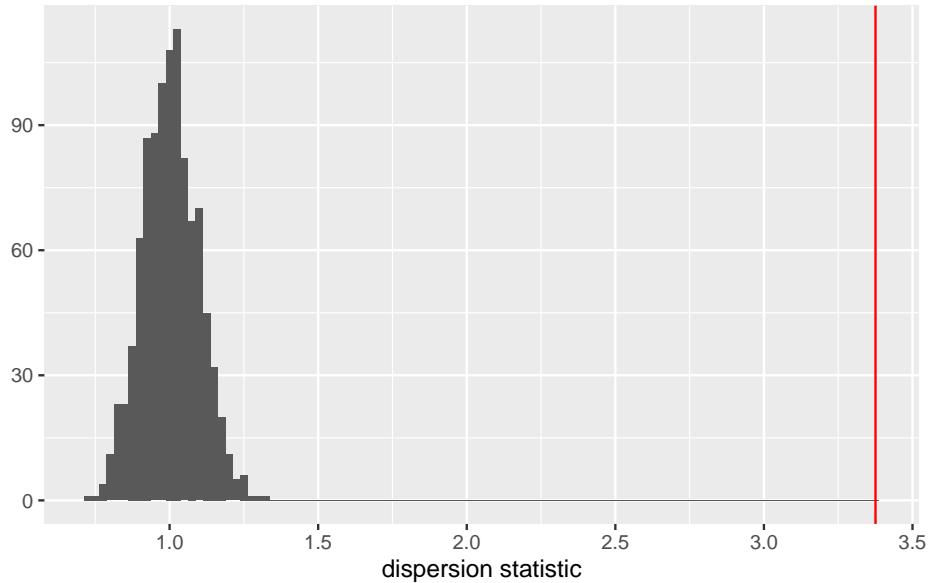


Figure 19.3: Observed vs. expected dispersion statistic. The observed statistic is marked by the red line. The histogram of expected statistics are from 1000 simulations of the observed data.

3. Repeat 1 and 2 N_{iter} times.

The plot below shows a histogram of the dispersion statistic computed for 1000 simulations of the worm data. The observed dispersion statistic is 3.4. The expected value is 1.0. The mean of the simulated values is 1.

19.2.7 Fitting a GLM with a Negative Binomial distribution to the worm data

The model is

$$Diplo_intensity \sim NB(\mu, k) \quad (19.18)$$

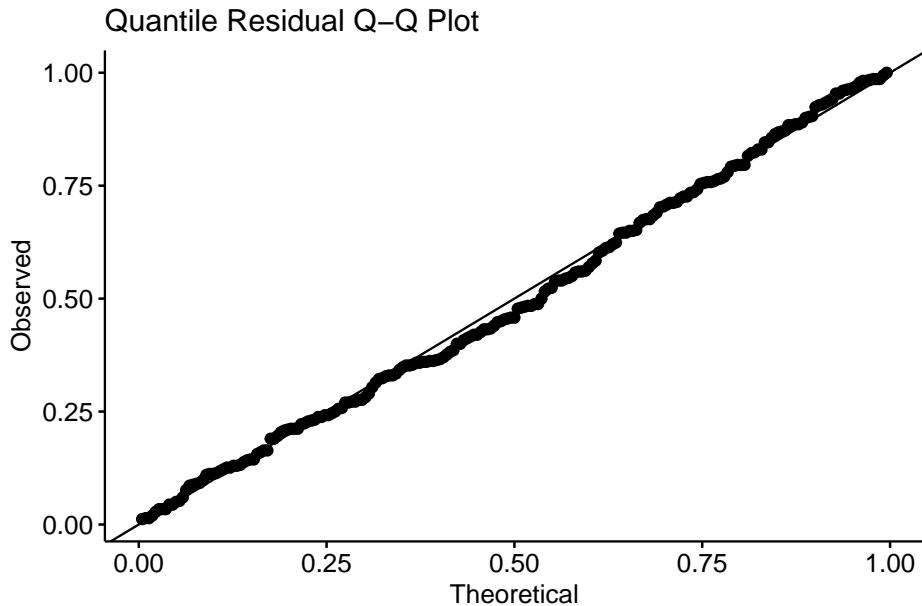
$$E(Diplo_intensity | Treatment) = \mu \quad (19.19)$$

$$\mu = \exp(\eta) \quad (19.20)$$

$$\eta = \beta_0 + \beta_1 Uninfected + \beta_2 Infected_LG + \beta_3 Infected_HG \quad (19.21)$$

This model specifies a negative binomial distribution but otherwise is just like that above specifying a Poisson distribution.

19.2.7.1 Model checking



19.2.7.2 Model means and coefficients

In a Generalized Linear Model of counts using either a Poisson or negative binomial distribution, modeled means, coefficients, and contrasts can be reported either on the link or response scale. Remember, the response scale is a count, while the link scale is a log(count).

The modeled means on the link scale are

```
##   Treatment     emmean      SE  df asymp.LCL asymp.UCL
##   Control      1.82 0.0804 Inf    1.66    1.98
##   Uninfected   1.50 0.1093 Inf    1.29    1.72
##   Infected LG  1.62 0.1362 Inf    1.36    1.89
##   Infected HG  2.36 0.0714 Inf    2.22    2.50
##
## Results are given on the log (not the response) scale.
## Confidence level used: 0.95
```

19.2. COUNT DATA EXAMPLE – NUMBER OF TREMATODE WORM LARVAE IN EYES OF THREE-SPINE STickleBACKS

While the means on response scale are

```
## Treatment      response      SE   df asympt.LCL asympt.UCL
## Control        6.18 0.497 Inf    5.28     7.24
## Uninfected     4.50 0.492 Inf    3.63     5.58
## Infected LG    5.07 0.691 Inf    3.89     6.63
## Infected HG    10.60 0.757 Inf   9.22    12.20
##
## Confidence level used: 0.95
## Intervals are back-transformed from the log scale
```

1. A mean on the response scale is simply the exponent of the mean on the link scale. For example, the mean of the Control treatment level on the response scale is $\exp(1.821408) = 6.180555$.
2. The CIs on the link scale are symmetric around the mean but those on the response scale are not. This is a feature, not a bug. Remember that counts are right skewed which means a CI will have a wider right than left interval. Check this!
3. If a plot includes a 1 SE error bar on the response scale, this is technically correct but it encourages the practice of computing CIs using the 2*SE rule of thumb. This rule breaks down for count data with right skewed distributions.
4. Plotting the response scale CIs is both technically correct and makes the 2*SE rule of thumb unnecessary.

The model coefficients on the link scale are

```
## contrast            estimate      SE   df asympt.LCL asympt.UCL z.ratio
## Uninfected - Control -0.317 0.136 Inf    -0.583   -0.0514 -2.339
## Infected LG - Control -0.197 0.158 Inf    -0.507    0.1126 -1.248
## Infected HG - Control  0.540 0.108 Inf     0.329    0.7504  5.019
## p.value
## 0.0193
## 0.2122
## <.0001
##
## Results are given on the log (not the response) scale.
## Confidence level used: 0.95
```

Backtransforming the coefficients (but not the intercept) to the response scale (using $\exp b_j$) results in a **response ratio**.

```

## contrast           ratio      SE   df asymp.LCL asymp.UCL z.ratio
## Uninfected / Control 0.728 0.0988 Inf      0.558      0.95 -2.339
## Infected LG / Control 0.821 0.1298 Inf      0.602      1.12 -1.248
## Infected HG / Control 1.715 0.1845 Inf      1.389      2.12  5.019
## p.value
## 0.0193
## 0.2122
## <.0001
##
## Confidence level used: 0.95
## Intervals are back-transformed from the log scale
## Tests are performed on the log scale

```

1. Note how the emmeans package reports the name of the term as the ratio of the coefficient term to the intercept term (the reference treatment level). Why are the coefficients tranformed to ratios on the response scale? Remember that a coefficient is a difference in conditional means and that $\exp(B - A) = \frac{\exp(B)}{\exp(A)}$. For a dummy variable as here (say “Infected HG”), the response ratio is

$$RR_{\text{Infected_HG}} = \frac{\overline{\text{Infected_HG}}}{\overline{\text{Control}}} \quad (19.22)$$

which give us the **relative effect** of Infected_HG compared to the Control. Relative effects could be reported as a response ratio in a table, or in the text it could be reported as a percent “Infected HG fish had 71.5% (95%CI: 38.9% - 111.8%) more *Diplostomum* larvae than Control fish.” Where do these percents come from? The percent effect is $100(RR_j - 1)$ larger than the reference mean if the $RR_j > 1$ or $100(1 - RR_j)$ smaller than the reference mean if the $RR_j < 1$.

2. Backtransforming the intercept does not generate a ratio since the intercept on the link scale is not a difference. For the worm analysis, the intercept on the link scale is the mean count of the control group on the link scale and the backtransformed intercept is the mean count of the control group on the response scale.
3. Effects on the response scale are not additive but multiplicative! So, for example, the mean of the Infected HG treatment level on the response scale is $\overline{\text{Control}} * RR_{\text{Infected_HG}}$ (remember that with a linear model the mean would be $b_{\text{Control}} + b_{\text{Infected_HG}}$). Check and see if this works.

19.3 Working in R

Fitting a GLM to count data. The poisson family is specified with the base R `glm()` function. For negative binomial, use `glm.nb` from the MASS package

```
# poisson - less likely to fit to real biological data well because of overdispersion
fit <- glm(y ~ treatment, family = "poisson", data = dt)

# two alternatives to overdispersed poisson fit
# quasipoisson
fit <- glm(y ~ treatment, family = "quasipoisson", data=dt)
# negative binomial - more likely to fit to real biological data well
# note that "family" is not an argument since this function is used only to fit a negative binomial
fit <- glm.nb(y ~ treatment, data = dt)
```

Fitting a GLM to a continuous conditional response with right skew.
The Gamma family is specified with the base R `glm()` function.

```
fit <- glm(y ~ treatment, family = Gamma(link = "log"), data = dt)
```

Fitting a GLM to a binary (success or failure, presence or absence, survived or died) response

The binomial family is specified with base R `glm()` function.

```
# if the data includes a 0 or 1 for every observation of y
fit <- glm(y ~ treatment, family = "binomial", data = dt)

# if the data includes the frequency of success AND there is a measure of the total n
dt[, failure := n - success]
fit <- glm(cbind(success, failure) ~ treatment, family = "binomial", data = dt)
```

Fitting Generalized Linear Mixed Models Generalized linear mixed models are fit with `glmer` from the `lmer` package.

```
# random intercept of factor "id"
fit <- glmer(y ~ treatment + (1|id), family = "poisson", data = dt)

# random intercept and slope of factor "id"
fit <- glmer(y ~ treatment + (treatment|id), family = Gamma(link = "log"), data = dt)

# Again, negative binomial uses a special function
fit <- glmer.nb(y ~ treatment + (treatment|id), data = dt)
```

Another good package for GLMMs is `glmmTMB` from the `glmmTMB` package

```
# negative binomial
fit <- glmmTMB(y ~ treatment + (1|id), family="nbinom2", data = dt)

# nbinom1, the mean variance relationship is that of quasipoisson
fit <- glmmTMB(y ~ treatment + (1|id), family="nbinom1", data = dt)
```

19.3.1 Fitting a GLM to count data

Source publication: Benesh, D. P., & Kalbe, M. (2016). Experimental parasite community ecology: intraspecific variation in a large tapeworm affects community assembly. *Journal of Animal Ecology*, 85(4), 1004-1013.

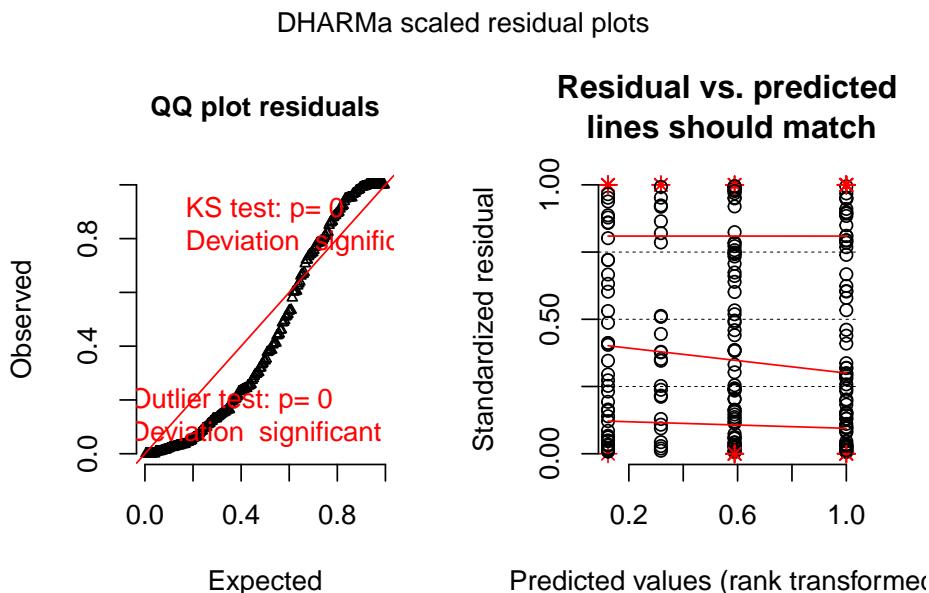
Source data URL: <https://datadryad.org/resource/doi:10.5061/dryad.bq8j8>

Source file: “Lab_exp.csv”

Poisson fit. A quantile residual Q-Q plot can be generated using the package DHARMa

```
fit.pois <- glm(Diplo_intensity ~ Treatment, family="poisson", data=worm)

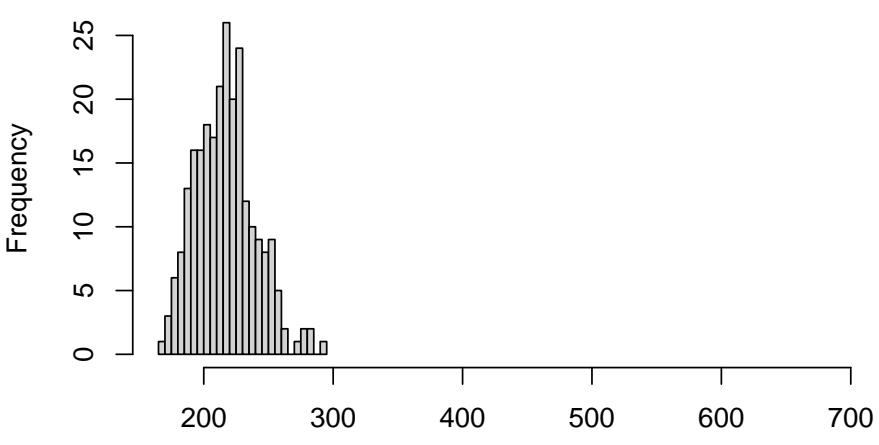
# from the DHARMa package
n_sim <- 250
simulationOutput <- simulateResiduals(fittedModel = fit.pois, n = n_sim)
plot(simulationOutput, asFactor = F)
```



A plot of the dispersion statistic can be generated using the object returned by the `SimulateOutput` function but with `refit = TRUE`, which refits a model each iteration. This refitting isn't necessary if only the quantile residuals are needed. The Dharma package does not divide the sum of squared Pearson residuals by the residual degrees of freedom and so the expected value of the statistic is df .

```
# from the DHARMA package
n_sim <- 250
simulationOutput <- simulateResiduals(fittedModel=fit.pois, n=n_sim, refit=TRUE)
testDispersion(simulationOutput)
```

DHARMA nonparametric dispersion test via mean deviance residual fitted vs. simulated-refitted



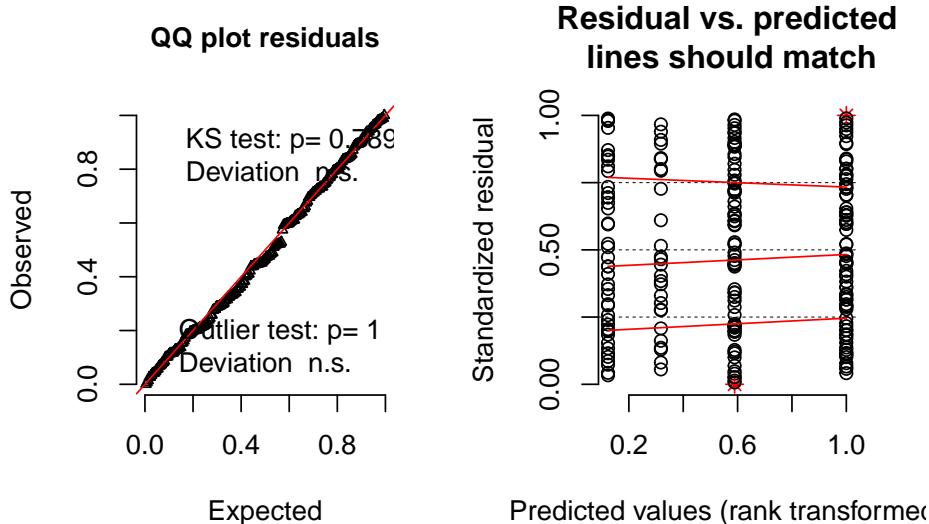
Simulated values, red line = fitted model. p-value (two.sided) = 0

```
##  
## DHARMA nonparametric dispersion test via mean deviance residual  
## fitted vs. simulated-refitted  
##  
## data: simulationOutput  
## dispersion = 3.3788, p-value < 2.2e-16  
## alternative hypothesis: two.sided
```

Negative binomial fit.

```
fit.nb <- glm.nb(Diplo_intensity ~ Treatment, data=worm)
# from the DHARMA package
simulationOutput <- simulateResiduals(fittedModel = fit.nb, n = n_sim)
plot(simulationOutput, asFactor = F)
```

DHARMA scaled residual plots



```
# link scale
emm <- emmeans(fit.nb, specs="Treatment")
emm

##   Treatment    emmean     SE  df asymp.LCL asymp.UCL
##   Control      1.82 0.0804 Inf   1.66      1.98
##   Uninfected   1.50 0.1093 Inf   1.29      1.72
##   Infected LG  1.62 0.1362 Inf   1.36      1.89
##   Infected HG  2.36 0.0714 Inf   2.22      2.50
##
## Results are given on the log (not the response) scale.
## Confidence level used: 0.95

summary(contrast(emm, method="trt.vs.ctrl", adjust="none"), infer=c(TRUE, TRUE))

##   contrast           estimate     SE  df asymp.LCL asymp.UCL z.ratio
##   Uninfected - Control -0.317 0.136 Inf   -0.583   -0.0514 -2.339
##   Infected LG - Control -0.197 0.158 Inf   -0.507    0.1126 -1.248
##   Infected HG - Control  0.540 0.108 Inf    0.329    0.7504  5.019
##
##   p.value
##   0.0193
##   0.2122
##   <.0001
##
## Results are given on the log (not the response) scale.
## Confidence level used: 0.95
```

```
# response scale
emm.response <- emmeans(fit.nb, specs="Treatment", type="response")
summary(contrast(emm, method="trt.vs.ctrl", adjust="none", type="response"), infer=c(TRUE, TRUE))

## contrast      ratio      SE  df asymp.LCL asymp.UCL z.ratio
## Uninfected / Control  0.728 0.0988 Inf    0.558     0.95 -2.339
## Infected LG / Control 0.821 0.1298 Inf    0.602     1.12 -1.248
## Infected HG / Control 1.715 0.1845 Inf    1.389     2.12  5.019
## p.value
## 0.0193
## 0.2122
## <.0001
##
## Confidence level used: 0.95
## Intervals are back-transformed from the log scale
## Tests are performed on the log scale
```

19.3.2 Fitting a generalized linear mixed model (GLMM) to count data

19.3.3 Fitting a generalized linear model to continuous data

19.4 Problems

Analyze the data that went into Fig 6B of Tena, A., Pekas, A., Cano, D., Wäckers, F. L., & Urbaneja, A. (2015). Sugar provisioning maximizes the biocontrol service of parasitoids. Journal of Applied Ecology, 52(3), 795-804.

1. Compute contrasts and CIs among all pairs of all three treatment levels
2. Make a better plot like 6b including 1) use the modeled mean instead of the simple group mean and 2) use the modeled CI of the mean instead of the SE computed within each group independently.

source URL: <https://datadryad.org/resource/doi:10.5061/dryad.bj001>

source file: “4_Parastism_Fig_6.csv”

Chapter 20

Linear models with heterogenous variance

20.1 gls

Part V: Expanding the Linear Model – Generalized Linear Models and Multilevel (Linear Mixed) Models

Chapter 21

Plotting functions (#ggplotsci)

```
# palettes
# http://mkweb.bcgsc.ca/colorblind/palettes.mhtml#page-container
pal_nature <- c(
  "#2271B2", # honolulu blue
  "#3DB7E9", # summer sky
  "#F748A5", # barbi pink
  "#359B73", # ocean green
  "#d55e00", # bamboo
  "#e69f00", # gamboge, squash, buttercup
  "#f0e442" # holiday,
)

pal_nature_black <- c(
  "#000000", # black
  "#2271B2", # honolulu blue
  "#3DB7E9", # summer sky
  "#F748A5", # barbi pink
  "#359B73", # ocean green
  "#d55e00", # bamboo
  "#e69f00", # gamboge, squash, buttercup
  "#f0e442" # holiday,
)

pal_nature_black_mod <- c(
  "#000000", # black
```

```

  "#3DB7E9", # summer sky
  "#e69f00", # gamboge, squash, buttercup
  "#359B73", # ocean green
  "#2271B2", # honolulu blue
  "#f0e442", # holiday,
  "#F748A5", # barbi pink
  "#d55e00" # bamboo
)

pal_nature_mod <- c(
  "#3DB7E9", # summer sky
  "#e69f00", # gamboge, squash, buttercup
  "#359B73", # ocean green
  "#2271B2", # honolulu blue
  "#f0e442", # holiday,
  "#F748A5", # barbi pink
  "#d55e00" # bamboo
)

pal_nature_alt <- c(
  "#AA0DB4", # barney
  "#FF54ED", # light magenta
  "#00B19F", # strong opal
  "#EB057A", # vivid rose
  "#F8071D", # vivid red
  "#FF8D1A", # dark orange
  "#9EFF37" # french lime,
)

# https://mikemol.github.io/technique/colorblind/2018/02/11/color-safe-palette.html
# https://thenode.biologists.com/data-visualization-with-flying-colors/research/

pal_okabe_ito <- c(
  "#E69F00",
  "#56B4E9",
  "#009E73",
  "#FOE442",
  "#0072B2",
  "#D55E00",
  "#CC79A7"
)

```

21.1 odd-even

This is a function used in the function to create a p-value table that is used in the plots. A user is unlikely to ever directly call this function.

```
odd <- function(x) x%%2 != 0
even <- function(x) x%%2 == 0
```

21.2 estimate response and effects with emmeans

```
estimate <- function(fit, # model fit
                      specs, # factor(s)
                      method = "revpairwise", # revpairwise
                      type = "response", # "link", "response"
                      adjust = "none" # p-value
) {
  # response table
  fit_emm <- emmeans(fit, specs = specs, type = type)
  response_table <- emm_table(fit_emm)

  # effect table
  fit_pairs <- summary(contrast(fit_emm,
                                 method = method,
                                 type = type,
                                 adjust = adjust,
                                 simple = "each",
                                 combine = TRUE),
                        infer = c(TRUE, TRUE)) %>%
    data.table()
  effect_table <- pairs_table(fit_pairs)

  return(list(
    response = response_table,
    effect = effect_table))
}
```

21.3 emm_table

```
emm_table <- function(fit_emm){
  table_out <- data.table(summary(fit_emm))
  if("response" %in% colnames(table_out)){
    # if the lm with log(y) or glm with log link, use " / "
    setnames(table_out,
              old = c("response"),
              new = c("emmean"))
  }
  if("asymp.LCL" %in% colnames(table_out)){
    # if the lm with log(y) or glm with log link, use " / "
    setnames(table_out,
              old = c("asymp.LCL", "asymp.UCL"),
              new = c("lower.CL", "upper.CL"))
  }
  return(table_out)
}
```

21.4 pairs_table

This function takes the output from emmeans::contrast and creates a table used by the plot functions to show p-values on the plot. A user would only call this function directly if they wanted to do some major modifications of the template here.

```
pairs_table <- function(fit_pairs){

  if("ratio" %in% colnames(fit_pairs)){
    # if the lm with log(y) or glm with log link, use " / "
    groups <- unlist(str_split(fit_pairs$contrast, " / "))
    setnames(fit_pairs, old = "ratio", new = "estimate")
  }else{
    # if lm use " - "
    groups <- unlist(str_split(fit_pairs$contrast, " - "))
  }

  if("asymp.LCL" %in% colnames(fit_pairs)){
    # if the lm with log(y) or glm with log link, use " / "
    setnames(fit_pairs,
              old = c("asymp.LCL", "asymp.UCL"),
              new = c("lower.CL", "upper.CL"))
```

```

}

# add the group1 and group 2 columns
fit_pairs[, group1 := groups[odd(1:length(groups))]]
fit_pairs[, group2 := groups[even(1:length(groups))]]

# for simple = each, beautify contrast column and group1, group2
if(which(colnames(fit_pairs)=="contrast") == 3){
  # replace "." with blank
  g_col <- colnames(fit_pairs)[1]
  x_col <- colnames(fit_pairs)[2]
  for(col in names(fit_pairs)[1:2]){
    set(fit_pairs,
        i=which(fit_pairs[[col]]=="."),
        j=col, value="")
  }

  fit_pairs[, contrast := paste0(get(names(fit_pairs)[1]),
                                 get(names(fit_pairs)[2]),
                                 ":",
                                 contrast)]
  fit_pairs[get(g_col) == "", group1 := paste0(get(names(fit_pairs)[1]),
                                             get(names(fit_pairs)[2]),
                                             ",",
                                             group1)]
  fit_pairs[get(g_col) == "", group2 := paste0(get(names(fit_pairs)[1]),
                                             get(names(fit_pairs)[2]),
                                             ",",
                                             group2)]
  fit_pairs[get(x_col) == "", group1 := paste0(group1,
                                                ",",
                                                get(names(fit_pairs)[1]),
                                                get(names(fit_pairs)[2]))]
  fit_pairs[get(x_col) == "", group2 := paste0(group2,
                                                ",",
                                                get(names(fit_pairs)[1]),
                                                get(names(fit_pairs)[2]))]

}

# create a column of nicely formatted p-values for display.
fit_pairs[, p := pvalString(p.value)]
}

```

21.5 gg_mean_error

`gg_mean_error` generates mean-and-CI plots that look like those published in most wet-bench experimental biology papers. I describe this more in the output below.

The arguments to the function are (the bold face arguments are required):

1. **data** the data.table
2. **x_col** the name of the treatment column (“diet” for this file) containing the treatment groups
3. **y_col** the name of response column (*not* the column of ratios!)
4. **x_label** = “none”, this is the label for the x-axis on the plot. “none” is the default (so no need to include) and results in no label (the plot looks better without it)
5. **y_label** = “Response (units)”, this is the label for the y-axis on the plot. You do want to replace the default with something meaningful and sense the y-variable is the *adjusted* response, it is important to include this in the label, so something like: **y_label** = “adj. SCAT (g)”
6. **dots** = “sina”, this controls how the dots are plotted. “sina” is clever.
7. **dodge_width** = 0.8, this controls how closely spaced the groups are in the plot
8. **adjust** = 0.5, this controls how wide the dots within each group
9. **p_adjust** = “none”, this controls p-value adjustment for multiple comparisons. This doesn’t matter for these data because there are only 2 groups in the diet treatment, so only 1 comparison (MR - CN).
10. **p_show** = 0, is the row number in **fit_pairs** with the p-values to show
11. **p_pos** = NULL, is the relative vertical position (bottom to top) of the p-values - the order matches the row index in **p_show**.

```
gg_mean_error <- function(data,
                           fit, # model fit from lm, lmer, nlme, glmmTMB
                           fit_emm,
                           fit_pairs,
                           x_col,
                           y_col,
                           g_col=NULL,
                           wrap_col=NULL,
                           x_label = "none",
                           y_label = "Response (units)",
                           g_label = NULL,
                           dots = "jitter",
                           dodge_width = 0.8,
                           adjust = 0.5,
                           p_show = 0,
```

```

p_pos = NULL){

dt <- data.table(data)
gg <- NULL

if(is.factor(dt[, get(x_col)]) == FALSE){
  dt[, (x_col) := factor(get(x_col))]
}
if(is.factor(dt[, get(g_col)]) == FALSE){
  dt[, (g_col) := factor(get(g_col))]
}

fit_emm_dt <- emm_table(fit_emm)
fit_pairs_dt <- pairs_table(data.table(fit_pairs))

if(g_col == x_col | is.null(g_col)){
  pd <- position_dodge(width = 0)
  fit_pairs_dt[, x_min_col := group1]
  fit_pairs_dt[, x_max_col := group2]
}else{
  pd <- position_dodge(width = dodge_width)
  fit_pairs_dt[, x_min_col := NA]
  fit_pairs_dt[, x_max_col := NA]
  if(is.null(g_label)){
    g_label <- g_col
  }
}

gg <- ggplot(data=dt, aes(x = get(x_col),
                           y = get(y_col),
                           color = get(g_col)))

# plot points
if(dots == "sina"){
  gg <- gg + geom_sina(alpha = 0.5,
                        position = pd,
                        adjust = adjust)
}
if(dots == "jitter"){
  gg <- gg + geom_point(alpha = 0.5,
                        position = "jitter")
}
if(dots == "dotplot"){
  gg <- gg + geom_dotplot(binaxis='y',

```

```

                stackdir='center',
                alpha = 0.5,
                position = pd)

}

# plot means and CI
gg <- gg +
  geom_errorbar(data = fit_emm_dt, aes(y = emmean,
                                         ymin = lower.CL,
                                         ymax = upper.CL,
                                         color = get(g_col)),
                 width = 0,
                 position = pd
  ) +
  geom_point(data = fit_emm_dt, aes(y = emmean,
                                      color = get(g_col)),
              size = 3,
              position = pd
  ) +
  # aesthetics
  ylab(y_label) +
  scale_color_manual(values=pal_nature_mod,
                     name = g_col) +
  theme_pubr() +
  theme(legend.position="top") +
  NULL

if(g_col == x_col){
  gg <- gg + theme(legend.position="none")
}

if(is.null(g_label)){
  gg <- gg + guides(color = guide_legend(title=NULL))
}else{
  gg <- gg + guides(color = guide_legend(title=g_label))
}

if(sum(p_show) > 0){ # show p-values
  # get x positions for p-values
  # [[3]] is 3rd layer which is means
  if(is.na(fit_pairs_dt[1, x_min_col])){
    gg <- gg +
      geom_text(data = fit_emm_dt, aes(x = 1.5, y = emmean - 0.5, label = paste0("p = ", round(p_value, 3))), color = "red")
  }
}

```

```

gg_data <- cbind(fit_emm_dt,
                  ggplot_build(gg)$data[[3]])

gg_data[, cell := paste(get(x_col), get(g_col), sep=",")]
match_it_ref <- match(fit_pairs_dt$group1, gg_data$cell)

fit_pairs_dt[, rowa := match(group1, gg_data$cell)]
fit_pairs_dt[, rowb := match(group2, gg_data$cell)]
fit_pairs_dt[, x_min_col := gg_data[rowa, x]]
fit_pairs_dt[, x_max_col := gg_data[rowb, x]]
}

if(is.null(p_pos)){
  p_pos <- 1:length(p_show)
}
max_y <- max(dt[, get(y_col)], na.rm=TRUE)
min_y <- min(dt[, get(y_col)], na.rm=TRUE)
increment <- 0.1*(max_y - min_y)
for(i in 1:length(p_pos)){
  pos <- p_pos[i]
  y_position <- max_y + increment*pos
  row <- p_show[i]
  gg <- gg +
    stat_pvalue_manual(fit_pairs_dt[row],
                        label = "p",
                        y.position=y_position,
                        xmin = "x_min_col",
                        xmax = "x_max_col",
                        tip.length = 0.01)
}
# make sure ylim includes p-value
y_hi <- max_y + 0.05*(max_y - min_y) +
  increment*max(p_pos)
y_lo <- min_y - 0.05*(max_y - min_y)
gg <- gg + coord_cartesian(ylim = c(y_lo, y_hi))
}

# remove x axis title
if(x_label == "none"){
  gg <- gg + theme(axis.title.x=element_blank())
} else{
  gg <- gg + xlab(x_label)}

```

```

gg

  return(gg)
}

```

21.6 gg_ancova

`gg_ancova` generates a classic “ANCOVA” plot. This function would be directly called by a user.

The arguments to the function are (the bold face arguments are required):

1. **data** the data.table
2. **x_col** the name of the treatment column (“diet” for this file) containing the treatment groups
3. **y_col** the name of response column (*not* the column of ratios!)
4. **cov_col**, the name of the column containing the covariate (“bw_02_05_18” for this file)
5. `cov_label` = “covariate (units)”, this is the label for the x-axis on the plot which is the name of the covariate column. You do want to replace the default with something meaningful. For these data, something like: `cov_label` = “Body Mass (g)”
6. `y_label` = “Response (units)”, this is the label for the y-axis on the plot. You do want to replace the default with something meaningful like: `y_label` = “SCAT (g)”
7. `add_p` = TRUE, the defaults adds the p-value of the treatment (`x_col`) effect
8. `p_pos` = “center”, controls the x-position of the p-value on the graph
9. `p_adjust` = “none”, this controls p-value adjustment for multiple comparisons. This doesn’t matter for these data because there are only 2 groups in the diet treatment, so only 1 comparison (MR - CN).

```

gg_ancova <- function(data,
                      fit, # model fit from lm, lmer, nlme, glmmTMB
                      fit_emm,
                      fit_pairs,
                      x_col,
                      y_col,
                      cov_col,
                      cov_label = "covariate (units)",
                      y_label = "Response (units)",
                      add_ci = TRUE,
                      add_p = TRUE,
                      p_show = NULL,

```

```

    p_pos = "center",
    p_adjust = "none"){

dt <- data.table(data)[, .SD, .SDcols = c(x_col, y_col, cov_col)]

dt <- data.table(fit$model)
if(is.factor(dt[, get(x_col)]) == FALSE){
  dt[, (x_col) := factor(get(x_col))]
}

fit_emm_dt <- emm_table(fit_emm)
fit_pairs_dt <- pairs_table(data.table(fit_pairs))

g <- x_col
y <- y_col
x <- cov_col

new_wide <- dt[, .(xmin = min(get(x)), xmax = max(get(x))), by = get(g)]
new_long <- melt(new_wide,
  measure.vars = c("xmin", "xmax"),
  value.name = "x")
setnames(new_long, old = c("get", "x"), new = c(g, x))
yhat <- predict(fit,
  new_long,
  interval = "confidence")
new_long <- cbind(new_long, yhat)
setnames(new_long, old = c("fit"), new = c(y))

gg <- ggplot(dt, aes(
  x = get(x),
  y = get(y),
  color = get(g)
)) +
  geom_point(size = 3) +
  labs(x = x,
    y = y,
    color = g) +
  NULL

g_levels <- levels(dt[, get(g)])
new_long[, get := get(g)] # not sure how to avoid this

# add ci ribbons

```

```

if(add_ci == TRUE){
  for(g_i in g_levels){
    gg <- gg +
      geom_ribbon(data = new_long[get == g_i,],
                   aes(ymin = lwr,
                        ymax = upr,
                        #                                     fill = get(g),
                        linetype = NA
                   ),
                   alpha = 0.2,
                   show.legend = FALSE)
  }
}

# add regression lines
for(g_i in g_levels){
  gg <- gg +
    geom_path(data = new_long[get == g_i,],
              aes(x = get(x),
                  y = get(y),
                  color = get(g)
                 ))
}

gg <- gg + # aesthetics
  xlab(cov_label) +
  ylab(y_label) +
  scale_color_manual(values=pal_nature_mod) +
  theme_pubr() +
  theme(legend.position="top") +
  NULL

if(is.null(p_show)){p_show := 1:nrow(fit_pairs_dt)}
if(sum(p_show) > 0){
  if(p_pos == "center"){
    x_p <- mean(range(dt[, get(cov_col)], na.rm = TRUE))
  }
  if(p_pos == "left"){
    x_p <- min(dt[, get(cov_col)], na.rm = TRUE) + diff(range(dt[, get(cov_col)], na.rm = TRUE)) / 2
  }
  if(p_pos == "right"){
    x_p <- max(dt[, get(cov_col)], na.rm = TRUE) - diff(range(dt[, get(cov_col)], na.rm = TRUE)) / 2
  }
}

```

```

pos_p <- 1:length(p_show)
max_y <- max(dt[, get(y_col)], na.rm = TRUE)
min_y <- min(dt[, get(y_col)], na.rm = TRUE)
increment <- 0.075*(max_y - min_y)

for(i in pos_p){
  y_p <- max_y + increment*i
  row <- p_show[i]
  gg <- gg + annotate("text",
    x = x_p,
    y = y_p,
    label = paste0(fit_pairs_dt[row, contrast],
      ": ",
      fit_pairs_dt[row, p]))
}

gg

return(gg)
}

```

21.7 gg_mean_ci_ancova

`gg_mean_ci_ancova` generates mean-and-CI plots that look like those published in most wet-bench experimental biology papers *except* that the dots are not the raw y-values but the y-values adjusted for the covariate (here “bw_02_05_18”). I describe this more in the output below.

The arguments to the function are (the bold face arguments are required):

1. **data** the data.table
2. **x_col** the name of the treatment column (“diet” for this file) containing the treatment groups
3. **y_col** the name of response column (*not* the column of ratios!)
4. **cov_col**, the name of the column containing the covariate (“bw_02_05_18” for this file)

5. x_label = “none”, this is the label for the x-axis on the plot. “none” is the default (so no need to include) and results in no label (the plot looks better without it)
6. y_label = “Response (units)”, this is the label for the y-axis on the plot. You do want to replace the default with something meaningful and sense the y-variable is the *adjusted* response, it is important to include this in the label, so something like: y_label = “adj. SCAT (g)”
7. dots = “sina”, this controls how the dots are plotted. “sina” is clever.
8. dodge_width = 0.8, this controls how closely spaced the groups are in the plot
9. adjust = 0.5, this controls how wide the dots within each group
10. p_adjust = “none”, this controls p-value adjustment for multiple comparisons. This doesn’t matter for these data because there are only 2 groups in the diet treatment, so only 1 comparison (MR - CN).

```
gg_mean_ci_ancova <- function(data,
                                x_col,
                                y_col,
                                cov_col,
                                x_label = "none",
                                y_label = "Response (units)",
                                dots = "sina",
                                dodge_width = 0.8,
                                adjust = 0.5,
                                p_adjust = "none"){

  dt <- data.table(data)[, .SD, .SDcols = c(x_col, y_col, cov_col)]
  ref_group <- levels(dt[, get(x_col)])[1]

  # center the covariate by the mean of the reference,
  # which means the intercept of the model will be EXP[Y]_ref
  # at the average value of the covariate
  mean_cov_ref <- mean(dt[get(x_col)==ref_group, get(cov_col)])
  dt[, cov_col_c := get(cov_col) - mean_cov_ref]
  cov_col_c <- paste0(cov_col, "_c")
  setnames(dt, old = "cov_col_c", new = cov_col_c)

  # two ways for computing adjusted y
  # "xside_xxx" is the "x side" of the formula
  # xside_1 doesn't use centered covariate but uses predicted value at mean covariate
  # xside_2 uses the centered covariate
  xside_1 <- paste(c(cov_col, x_col), collapse = " + ")
  xside_2 <- paste(c(cov_col_c, x_col), collapse = " + ")

  form1 <- formula(paste(y_col, "~", xside_1))
```

```

form2 <- formula(paste(y_col, "~", xside_2))

m1 <- lm(form1, data = dt)
m2 <- lm(form2, data = dt)

# using model 2 - centered covariate in model
b <- coef(m2)
dummy <- as.integer(dt[, get(x_col)]) - 1
dt[, y_cond := b[1] + b[3]*dummy + residuals(m2)]

# using model 1 - prediction at mean covariate
new_data <- copy(dt)
new_data[, bw_02_05_18 := mean_cov_ref]
dt[, y_cond2 := predict(m1, new_data) + residuals(m1)]

# emmeans
temp_emm <- emmeans(m1,
                      specs = x_col) %>%
  summary() %>%
  data.table()
emm_offset <- b[1] - temp_emm[get(x_col) == ref_group, emmean]

fit_emm <- emmeans(m1,
                      specs = x_col,
                      offset = emm_offset)

fit_pairs <- contrast(fit_emm,
                       method = "revpairwise",
                       adjust = p_adjust) %>%
  summary(infer = c(TRUE, TRUE))

fit_emm_dt <- data.table(summary(fit_emm))
fit_pairs_dt <- pairs_table(data.table(fit_pairs))

x_col1 <- x_col[1]
if(length(x_col)==2){
  g_col <- x_col[2]
}else{
  g_col <- x_col[1]
}

if(g_col == x_col1){
  pd <- position_dodge(width = 0)
}else{
  pd <- position_dodge(width = dodge_width)
}

```

```

}

gg <- ggplot(data=dt, aes(x = get(x_col1),
                           y = y_cond,
                           color = get(g_col)))
# plot points
if(dots == "sina"){
  gg <- gg + geom_sina(alpha = 0.5,
                        position = pd,
                        adjust = adjust)
}

if(dots == "jitter"){
  gg <- gg + geom_dotplot(alpha = 0.5,
                           position = pd)
}

# plot means and CI
gg <- gg +
  geom_errorbar(data = fit_emm_dt, aes(y = emmean,
                                         ymin = lower.CL,
                                         ymax = upper.CL,
                                         color = get(g_col)),
                width = 0,
                position = pd
  ) +
  geom_point(data = fit_emm_dt, aes(y = emmean,
                                      color = get(g_col)),
             size = 3,
             position = pd
  ) +
  # aesthetics
  ylab(y_label) +
  scale_color_manual(values=pal_nature_mod) +
  theme_pubr() +
  theme(legend.position="none") +
  NULL

# add p-value
y_positions <- max(dt[, y_cond]) +
  0.075*(max(dt[, y_cond]) - min(dt[, y_cond]))
gg <- gg +

```

```

stat_pvalue_manual(fit_pairs_dt, # only show sox effects
                   label = "p",
                   y.position=y_positions)

# make sure ylim includes p-value
ymax <- max(dt[, y_cond])
ymin <- min(dt[, y_cond])
y_hi <- ymax + 0.115*(ymax - ymin)
y_lo <- ymin - 0.05*(ymax - ymin)
gg <- gg + coord_cartesian(
  ylim = c(y_lo, y_hi),
)

# remove x axis title
if(x_label == "none"){
  gg <- gg + theme(axis.title.x=element_blank())
} else{
  gg <- gg + xlab(x_label)}

gg

return(gg)
}

```

21.8 gg_effects

`gg_effects` generates an “effects” plot, which is common in the clinical medicine but not experimental biology literature. This is a function that you would directly call in the analysis pipeline if you wanted plots like these. The arguments to the function are (the bold face arguments are required):

1. **data** the data.table
2. **x_col** the name of the treatment column (“diet” for this file) containing the treatment groups
3. **y_col** the name of response column (*not* the column of ratios!)
4. **cov_col**, the name of the column containing the covariate (“bw_02_05_18” for this file)
5. **x_label** = “none”, this is the label for the x-axis on the plot which is the name of the covariate column. You do want to replace the default with something meaningful. For these data, something like: `cov_label = “Body Mass (g)”`
6. **y_label** = “contrast”, this is the label for the y-axis on the plot. You do want to replace the default with something meaningful like: `y_label = “SCAT (g)”`

7. `p_adjust = "none"`, this controls p-value adjustment for multiple comparisons. This doesn't matter for these data because there are only 2 groups in the diet treatment, so only 1 comparison (MR - CN).

```
gg_effects <- function(data,
                        x_col,
                        y_col,
                        cov_col,
                        x_label = "none",
                        y_label = "contrast",
                        p_adjust = "none"){

  dt <- data.table(data)[, .SD, .SDcols = c(x_col, y_col, cov_col)]

  xside_1 <- paste(c(cov_col, x_col), collapse = " + ")

  form1 <- formula(paste(y_col, "~", xside_1))

  m1 <- lm(form1, data = dt)

  fit_emm <- emmeans(m1,
                      specs = x_col)

  fit_pairs <- contrast(fit_emm,
                         method = "revpairwise",
                         adjust = p_adjust) %>%
    summary(infer = c(TRUE, TRUE))

  fit_emm_dt <- data.table(summary(fit_emm))
  fit_effect <- pairs_table(data.table(fit_pairs))

  if(y_label == "contrast"){
    y_label <- fit_effect[, contrast]
  }

  fit_effect[, contrast := x_label]
  # # fit_effect[, contrast_pretty := paste(group1, "\n-", group2)]
  # # fit_effect[, contrast_pretty := paste(group1, "\n minus \n", group2)]
  # if(is.null(y_label)){
  #   y_label <- "Difference in means (units)"
  # }

  min_bound <- min(fit_effect[, lower.CL])
  max_bound <- min(fit_effect[, upper.CL])
  y_lo <- min(min_bound+min_bound*0.2,
```

```
    -max_bound)
y_hi <- max(max_bound + max_bound*0.2,
            -min_bound)
y_lims <- c(y_lo, y_hi)

gg <- ggplot(data=fit_effect, aes(x = fct_rev(contrast),
                                    y = estimate)) +
  geom_errorbar(aes(ymax=lower.CL,
                     ymax=upper.CL),
                width=0,
                color="black") +
  geom_point(size = 3) +
  geom_hline(yintercept=0, linetype = 2) +
  coord_flip(ylim = y_lims) +
  #coord_flip() +
  #scale_y_continuous(position="right") +
  theme_pubr() +
  ylab(y_label) +
  theme(axis.title.y = element_blank()) +
  NULL
gg

return(gg)
}
```

```
gg_multiple_effects <- function(){
}
```


Appendix 1: Getting Started with R

21.9 Get your computer ready

21.9.1 Start here

Watch this video. The links for installing R and R studio are in the next sections.

Andy Field's Installing R and RStudio

21.9.2 Install R

R is the core software. It runs under the hood. You never see it. To use R, you need another piece of software that provides a **user interface**. The software we will use for this is R Studio.

Download R for your OS

21.9.3 Install R Studio

R Studio is a slick (very slick) **graphical user interface** (GUI) for developing R projects.

Download R Studio Desktop

21.9.3.1 Additional resources for installing R and R Studio**

On Windows

On a Mac

21.9.4 Install R Markdown

In this class, we will write code to analyze data using R Markdown. R markdown is a version of Markdown. Markdown is tool for creating a document containing text (like microsoft Word), images, tables, and code that can be output to the three modern output formats: html (web pages), pdf (reports and documents), and microsoft word (okay, this isn't modern but it is widely used).

R Markdown can output pdf files. The mechanism for this is to first create a LaTeX ("la-tek") file. LaTeX is an amazing tool for creating professional pdf documents. You do not need PDF output for BIO 414/513. The directions for installing R Markdown include directions for installing LaTeX. This is optional, for this class, but I encourage you to do it.

Directions for installing R Markdown

21.9.5 (optional) Alternative LaTeX installations

On Windows

On a Mac

21.10 Start learning R Studio

R Studio Essentials, Programming Part 1 (Writing code in RStudio)

Getting Started with R Markdown

Andy Field's RStudio basics of R Markdown

Data Visualisation chapter from *R for Data Science*

Graphics for communication chapter from *R for Data Science*

Youtube: An Introduction to The data.table Package

Coursera: The data.table Package

Appendix 2: Online Resources for Getting Started with Statistical Modeling in R

Roughly, in order from most elementary to most advanced

Learning Statistics with R by Danielle Navarro and adapted to Bookdown (for web viewing) by Emily Kothe.

Statistical Thinking for the 21st Century by Russell A. Poldrack

Regression Models for Data Science in R by Brian Caffo

Broadening Your Statistical Horizons: Generalized Linear Models and Multi-level Models by J. Legler and P. Roback

Modern Statistics for Modern Biology

The Art of Data Science by Roger D. Peng and Elizabeth Matsui

Appendix 3: Fake Data Simulations

21.11 Performance of Blocking relative to a linear model

```
# use these to debug code
sigma=1
sigma_b0=1
sigma_b1=1
beta_0=10
beta_1=1
n_batch=4
n_sub samp=2
y_label="Y"
trt_levels=c("cn","tr")
block_label="block"

fake_lmm_data <- function(iterations=1000, sigma=1, sigma_b0=1, sigma_b1=1, beta_0=10, beta_1=1,
  # this function is constrained to simulate a single treatment with two levels
  #
  #           arguments
  # iterations - number of datasets to generate
  # sigma: conditional error sd
  # sigma_b0: sd of random intercepts
  # sigma_b1: sd of random slope
  # beta_0: fixed intercept (mean of reference)
  # beta_1: fixed slope (difference tr - cn)
  # n_batch: number of batches
  # n_sub samp: number of observations per batch per treatment level
  # confound: FALSE is randomized complete block, TRUE is confounded case where
  #           there is only one treatment level per batch
```

```

#
#           output
# A single matrix with each dataset stacked. The datasets are identified by
# the first column ("data_id")

if(sigma_b0==0){sigma_b0 <- 1e-10}
if(sigma_b1==0){sigma_b1 <- 1e-10}
n_iter <- iterations

levels_per_batch <- ifelse(confound==FALSE, 2, 1)
fake_data <- data.table(
  data_id = rep(1:n_iter, each=n_batch*n_subsample*levels_per_batch),
  sigma = sigma,
  sigma_b0 = sigma_b0,
  sigma_b1 = sigma_b1,
  beta_0 = beta_0,
  beta_1 = beta_1,
  treatment = rep(rep(trt_levels, each=n_subsample), n_batch*levels_per_batch/2*n_iter),
  batch = rep(rep(paste0("batch_", 1:(n_batch)), each=n_subsample*levels_per_batch), n_iter),
  beta_0_j = rep(rnorm(n_batch*n_iter, mean=0, sd=sigma_b0), each=n_subsample*levels_per_batch),
  beta_1_j = rep(rnorm(n_batch*n_iter, mean=0, sd=sigma_b1), each=n_subsample*levels_per_batch),
  x = rep(rep(c(0, 1), each=n_subsample), n_batch*levels_per_batch/2*n_iter),
  e = rnorm(n_subsample*n_batch*levels_per_batch*n_iter, mean=0, sd=sigma)
)
fake_data[, y:= (beta_0 + beta_0_j) + (beta_1 + beta_1_j)*x + e]
setnames(fake_data, old=c("y", "batch"), new=c(y_label, batch_label))
fake_data[, treatment := factor(treatment)]
return(fake_data)
}

# depending on parameterization, can get many "failed to converge"
# and "isSingular" warnings
write_it <- FALSE

n_iter <- 5000
beta_1_i <- 0 # 0 = Type I, !0 = Power.
confound_i <- FALSE # FALSE is randomized complete block, TRUE is confounded
#case where there is only one treatment level per batch
n <- 3 # subsamples
k <- 8 # batches

# model_list <- c("lm_complete", "lm_mean", "lmm_slope", "lmm_inter")
model_list <- c("lm_complete", "lm_mean", "lmm_inter")

se <- matrix(NA, nrow=n_iter, ncol=length(model_list))

```

```

colnames(se) <- model_list
prob <- matrix(NA, nrow=n_iter, ncol=length(model_list))
colnames(prob) <- model_list
ci <- matrix(NA, nrow=n_iter, ncol=length(model_list))
colnames(ci) <- model_list

fd_set <- fake_lmm_data(n_iter,
                         sigma = 1,
                         sigma_b0 = 1, # 1 for big, 0.1 for small
                         sigma_b1 = 0.1,
                         beta_0 = 10,
                         beta_1 = beta_1_i,
                         n_batch = k,
                         n_subsample = n,
                         confound = confound_i)

for(iter in 1:n_iter){
  fd <- fd_set[data_id==iter,]

  m1 <- lm(y ~ treatment, data=fd)
  m2 <- lm(y ~ treatment, data=fd[, .(y=mean(y)), by=(treatment, block)])

  if("lmm_slope" %in% length(model_list)){
    m3 <- lmer(y ~ treatment + (treatment|block), data=fd)
    m3.pairs <- summary(contrast(emmeans(m3, specs="treatment"), method="revpairwise"), infer=c(TRUE))
  }
  m4 <- lmer(y ~ treatment + (1|block), data=fd)
  m4.pairs <- summary(contrast(emmeans(m4, specs="treatment"), method="revpairwise"), infer=c(TRUE))

  se[iter, "lm_complete"] <- coef(summary(m1))["treatmenttr", "Std. Error"]
  se[iter, "lm_mean"] <- coef(summary(m2))["treatmenttr", "Std. Error"]
  if("lmm_slope" %in% length(model_list)){
    se[iter, "lmm_slope"] <- coef(summary(m3))["treatmenttr", "Std. Error"]
  }
  se[iter, "lmm_inter"] <- coef(summary(m4))["treatmenttr", "Std. Error"]

  prob[iter, "lm_complete"] <- coef(summary(m1))["treatmenttr", "Pr(>|t|)"]
  prob[iter, "lm_mean"] <- coef(summary(m2))["treatmenttr", "Pr(>|t|)"]
  if("lmm_slope" %in% length(model_list)){
    prob[iter, "lmm_slope"] <- coef(summary(m3))["treatmenttr", "Pr(>|t|)"]
  }
  prob[iter, "lmm_inter"] <- coef(summary(m4))["treatmenttr", "Pr(>|t|)"]

  ci[iter, "lm_complete"] <- confint(m1)["treatmenttr", 2] -

```

```

confint(m1)[ "treatmenttr", 1]
ci[iter, "lm_mean"] <- confint(m2)[ "treatmenttr", 2] -
  confint(m2)[ "treatmenttr", 1]
if("lmm_slope" %in% length(model_list)){
  ci[iter, "lmm_slope"] <- m3.pairs[, "upper.CL"] - m3.pairs[, "lower.CL"]
}
ci[iter, "lmm_inter"] <- m4.pairs[, "upper.CL"] - m4.pairs[, "lower.CL"]
# m4.pairs.LT <- diffLsmeans(m4, which="treatment", ddf="Kenward-Roger")
# m4.pairs.LT[, "upper"] - m4.pairs.LT[, "lower"]
}

if(write_it ==TRUE){
  id <- paste(sample(c(letters, LETTERS), 4), collapse="")
  fn <- paste0("lmm_fd_beta1=", beta_1_i,
              "_confound=", confound_i,
              "_id=", id,
              ".txt")
  fp <- here("output", "chapter_lmm", fn)
  write.table(fd_set, fp, sep="\t", quote=FALSE, row.names=FALSE)
  fp <- here("output", "chapter_lmm", paste0("lmm_se-", id, ".txt"))
  write.table(se, fp, sep="\t", quote=FALSE, row.names=FALSE)
  fp <- here("output", "chapter_lmm", paste0("lmm_prob-", id, ".txt"))
  write.table(prob, fp, sep="\t", quote=FALSE, row.names=FALSE)
  fp <- here("output", "chapter_lmm", paste0("lmm_ci-", id, ".txt"))
  write.table(ci, fp, sep="\t", quote=FALSE, row.names=FALSE)
}

apply(se, 2, quantile, c(0.1, 0.5, 0.9))
apply(prob, 2, function(x) sum(x<0.05)/n_iter)
apply(ci, 2, quantile, c(0.1, 0.5, 0.9))

```